# AI for Business Research

LECTURE NOTES • SPRING 2026

## Reinforcement Learning & Agentic AI

**DOTE 6635**

CUHK Business School

Instructor: Renyu (Philip) Zhang

Department of Decisions, Operations and Technology
The Chinese University of Hong Kong

Course materials, code, and slides:

github.com/rphilipzhang/AI-PhD-S26

June 29, 2026

**Abstract**

These notes accompany *AI for Business Research* (DOTE 6635), the 2026 offering of a PhD course taught annually since 2024. Where the first two offerings centered on natural language processing and computer vision (2024) and on large language models and causal machine learning (2025), the 2026 edition is organized around **reinforcement learning** and **agentic AI**. The notes move from the role of prediction in business research, through the foundations of machine learning and deep learning, into the core of modern reinforcement learning—multi-armed bandits and Markov decision processes, model-free prediction and control, value-function approximation and deep reinforcement learning, policy-gradient and actor–critic methods, and the reinforcement-learning pipeline behind modern large language models. The final chapters turn to **agentic AI**—LLM agents, skills, the Model Context Protocol, context and harness engineering—and to a survey of frontier developments in early 2026. Throughout, the emphasis is on what these methods mean for *business research*: the questions they let us ask, the decisions they let us improve, and the mechanisms they let us understand.

*Course repository (materials, code, slides, and the LaTeX source of these notes):*
https://github.com/rphilipzhang/AI-PhD-S26

# Contents


Chapter 1: Introduction **13**

1.1 Introduction **13**

1.2 Course Evolution: Three Seasons of AI for Business Research **13**

1.3 About the Instructor **14**

1.4 The Thinking Game and the Power of AI Agents **15**
1.4.1 Demonstration of Coding Agents . . . . . 15

1.5 The Bitter Lesson **15**

1.6 Defining AI for Business Research **16**
1.6.1 The AI Flywheel . . . . . 17

1.7 Course Purpose and Objectives **18**
1.7.1 The Coverage vs. Depth Trade-off . . . . . 18
1.7.2 Distinguishing CS and Business Research Impact . . . . . 18

1.8 Prerequisites and Expectations **19**
1.8.1 Prior Knowledge Assumptions . . . . . 19
1.8.2 Important Warnings . . . . . 19

1.9 Course Format and Structure **19**
1.9.1 Weekly Session Format . . . . . 19
1.9.2 Group Work . . . . . 19

1.10 Coursework and Grading **20**

1.11 Course Materials and Resources **20**
1.11.1 Primary Resources . . . . . 20
1.11.2 Course Communications . . . . . 20
1.11.3 Python Tutorial Sessions . . . . . 20

1.12 Alternative Resources for Learning AI **21**

1.13 What is AI/ML? **21**

1.14 The Landscape of AI/ML for Business Research **22**
1.14.1 A Typical Applied Business Research Paper . . . . . 22
1.14.2 AI/ML as Data Source . . . . . 22

1.14.3 Issues with ML as Data Source . . . 22
1.14.4 LLM for Social Simulations . . . 23
1.14.5 AI/ML for Causal Inference . . . 23
1.14.6 AI/ML for Predictive Decision-Making and Optimization . . . 24
1.14.7 LLM for Operations Research . . . 24
1.14.8 Generative AI for Content and Decision . . . 24
1.14.9 ML as Subject . . . 24
1.14.10 ML for Structural Estimation . . . 25
1.14.11 OR for LLM Inference . . . 25

1.15 Tentative Course Schedule **25**

1.16 Conclusion **26**

Chapter 2: Prediction Problems in Business Research **26**

2.1 Introduction: The Ubiquity and Utility of Prediction **26**

2.2 Why We Care About Predictions **27**

2.3 Applications of Prediction in Research **28**
2.3.1 Macro Predictions . . . 29
2.3.2 Demand Forecasting . . . 29
2.3.3 Recommendation Systems . . . 29
2.3.4 Other Prediction Domains . . . 30

2.4 From Prediction to Decision **31**

2.5 When Do Predictions Make No Sense? **31**

2.6 Prediction vs. Estimation: A Broader Perspective **32**

2.7 Conclusion **33**

Chapter 3: Machine Learning Basics **33**

Abstract **33**

3.1 Introduction **34**

3.2 The Supervised Learning Framework **34**
3.2.1 Prediction vs. Inference . . . 35
3.2.2 Parametric vs. Non-Parametric Approaches . . . 36

3.3 Model Evaluation and the Bias-Variance Trade-off **36**
3.3.1 The Bias-Variance Decomposition . . . . . 37
3.3.2 Cross-Validation . . . . . 39
3.3.2.1 Python Implementation: K-Fold CV . . . . . 40

3.4 K-Nearest Neighbors **41**
3.4.1 Implementation Considerations . . . . . 42

3.5 Decision Trees **43**
3.5.1 The CART Algorithm . . . . . 43
3.5.2 Pruning to Prevent Overfitting . . . . . 44

3.6 Ensemble Methods **45**
3.6.1 Bagging (Bootstrap Aggregating) . . . . . 45
3.6.2 Random Forests . . . . . 46
3.6.3 Boosting . . . . . 46
3.6.3.1 Python Example: Random Forest vs. Boosting . . . . . 47

3.7 Advanced Topic: Causal Forests **48**

References **50**

Chapter 4: Introduction to Deep Learning **50**

Abstract **50**

4.1 The Framework of Supervised Learning and Optimization **51**
4.1.1 Gradient Descent: The Engine of Optimization . . . . . 51
4.1.2 Foundational Models as Illustrations . . . . . 53
4.1.3 Enhancements to Gradient-Based Optimization . . . . . 54

4.2 The Architecture and Theory of Deep Neural Networks **55**
4.2.1 The Connectionist Paradigm . . . . . 56
4.2.2 The Power of Representation: Universal Approximation . . . . . 56
4.2.3 Are Simple Multilayer Perceptrons Outdated? Applications in Business Research . . . . . 58
4.2.4 The Mechanics of Learning: Backpropagation and Regularization 59

4.3 The Computational Landscape of Deep Learning **61**
4.3.1 Deep Learning Software Libraries . . . . . 61
4.3.2 Computing Hardware: GPUs and Beyond . . . . . 63
4.3.3 Model Size, Training Time, and Computational Costs . . . . . 64

4.3.4 Scaling Laws and the Economics of Large Models . . . . . 65
4.3.5 Geopolitical Considerations: GPU Export Controls . . . . . 66

4.4 Conclusion **67**

4.5 Compute-Efficient Training with GPUs **67**
4.5.1 Hardware-Aware Implementation and Parallelism . . . . . 67
4.5.2 Batch Size, Gradient Accumulation, and Learning-Rate Scaling . . 68
4.5.3 Precision Reduction and Efficient Attention . . . . . 68
4.5.4 Optimization Hygiene: Schedules and Initialization . . . . . 68

Chapter 5: Introduction to Reinforcement Learning **69**

5.1 Introduction: The New Era of Artificial Intelligence **69**
5.1.1 Why Reinforcement Learning? . . . . . 70

5.2 Notation and Conventions **70**

5.3 The Simplest Form of Reinforcement Learning: Multi-Armed Bandits **71**
5.3.1 Naive Approaches and the Need for a Strategy . . . . . 72
5.3.2 Advanced Strategies: Optimism in the Face of Uncertainty . . . . 73
5.3.3 MAB in Action: Real-World Applications . . . . . 74

5.4 Sequential Decision Making: Markov Decision Processes **75**
5.4.1 The Components of an MDP . . . . . 75
5.4.2 From Markov Processes to MDPs . . . . . 76
5.4.3 Policies and Value Functions . . . . . 77
5.4.4 Finding the Optimal Policy . . . . . 78

5.5 Solving MDPs with Dynamic Programming **79**
5.5.1 Value Iteration . . . . . 79
5.5.2 Policy Iteration . . . . . 80
5.5.3 A Comparison of Dynamic Programming Methods . . . . . 80

5.6 An Alternative Perspective: Linear Programming **82**

5.7 Connections to Econometrics: Structural Estimation **83**

5.8 Conclusion **84**

Chapter 6: Model-Free Reinforcement Learning **84**

Abstract **85**

6.1 Introduction to Model-Free Reinforcement Learning **85**
6.1.1 Core Challenges of Reinforcement Learning . . . . . 85
6.1.2 Two Paradigms: RL for Planning vs. RL for Learning . . . . . 85
6.1.3 From Model-Based to Model-Free RL . . . . . 86

6.2 Monte Carlo Policy Evaluation (Model-Free Prediction) **87**
6.2.1 The Monte Carlo Approach . . . . . 87
6.2.2 MC Policy Evaluation . . . . . 87
6.2.3 First-Visit vs. Every-Visit MC . . . . . 88
6.2.4 Incremental MC Policy Evaluation . . . . . 89
6.2.5 Example: Blackjack (Policy Evaluation) . . . . . 90

6.3 Temporal-Difference Policy Evaluation (Model-Free Prediction) **91**
6.3.1 The Central Idea of TD Learning . . . . . 91
6.3.2 From MC to TD: The Intuition . . . . . 91
6.3.3 The TD(0) Algorithm . . . . . 92
6.3.4 MC vs. TD: A Comprehensive Comparison . . . . . 93
6.3.5 The Bias-Variance Trade-off . . . . . 93
6.3.6 Empirical Comparison: Random Walk Example . . . . . 95
6.3.7 $n$-Step Prediction: Bridging MC and TD . . . . . 96

6.4 Model-Free Control (Learning an Optimal Policy) **96**
6.4.1 The Control Objective and Why We Learn $Q(s, a)$ . . . . . 96
6.4.2 MC Control and the Exploration Problem . . . . . 97
6.4.3 GLIE (Greedy in the Limit with Infinite Exploration) . . . . . 98
6.4.4 TD Control: Learning $Q$ Online . . . . . 98
6.4.5 SARSA (On-Policy TD Control) . . . . . 99
6.4.6 $n$-Step SARSA and SARSA($\lambda$) . . . . . 99
6.4.7 Off-Policy Learning and Q-Learning . . . . . 100
6.4.8 SARSA vs. Q-Learning: The Cliff-Walking Intuition . . . . . 101

6.5 Conclusion **102**

Chapter 7: Deep Reinforcement Learning **102**

7.1 Where Are We? **102**

7.2 Value Function Approximation **103**
7.2.1 The Limits of Tabular RL . . . . . 103
7.2.2 The Idea of Function Approximation . . . . . 104
7.2.3 Value Function Approximation with an Oracle . . . . . 105

7.2.4 Linear Approximation with an Oracle . . . 106
7.2.5 Model-Free Evaluation with Approximation . . . 107

7.3 Generalized Policy Iteration with Value Function Approximation **108**
7.3.1 The Framework . . . 108
7.3.2 Model-Free Control with Linear Approximation . . . 109
7.3.3 Incremental Control Algorithms . . . 109
7.3.4 Convergence of Control Algorithms . . . 110

7.4 Going Beyond Linear: Deep Reinforcement Learning **111**
7.4.1 Motivation . . . 111
7.4.2 Deep Reinforcement Learning . . . 111

7.5 Deep Q-Network (DQN) for Atari **112**
7.5.1 The Breakthrough . . . 112
7.5.2 Architecture and Setup . . . 112
7.5.3 Experience Replay . . . 113
7.5.4 Fixed Q-Targets . . . 114
7.5.5 The Complete DQN Algorithm . . . 115
7.5.6 Performance of DQN on Atari . . . 116

7.6 Extensions to DQN **116**

7.7 Applications: Deep RL for Business Decision Making **117**
7.7.1 Dynamic Coupon Targeting . . . 117
7.7.2 Ensembling Experimentation Along the Customer Journey . . . 118
7.7.3 Sequential Targeting . . . 119
7.7.4 Career-Path Recommendations . . . 119

7.8 Policy-Based Methods **120**
7.8.1 From Value to Policy Approximations . . . 120
7.8.2 Value-Based RL vs. Policy-Based RL . . . 120
7.8.3 Pros and Cons of Policy-Based RL . . . 121
7.8.4 Policy Gradient for One-Step MDPs . . . 121
7.8.5 Policy Gradient for Multi-Step MDPs . . . 122
7.8.6 Taking the Gradient . . . 123
7.8.7 The REINFORCE Algorithm . . . 123
7.8.8 Policy Gradient vs. Maximum Likelihood . . . 124
7.8.9 The Policy Gradient Theorem . . . 124
7.8.10 Score Function . . . 125
7.8.11 Reducing the Variance of the Policy Gradient . . . 126

7.8.12 Vanilla Policy Gradient with Baseline . . . 127
7.8.13 Off-Policy Policy Gradient . . . 127
7.8.14 Reducing Variance with a Critic . . . 129
7.8.15 Compatible Function Approximation Theorem . . . 130
7.8.16 Simple Action-Value Actor-Critic (QAC) Algorithm . . . 131
7.8.17 Asynchronous Advantage Actor-Critic (A3C) . . . 131

7.9 Application: Deep RL for Inventory Control **132**

7.10 Modern Deep RL: From TRPO to PPO **133**
7.10.1 Recap: The Policy Gradient Landscape . . . 134
7.10.2 Issues with Vanilla Policy Gradient . . . 134
7.10.3 Relative Policy Performance . . . 135
7.10.4 Importance Sampling for Policy Optimization . . . 136
7.10.5 The Surrogate Objective . . . 137
7.10.6 Trust Region Policy Optimization (TRPO) . . . 137
7.10.7 Limitations of TRPO . . . 137
7.10.8 Proximal Policy Optimization (PPO) . . . 138
7.10.9 PPO-Clip . . . 139
7.10.10 PPO: Implementation Challenges . . . 141

7.11 Reinforcement Learning for Large Language Models **142**
7.11.1 The LLM Training Pipeline . . . 142
7.11.2 Post-training Components . . . 143
7.11.3 Reward Modeling . . . 143
7.11.4 The MDP Formulation of RLHF . . . 145
7.11.5 PPO for RLHF . . . 146
7.11.6 PPO v0: Susceptibility to Reward Hacking . . . 146
7.11.7 Reward Hacking and Goodhart's Law . . . 146
7.11.8 KL-Penalty and Pre-training Loss . . . 147
7.11.9 Full PPO Algorithm for RLHF . . . 149
7.11.10 Empirical Performance of RLHF . . . 149
7.11.11 RLHF Can Be (Too) Complex . . . 150
7.11.12 Direct Preference Optimization (DPO) . . . 150
7.11.13 DPO: The Final Objective . . . 151
7.11.14 Empirical Performance of DPO . . . 153
7.11.15 Application: DPO for Balancing Engagement and Polarization . . . 153
7.11.16 Chain-of-Thought (CoT) Reasoning . . . 154
7.11.17 Reinforcement Learning with Verifiable Rewards (RLVR) . . . 154
7.11.18 Group Relative Policy Optimization (GRPO) . . . 155

7.11.19 DeepSeek-R1: Scaling Up RLVR . . . 157
7.11.20 Emergence of Reasoning Through RLVR . . . 158

7.12 Conclusion **158**

Chapter 8: Agentic AI **161**

8.1 Where Are We? **161**

8.2 LLM Agents **162**
8.2.1 What Is an Agent? . . . 162
8.2.2 From LLM to Agents . . . 162
8.2.3 Conceptual Framework of LLM Agents . . . 163
8.2.4 The Agent as a Smart PhD with SOPs . . . 164

8.3 Skills and the Model Context Protocol (MCP) **165**
8.3.1 Agent Skills . . . 165
8.3.2 Skill Archetypes . . . 165
8.3.3 Anatomy of a Skill . . . 166
8.3.4 Progressive Disclosure . . . 167
8.3.5 Frontmatter: When Skills Are Triggered . . . 168
8.3.6 Nonambiguous Instructions . . . 169
8.3.7 The Model Context Protocol (MCP) . . . 170

8.4 OpenClaw: A Case Study in Agentic AI **172**
8.4.1 What Is OpenClaw? . . . 172
8.4.2 Behind OpenClaw's Viral Growth . . . 172
8.4.3 The Token Economy and Alibaba Token Hub . . . 173
8.4.4 Approachability vs. Deep Work . . . 175
8.4.5 Unified Context . . . 175
8.4.6 Self-Evolving Memory Engine . . . 176
8.4.7 The OpenClaw Flywheel . . . 176
8.4.8 The Lethal Trifecta . . . 177
8.4.9 Reimplementation: Reconstructing the Magic Safely . . . 178

8.5 Context Engineering **179**
8.5.1 Context Is the Key . . . 179
8.5.2 Why Context Engineering Matters: Context Rot . . . 180
8.5.3 Anatomy of Effective Context . . . 180
8.5.4 System Prompts: Finding the Right Altitude . . . 181
8.5.5 Token-Efficient Tool Design . . . 182

8.5.6 Progressive Information Retrieval vs. RAG . . . 182
8.5.7 Architecting for Long Horizons . . . 183
8.5.7.1 Compaction: Distilling the State . . . 183
8.5.7.2 Agentic Memory: Structured Note-Taking . . . 184
8.5.7.3 Sub-Agent Architectures . . . 185
8.5.8 Context Engineering Capability Map . . . 186
8.5.9 Formal Representation of Context Engineering . . . 186

8.6 Harness Engineering **188**
8.6.1 From Context Engineering to Harness Engineering . . . 188
8.6.2 The Engineering (and Research) Paradigm Shift . . . 188
8.6.3 Provide Agents a Map, Not a Manual . . . 189
8.6.4 Context, Context, and Context . . . 190
8.6.5 Become an AI Manager . . . 191

8.7 Conclusion **192**

Chapter 9: What's New in AI (Spring 2026) **194**

9.1 The AI Landscape in Early 2026: A Bird's-Eye View **195**
9.1.1 The Technology Tree Is Growing Fast . . . 195
9.1.2 2025 in Review: Nature's Ten and Karpathy's Reflections . . . 195
9.1.3 The Four Maintracks of AI Progress . . . 196
9.1.4 The AI Industry by the Numbers . . . 197
9.1.5 Key Model Launches . . . 198
9.1.6 The 2026 Stanford AI Index Report . . . 200

9.2 AI Coding and the Zero Marginal Cost of Code **202**
9.2.1 The AI-Native Cost Structure . . . 202
9.2.2 Quality of AI-Generated Code . . . 203

9.3 AI-Driven Scientific Replication and Reproducibility **204**
9.3.1 Complete Replication of a PNAS Paper . . . 204
9.3.2 Live Demo: Replicating a Published Figure in 10 Minutes . . . 205
9.3.3 Scaling Reproducibility . . . 206
9.3.4 Automating Policy Evaluation at Scale (APE) . . . 207
9.3.5 Will Peer Review Be Disrupted? . . . 209
9.3.6 Auto-Research Published in *Nature* . . . 210
9.3.7 AI Workflow for Statistical Package Development (StatsClaw) . . . 211
9.3.8 Social Science Simulation Automation (YuLan-OneSim) . . . 212

9.4 AI for Mathematics: From Putnam to Unsolved Problems **213**
9.4.1 AxiomProver and the Putnam Competition . . . 213
9.4.2 Terry Tao and AI for Erdos Problems . . . 214
9.4.3 First Proof and Google Aletheia . . . 214
9.4.4 LLM Agents for Stylized Modeling . . . 216

9.5 AI and the Labor Market: Displacement, Augmentation, and the Intelligence Crisis **217**
9.5.1 AI Starts to Replace Human Workers . . . 217
9.5.2 The Intelligence Crisis . . . 218
9.5.3 Anthropic's Labor Market Impact Research . . . 218

9.6 AI Assistance vs. Human Learning **220**
9.6.1 The Cognitive Offloading Problem . . . 220
9.6.2 Implications for Business Education . . . 221

9.7 Agentic AI: Social Networks, Commerce, and Workflows **222**
9.7.1 Agents-Only Social Networks . . . 222
9.7.2 Research on Moltbook . . . 222
9.7.3 Agentic E-Commerce . . . 223
9.7.4 Agentic Workflows for Academics . . . 224
9.7.5 The Agentic Economy . . . 226

9.8 The Business of AI: Economics, IPOs, and Open vs. Closed Models **228**
9.8.1 Can AI Companies Become Profitable? . . . 228
9.8.2 IPOs of Chinese AI Companies . . . 228
9.8.3 Open vs. Closed Models . . . 229
9.8.4 Doing AI without a PhD? . . . 230
9.8.5 Nvidia GTC 2026: The AI Infrastructure Stack . . . 232
9.8.6 The Iran War and AI Infrastructure . . . 232
9.8.7 Sora's Shutdown and the AI Business Model Question . . . 233
9.8.8 TurboQuant, Memory Prices, and Research Integrity . . . 234

9.9 AI in High-Stakes Domains: Medicine, Law, and Policy **236**
9.9.1 AI in Clinical Medicine . . . 236
9.9.2 AI in Legal Practice . . . 237
9.9.3 AI in Space Engineering . . . 237

9.10 Organizational Transformation in the AI Age **238**
9.10.1 The Anthropic Hive Mind, and Its Discontents . . . 239
9.10.2 Resolving Organizational Frictions . . . 240

9.10.3 The AI Productivity Paradox and Meta's Pod Experiment . . . . . 241

9.11 Ethics, Privacy, and Governance **243**
9.11.1 Privacy: We Are Naked in Front of LLM Firms . . . . . 243
9.11.2 AI and Warfare . . . . . 243
9.11.3 Accountability of AI-Generated Content . . . . . 245
9.11.4 Generative Engine Optimization and the New "Soft Ads" . . . . . 246
9.11.5 Sycophantic AI and How AI Should Treat Humans . . . . . 247
9.11.6 Attack on Sam Altman's Home . . . . . 247

9.12 Towards AGI: Self-Improvement and the Road Ahead **248**
9.12.1 The Self-Improvement Prediction . . . . . 248
9.12.2 Three Questions Towards AGI . . . . . 251
9.12.3 Do LLMs Have Emotions? Anthropic's Neuroscience of Claude . 252
9.12.4 2026: Best of Times, Worst of Times . . . . . 253

9.13 Discussion Questions for PhD Researchers **254**

9.14 Sources and Further Reading **255**
9.14.1 Academic Papers and Research Reports . . . . . 255
9.14.2 Industry Reports and Blog Posts . . . . . 257
9.14.3 Open-Source Projects and Datasets . . . . . 258
9.14.4 News Coverage . . . . . 259

# Chapter 1: Introduction

These lecture notes are based on the introductory session of *DOTE 6635: Artificial Intelligence for Business Research*, taught by Renyu (Philip) Zhang at the Chinese University of Hong Kong (CUHK) Business School in Spring 2026. The notes are written in an academic style to facilitate deeper understanding and to serve as a reference for students throughout the semester.

## 1.1 Introduction

These lecture notes are based on the introductory session for the doctoral-level course DOTE 6635: Artificial Intelligence for Business Research, offered in the Spring 2026 semester at the Chinese University of Hong Kong (CUHK). This document provides a comprehensive overview of the course objectives, structure, and the broader landscape of AI applications in business research. The notes are written in an academic article style to facilitate deeper understanding and serve as a reference for students throughout the semester.

## 1.2 Course Evolution: Three Seasons of AI for Business Research

The Spring 2026 semester marks the third iteration of this course, with each "season" exploring different frontiers of artificial intelligence as they relate to business research.

**Season 1 (Spring 2024)** focused on foundational AI technologies, specifically **Natural Language Processing (NLP)** and **Computer Vision (CV)**. These technologies form the backbone of many modern AI applications, from sentiment analysis in marketing to image recognition in retail. The course materials and lecture notes from this season are available on GitHub and as a PDF document.

**Season 2 (Spring 2025)** advanced into the realm of **Large Language Models (LLMs)** and **AI-Powered Causal Inference**. This season reflected the explosive growth of generative AI and its implications for rigorous empirical research. Resources from this season are similarly available on GitHub and as lecture notes.

**Season 3 (Spring 2026)**, the current iteration, ventures into **Reinforcement Learning (RL)** and **Agentic AI**. These topics represent the cutting edge of AI research, where systems learn through interaction with their environment and can act autonomously to achieve complex goals. Approximately 80% of the content in this season is new compared to previous years, with only the introductory sessions on machine learning and deep learning remaining similar.

**Orienting vocabulary for readers new to AI/ML.** For a reader trained in econometrics, statistics, or operations research rather than computer science, it is worth pinning down the recurring technology labels before they appear in later chapters.

- **Natural Language Processing (NLP)**: computational methods that turn human-language text into structured measurements — sentiment, topics, similarity, named entities — so that a corpus of reviews, filings, earnings calls, or social-media posts becomes a set of variables one can place on either side of a regression.
- **Computer Vision (CV)**: the analogous task for images and video, e.g. recognizing objects, faces, or store layouts, or scoring the visual content of an advertisement.
- **Large Language Models (LLMs)**: very large neural networks trained to predict the next *token* (roughly, the next word-piece) over internet-scale text; the chat assistants you have used are LLMs. Most are built on the *Transformer* architecture of Vaswani et al. (2017), covered later in these notes.
- **Reinforcement Learning (RL)**: a framework in which an *agent* repeatedly observes a state, chooses an action, and receives a reward, learning a *policy* that maximizes expected cumulative (discounted) reward. This is precisely *sequential decision-making under uncertainty* — the object of dynamic programming and stochastic optimal control (Bellman, 1957; Sutton and Barto, 2018). RL is the data-driven counterpart: rather than assuming the transition dynamics are known and solving the Bellman equation exactly, it learns value functions and policies from sampled interaction.
- **Agentic AI**: AI systems (typically built around an LLM) that pursue a goal over many steps, plan, call external *tools* — running code, searching the web, querying a database — and act with limited human supervision, rather than producing a single response to a single prompt.

## 1.3 About the Instructor

Professor Renyu (Philip) Zhang describes himself as "a mostly harmless AI/data science scholar, teacher, and practitioner." His professional roles span academia and industry, serving as a professor at CUHK Business School, an economist at Kuaishou (a major Chinese short-video platform), and the founder of an AI startup. His educational background includes degrees from Peking University (2011) and Washington

University in St. Louis (2016), followed by an assistant professorship at NYU Shanghai from 2016 to 2021.

His research agenda centers on a fundamental question: **How can we leverage AI and data science to empower business decision-making, especially for digitalized online platforms?** This question motivates much of the course content and provides a unifying theme for the various AI applications discussed throughout the semester.

## 1.4 The Thinking Game and the Power of AI Agents

The lecture opened with a reference to "The Thinking Game,"[1] a concept associated with Sir Demis Hassabis, co-founder of DeepMind and a pioneer in artificial intelligence research.[2] This framing sets the stage for understanding AI as a tool for augmenting human cognitive capabilities.

### 1.4.1 Demonstration of Coding Agents

A compelling demonstration of modern AI capabilities was presented through the example of **coding agents**. Professor Zhang described a replication exercise he uses to assess new students and research assistants: replicating Figure 7 from one of his research papers on deep learning.[3] Traditionally, this task would require significant programming expertise and time. However, with the advent of AI coding agents such as OpenAI Codex, this task can now be accomplished with minimal human intervention. The demonstration showed how a coding agent could understand the requirements of the replication task, write the necessary code to reproduce the figure, and execute the code to generate the output. This example illustrates a broader trend in AI: the emergence of systems that can autonomously complete complex, multi-step tasks that previously required human expertise. These "agentic" AI systems represent a paradigm shift in how we interact with computers and will be a major focus of this course.

## 1.5 The Bitter Lesson

A foundational concept introduced in this lecture is **"The Bitter Lesson,"** articulated by Professor Richard Sutton, a pioneering figure in reinforcement learning research

[1] https://www.bilibili.com/video/BV1bbU8BREog/?t=1468s

[2] *The Thinking Game* (2024), directed by Greg Kohs, is a documentary that follows Hassabis and Google DeepMind — including the development of AlphaFold, the protein-structure-prediction system for which Hassabis and John Jumper shared the 2024 Nobel Prize in Chemistry — in their pursuit of artificial general intelligence (AGI).

[3] https://rphilipzhang.github.io/rphilipzhang/DeDL-nonblind.pdf

(Sutton, 2019).[4] This lesson encapsulates decades of experience in AI development and offers crucial guidance for researchers.

> "The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin." (Sutton, 2019)

The "bitter" aspect of this lesson lies in its implications for researchers. While incorporating domain-specific knowledge into AI systems can yield short-term gains and is often intellectually satisfying, such approaches tend to plateau over time. In contrast, general methods that can scale with increasing computational resources have consistently proven more effective in the long run. This principle has profound implications for business research. Domain-specific approaches may provide quick wins but can inhibit further progress, so research that scales with data and compute is more likely to have lasting impact. General methods, while potentially less elegant, tend to outperform specialized approaches as resources increase. The course will emphasize research approaches that align with this principle, focusing on methods that can leverage the exponential growth in computational power and data availability.

*Remark* (Why "bitter," and the link to scaling laws). The lesson is "bitter" because the very thing researchers most enjoy — encoding their hard-won domain knowledge into the model — is what tends to be outgrown. The empirical regularity behind it is today often stated quantitatively as a *scaling law*: across many tasks, a model's error falls as a smooth power law in the amount of compute, training data, and parameters (Kaplan et al., 2020). The point has a familiar flavor for an empirical economist. Hand-engineered features and tightly parameterized specifications impose strong, low-dimensional priors that help most when data are scarce; as the sample size and compute budget grow, flexible, weakly-structured estimators — the AI analog of nonparametric or sieve methods — tend to overtake them in out-of-sample performance. The bitter part is that the human ingenuity invested in the priors is precisely what is eventually outscaled by raw computation and data.

## 1.6 Defining AI for Business Research

The course defines **AI for Business Research** as the application of AI technologies to address, or facilitate the addressing of, business research questions. This definition encompasses a wide range of applications, from using machine learning to analyze consumer behavior to employing deep learning for demand forecasting.

[4] Sutton, together with Andrew Barto, received the 2024 ACM A.M. Turing Award — often described as the "Nobel Prize of computing" — for developing the conceptual and algorithmic foundations of reinforcement learning (announced March 2025).

### 1.6.1 The AI Flywheel

A central concept in understanding AI's role in business is the **AI Flywheel**, a virtuous cycle that drives continuous improvement. Its four components are summarized in Table 1.6.1.

**Table 1.6.1.** The four components of the AI Flywheel.

| Component | Description |
|---|---|
| Business Applications | Real-world use cases that generate value and data |
| Data | The fuel for training and improving AI models |
| AI Models | Algorithms that learn patterns and make predictions |
| Compute | Computational resources that enable model training and inference |

This flywheel effect creates a self-reinforcing cycle, as depicted in Figure 1.6.1: better AI models lead to better business applications, which generate more data, which in turn enables even better AI models. The course emphasizes that researchers should focus on work that can scale with data and compute, aligning with the principles of the Bitter Lesson.

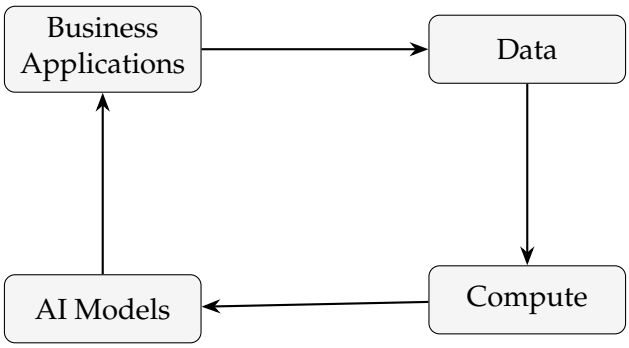


**Figure 1.6.1.** The AI Flywheel: a self-reinforcing cycle in which business applications generate data, which (together with compute) improves AI models, which in turn power better business applications.

*Remark* (The flywheel as increasing returns). Economists will recognize the flywheel as a mechanism of *increasing returns*. Data are a non-rival input that is strongly complementary to compute, so each turn of the cycle lowers the marginal cost — and raises the marginal return — of the next improvement. This is closely related to data network effects and to learning-by-doing, and it is a central reason why, on digital platforms, an early advantage in scale or data can compound into a durable competitive moat.

## 1.7 Course Purpose and Objectives

The primary purpose of this course is to provide students with a foundational understanding of the key concepts and methods in machine learning (ML) and artificial intelligence (AI) that are relevant to business research. The course aims to develop a basic understanding of fundamental ML and AI concepts and methods used in business research. It also seeks to help students understand how business researchers have utilized ML/AI and what managerial questions have been addressed. Finally, the course will cultivate a sense of what state-of-the-art AI/ML technologies can accomplish, both in the ML/AI community and potentially in students' own research fields.

### 1.7.1 The Coverage vs. Depth Trade-off

A fundamental challenge in designing this course is the trade-off between breadth and depth of coverage. The course takes a deliberate approach by providing a concise introduction to AI/ML topics relevant to applied business research. It covers enough necessary knowledge to help students understand key trade-offs and potentially invent new applied methods. The course also informs students about literature development in relevant domains and prepares them with the necessary sense to conduct rigorous business research using these methods. A key emphasis is placed on the combination of coding and theory for practical implementation.

### 1.7.2 Distinguishing CS and Business Research Impact

The course draws an important distinction between the factors that drive impact in computer science versus business research, as summarized in Table 1.7.1.

**Table 1.7.1.** Heuristic "impact formulas" contrasting computer science and business research.

| Field | Impact Formula |
|---|---|
| Computer Science | Problem Importance $\times$ Technical Novelty $\times$ Performance Improvement |
| Business Research | Problem Importance $\times$ Identification Rigor $\times$ Insight Novelty |

This distinction helps students understand how to position their research and what aspects to emphasize when applying AI methods to business questions.

## 1.8 Prerequisites and Expectations

### 1.8.1 Prior Knowledge Assumptions

The course assumes students have a working knowledge in calculus, linear algebra, and statistics. A working knowledge of Python programming is also expected, though AI coding assistants like Codex, Claude Code, and Cursor can now help with this. Finally, some basic sense of ML, causal inference, and econometrics is preferred but not strictly required.

### 1.8.2 Important Warnings

The course comes with several important caveats. At CUHK, there is an Economics course (ECON 5180) covering similar topics to Season 1 without the coding emphasis. This may be the most time-consuming course at CUHK by a wide margin. While ideas and methods will be discussed in class with demos, students will need coding skills (potentially "vibe coding"[5] with AI assistance) to complete homework and replication projects.

## 1.9 Course Format and Structure

### 1.9.1 Weekly Session Format

Each weekly session spans 2 hours and 45 minutes, structured as shown in Table 1.9.1.

**Table 1.9.1.** Format of each weekly session.

| Duration | Activity |
|---|---|
| 15 minutes | Homework discussions and review of previous content |
| 105 minutes | Theories and coding demos |
| 30 minutes | Student presentations |

### 1.9.2 Group Work

All coursework is completed in groups of at most **two students**. Students were required to register their group members, majors, and group name by 11:59 PM on January 7, 2026. Unregistered students would be matched with others based on their majors. Each student must evaluate their partner's contribution to all coursework.

[5] *Vibe coding* is an informal term, popularized by Andrej Karpathy in early 2025, for producing software by iteratively describing what you want to an AI coding assistant in natural language and largely accepting its generated code, rather than authoring each line by hand.

## 1.10 Coursework and Grading

The course includes four main components of coursework. Each group will be responsible for scribing the lecture notes for one session/topic throughout the semester. In addition, each group will replicate and present one paper per week. There will be weekly coding assignments, due two weeks after distribution, with the best five assignments counting toward the final grade. Finally, each group will complete a research project based on their own choice. All homework and the final project must be completed in **Python**. Detailed grading information is available in the course syllabus.

## 1.11 Course Materials and Resources

### 1.11.1 Primary Resources

The primary online resources for the course are listed in Table 1.11.1.

**Table 1.11.1.** Primary course resources.

| Resource | URL |
|---|---|
| GitHub Repository | https://github.com/rphilipzhang/AI-PhD-S26 |
| Google Sheet (Registration, Sign-ups, Submissions) | https://docs.google.com/spreadsheets/d/1YIwCR-X8VVLTv-OfGr7DqZJF6YIll-JE3SGI1q7JEus/edit?usp=sharing |
| Google CoLab (Code Demos) | https://drive.google.com/drive/folders/19806VHq6Vybrx-z4BbvV01sTMs9G5Hrh?usp=sharing |

### 1.11.2 Course Communications

The course communication channels are summarized in Table 1.11.2.

**Table 1.11.2.** Course communication channels.

| Channel | Details |
|---|---|
| Class Meeting | Tuesday, 9:30 AM - 12:15 PM @ CYT 209A (CYT LT5 on March 3) |
| Office Hour | By appointment @ CYT_911 |
| WeChat Group | Online discussion forum (200+ members) |
| Instructor Email | philipzhang@cuhk.edu.hk |
| Teaching Assistant | Ignis Jiang (ignisjiang@cuhk.edu.hk) |

### 1.11.3 Python Tutorial Sessions

Two optional Python tutorial sessions are offered online on Friday evenings (7:00 PM - 9:00 PM), taught by Xinyu Li, an MIS PhD Candidate at CUHK Business School. The first session on January 9, 2026, will cover Python Basics. The second session on January 16, 2026, will cover PyTorch Basics & FBA/DOT Server.

## 1.12 Alternative Resources for Learning AI

For students seeking additional learning resources, the course recommends several excellent options, listed in Table 1.12.1.

**Table 1.12.1.** Recommended external resources for learning AI/ML.

| Topic | Instructor | Resource |
|---|---|---|
| Basic ML Introduction | Andrew Ng | Coursera |
| Deep Learning Introduction | Andrew Ng | Coursera |
| Natural Language Processing | Chris Manning | Stanford CS224n |
| Computer Vision | Fei-Fei Li | Stanford CS231n |
| Deep Reinforcement Learning | Sergey Levine | UC Berkeley |
| Deep Learning Theory | Matus Telgarsky | UIUC |
| Machine Learning Fairness | Moritz Hardt | Fairness and Machine Learning |
| Large Language Models | Danqi Chen | Princeton COS 597G |
| Generative AI Short Courses | Various | DeepLearning.AI |

## 1.13 What is AI/ML?

Machine learning is defined as a **computer science subfield that automates computers to learn from data without being explicitly programmed**. This field goes by various names depending on the context and community, including Data Mining, Statistical Learning, and Data Science. The course will use these terms somewhat interchangeably while acknowledging their subtle differences in emphasis and methodology.

*Remark* (Machine learning as function estimation). For a reader fluent in regression, the most useful first mental model is this: most *supervised* machine learning estimates an unknown function $f$ mapping inputs $x$ to an output $y$ (a numeric value or a class label) by minimizing a loss over examples $\{(x_i, y_i)\}_{i=1}^{n}$ — exactly what OLS, logit, or a nonparametric regression already do. What differs is emphasis. Machine learning typically (i) uses far more flexible function classes (trees, kernels, neural networks), (ii) targets *out-of-sample* predictive accuracy rather than the consistency or standard error of one structural coefficient, and (iii) controls the resulting bias–variance tradeoff through explicit *regularization* and cross-validation. The slogan "learning from data without being explicitly programmed" simply means the parameters of $f$ are fit to data, not hand-coded by the analyst.

## 1.14 The Landscape of AI/ML for Business Research

A significant portion of the introductory lecture was devoted to mapping the landscape of how AI and ML are used in business research. This landscape can be organized into several distinct categories, each representing a different way that AI technologies intersect with business research questions.

### 1.14.1 A Typical Applied Business Research Paper

The structure of a typical applied business research paper can be understood through the lens of AI applications, as mapped in Table 1.14.1.

**Table 1.14.1.** Stages of a typical applied business research paper, and how AI enhances each stage.

| Stage | Traditional Approach | AI Enhancement |
|---|---|---|
| Setting & Problems | Manual identification | ML as Data Source |
| Data Generation | Survey, observational | ML for data extraction |
| Prediction/Optimization Models | Econometric models | ML for Predictive Decision-Making |
| Hypothesis Testing | Statistical tests | ML for Causal Inference |
| Mechanisms | Theoretical analysis | ML for Structural Estimation |
| Counterfactuals | Structural models | OR for LLM Inference |
| Decisions | Policy recommendations | Generative AI for Content/Decision |

### 1.14.2 AI/ML as Data Source

One of the most fundamental applications of AI in business research is using machine learning to transform unstructured data into analyzable information. Any recordable information that is not inherently numerical can potentially be analyzed with ML to answer business questions. This includes text (Natural Language Processing), images and video (Computer Vision), sound (Deep Learning), and even genetic information (Bioinformatics). Generative AI has significantly expanded the scope of analyzable information, enabling researchers to extract insights from data sources that were previously inaccessible. The benefits of using ML to understand unstructured data include cost reduction, scalability, objectivity, and easy integration into other systems. However, this approach also presents challenges related to measurement errors, interpretation, and potential biases in training data.

### 1.14.3 Issues with ML as Data Source

When using ML-generated data in empirical models, researchers must carefully consider how measurement errors can affect their analyses. The key parameter of interest

in any empirical model may be biased if the outcome, treatment, or control variables are generated through ML with error. For instance, if the treatment variable is generated with an error that is correlated with the outcome, this can lead to biased causal estimates. Similarly, errors in ML-generated control variables can affect the validity of the analysis. Readers from econometrics will recognize these concerns as instances of *measurement error*: when an ML-generated regressor stands in for an unobserved true variable, classical (mean-zero, independent) measurement error attenuates the corresponding coefficient toward zero, whereas error that is *correlated* with the outcome or the treatment — which arises easily when the same model or the same features are used to construct both an explanatory variable and the outcome — generally biases estimates in a direction that is hard to sign *a priori*. In these contexts, double machine learning[6] can be applied for effective debiasing.

### 1.14.4 LLM for Social Simulations

A cutting-edge application of AI in business research involves using Large Language Models for social simulations. Recent research has explored using LLMs to simulate human research subjects, providing an accessible data source for understanding human behavior. This includes building "digital twins" of individuals based on their survey responses and addressing challenges related to diversity, bias, sycophancy, alienness, and generalization in LLM simulations. Representative papers in this area include work on improving behavioral alignment in LLM social simulations (Kong et al., 2026) and creating datasets for building digital twins of over 2,000 people.

### 1.14.5 AI/ML for Causal Inference

Machine learning techniques are increasingly being integrated with causal inference methods to improve the rigor and scalability of empirical research. Key resources in this area include the CausalML Book, the Stanford GSB SI Lab ML-CI Tutorial, and previous course materials from AI-PhD-S25. Representative research includes work on deep learning-based causal inference for large-scale combinatorial experiments (Ye et al., 2024), influencer selection using follower elasticity, and targeting for long-term outcomes.

[6] Double/debiased machine learning (DML) estimates the nuisance components (for example, the conditional-outcome and the propensity/treatment models) with flexible ML, but combines them in a *Neyman-orthogonal* moment condition and uses cross-fitting (sample splitting), so that first-stage estimation error is first-order irrelevant for the target causal parameter. The orthogonalization step generalizes the partialling-out logic of the Frisch–Waugh–Lovell theorem. See V. Chernozhukov, D. Chetverikov, M. Demirer, E. Duflo, C. Hansen, W. Newey, and J. Robins, "Double/Debiased Machine Learning for Treatment and Structural Parameters," *The Econometrics Journal* 21(1), 2018, C1–C68.

### 1.14.6 AI/ML for Predictive Decision-Making and Optimization

This category represents the classic application of machine learning in business contexts. Examples include comparing customer choice models with machine learning for finding optimal product displays on platforms like Alibaba, using dynamic coupon targeting with batch deep reinforcement learning for livestream shopping, and developing cold start algorithms to improve market thickness on online advertising platforms (Ye et al., 2023). These applications demonstrate how ML can be used not just for prediction but for optimizing complex business decisions in real-time.

### 1.14.7 LLM for Operations Research

The intersection of Large Language Models and Operations Research represents an emerging frontier. Research in this area includes the development of ORLM, a customizable framework for training large models for automated optimization modeling, and CALM before the STORM, which aims to unlock native reasoning for optimization modeling using Large Reasoning Models (LRMs). These approaches aim to leverage the reasoning capabilities of LLMs to solve complex optimization problems that traditionally required specialized expertise.

### 1.14.8 Generative AI for Content and Decision

Generative AI is being applied to create content and inform decisions in various business contexts. This includes applying LLMs to generate ad text optimized for click-through rates in search engine advertising, developing unified generative models like OneRec for retrieval and ranking with iterative preference alignment in recommendation systems, and creating end-to-end generative recommendation and reinforcement learning-based bidding models for marketing services.

### 1.14.9 ML as Subject

AI and ML have themselves become subjects of academic study. Researchers are examining the economics of AI, machine-human collaboration, ML fairness and discrimination, the impact of ML on the labor market, data privacy, and the role of data and ML in Industrial Organization. Some are even considering AI as a new kind of species. Representative research includes experimental evidence on the productivity effects of generative AI (Noy and Zhang, 2023) and empirical studies of algorithmic bias in online advertising (Lambrecht and Tucker, 2019).

### 1.14.10 ML for Structural Estimation

Machine learning is being used to estimate parameters of structural models, which are widely used in economics and marketing. Key approaches include adversarial estimation, which uses generative adversarial networks for structural estimation (Kaji et al., 2023), and neural network estimators, which train neural nets to provide parameter estimates for structural econometric models (Wei and Jiang, 2024). These methods can offer computational advantages and robustness to model misspecification.

*Remark* (How these connect to tools you already know). A *generative adversarial network* (GAN) trains two networks against each other: a *generator* that produces synthetic data and a *discriminator* that tries to tell synthetic data from real data. In the adversarial estimator of Kaji et al. (2023), the structural economic model plays the role of the generator, and a flexible discriminator replaces the fixed set of moments used in simulated method of moments or indirect inference: one chooses the structural parameters so that simulated data are as hard as possible for the discriminator to distinguish from the observed data. The discriminator effectively *learns* which features of the data are most informative about the parameters, instead of the researcher fixing those moments in advance. The neural-network estimators of Wei and Jiang (2024) share the same spirit through a different route: a network is trained on data simulated from the structural model to approximate the inverse mapping from data (or summary statistics) back to parameters, which can be read as a sieve-style flexible approximation to an otherwise intractable likelihood or moment inversion.

### 1.14.11 OR for LLM Inference

Operations Research techniques are being applied to optimize the inference process for Large Language Models. This includes developing fluid-guided online scheduling with memory constraints for LLM inference optimization and fundamental modeling for LLM inference with exploding KV cache demands. These approaches address the practical challenges of deploying LLMs at scale, including memory management and scheduling efficiency.

## 1.15 Tentative Course Schedule

The course is organized into four main modules, as summarized in Table 1.15.1.

The schedule is tentative and subject to changes. Students should refer to the syllabus and GitHub repository for the most up-to-date information.

**Table 1.15.1.** Tentative organization of the course into four modules.

| Module | Sessions |
|---|---|
| Introduction to Supervised Learning | 1 |
| Introduction to Deep Learning | 1 |
| Reinforcement Learning | 6 |
| Agentic AI | 5 |

## 1.16 Conclusion

This introductory lecture has established the foundation for the course "DOTE 6635: Artificial Intelligence for Business Research." The course aims to equip doctoral students with the knowledge and skills necessary to engage with the rapidly evolving field of AI and to apply these powerful tools to their own research. Key takeaways from this introduction include the importance of the Bitter Lesson and focusing on methods that scale with data and compute, the AI Flywheel as a framework for understanding AI's role in business, the diverse landscape of AI applications in business research, and the emergence of Agentic AI as a transformative paradigm for autonomous problem-solving. By combining theoretical understanding with practical coding skills, students will be prepared to contribute to the next generation of AI-powered business research.

# Chapter 2: Prediction Problems in Business Research

This chapter examines the central role of prediction in business research and decision-making. We survey the breadth of prediction problems across economics, operations, finance, and the sciences, clarify how prediction relates to, and differs from, estimation and causal inference, and identify the conditions under which prediction is, or is not, the right tool for a given problem.

## 2.1 Introduction: The Ubiquity and Utility of Prediction

Prediction is a fundamental component of decision-making in a vast array of contexts, from macroeconomic policy to individual consumer choices. The ability to accurately forecast future outcomes is of paramount importance to researchers, policymakers, and business leaders alike. This chapter explores the critical role of prediction, its applications across various domains, and the theoretical underpinnings that distinguish it from related concepts such as estimation and causal inference.

## 2.2 Why We Care About Predictions

The importance of prediction can be understood through three primary lenses.

First, we are intrinsically interested in forecasting **macro-level outcomes** that affect society at large. These include, but are not limited to, population growth, election results, Gross Domestic Product (GDP), poverty rates, and the impact of tax policies.

Second, in many instances, **good predictions directly inform good decisions and policies**. For example, accurate weather forecasts can save lives and property, while precise demand forecasting can optimize supply chains and minimize waste. In the financial sector, predicting stock and asset returns is the bedrock of investment strategies. Recommendation systems, which predict user preferences, are now a ubiquitous feature of e-commerce and content platforms. In healthcare, predicting patient lifetime value (LTV) or the likelihood of disease can lead to better resource allocation and preventative care.

Third, prediction is a cornerstone of **causal inference**. To estimate the causal effect of an intervention, we must predict the counterfactual outcome, what would have happened in the absence of the intervention.[7] This is the foundation of many modern causal machine learning techniques, such as Double Machine Learning (DML), honest trees, and matrix completion methods.[8]

A key insight from Kleinberg et al. (2015) is the decomposition of a policy problem into prediction and causation components. Let $\pi(X_0, Y)$ denote a policy payoff that depends on a policy lever $X_0$ and on an outcome variable $Y$. The total change in the policy outcome with respect to the lever can be expressed as

$$\frac{d\,\pi(X_0, Y)}{dX_0} = \underbrace{\frac{\partial \pi}{\partial X_0}(Y)}_{\text{prediction}} + \underbrace{\frac{\partial \pi}{\partial Y} \cdot \frac{\partial Y}{\partial X_0}}_{\text{causation}}. \tag{2.2.1}$$

Equation (2.2.1) highlights that the overall policy effect is the sum of a direct effect (the *prediction* component, which captures how the payoff changes with the lever

[7] This is the *fundamental problem of causal inference* (Holland, 1986): for any single unit we observe the outcome under at most one treatment status, never both, so the unrealized potential outcome is missing data that must be imputed — that is, *predicted* — from comparable units. Causal inference is, in this precise sense, a prediction problem about counterfactuals.

[8] Briefly, for the reader meeting these tools for the first time. *Double Machine Learning* (Chernozhukov et al., 2018) estimates a treatment effect by first using flexible machine-learning models to predict both the outcome and the treatment from covariates, then relating the two sets of residuals to each other — a machine-learning generalization of the Frisch–Waugh–Lovell "partialling-out" logic that renders the effect estimate first-order insensitive (orthogonal) to small errors in those nuisance predictions. *Honest (causal) trees* (Athey and Imbens, 2016) use one subsample to decide *how* to split the covariate space and a *separate* subsample to estimate the effect within each leaf, restoring the valid confidence intervals that naive in-sample tree fitting destroys. *Matrix completion* treats the unobserved counterfactual outcomes as missing entries of a low-rank units-by-time matrix and imputes them, generalizing synthetic-control and panel-data estimators. In each case the causal estimand is recovered by *predicting* a nuisance or counterfactual quantity — the recurring theme of this chapter.

holding the outcome fixed) and an indirect effect mediated by the outcome variable $Y$ (the *causation* component, which captures how the lever changes the outcome and how the outcome in turn changes the payoff). Crucially, the prediction component requires only an accurate forecast of $Y$, whereas the causation component requires a well-identified causal effect $\partial Y / \partial X_0$ (Kleinberg et al., 2015).

**Example 2.2.1** (Umbrella vs. rain dance)**.** Kleinberg et al. (2015) motivate the decomposition with two weather-related decisions. Deciding whether to *carry an umbrella* is a pure *prediction* problem: the umbrella has no effect on whether it rains, so $\partial Y / \partial X_0 = 0$ and the causation term vanishes; all that remains is the prediction term, which simply needs an accurate forecast of the outcome $Y$ (rain). Deciding whether to perform a *rain dance*, by contrast, is a pure *causation* problem: the entire payoff hinges on whether the dance actually changes the probability of rain, $\partial Y / \partial X_0$ — a quantity that no amount of forecasting can supply, only a well-identified causal effect can. Most real policy levers — a central bank's rate change, a firm's price cut, a court's release rule — mix the two terms, but the decomposition makes precise which estimation tool each component demands.

*Remark* ("$\hat{y}$ problems" vs. "$\hat{\beta}$ problems"). Kleinberg et al. (2015) call decisions dominated by the prediction term *prediction policy problems*. It helps the econometrically trained reader to contrast the two estimation targets head-on. Classical econometrics is organized around the *coefficient* $\hat{\beta}$: we want a consistent, well-identified estimate of a structural or causal parameter (a treatment effect, an elasticity), together with valid standard errors for inference. The umbrella decision instead needs only a good *fitted value* $\hat{y}$: we care about the accuracy of the prediction itself, not about the unbiasedness or interpretability of any individual coefficient. The two goals genuinely pull in different directions — regularization (shrinkage) deliberately *biases* coefficients in order to lower out-of-sample prediction error, which is anathema to unbiased $\hat{\beta}$ estimation but exactly right for producing a good $\hat{y}$. Machine learning, the subject of the chapters that follow, is largely the science of producing good $\hat{y}$; recognizing whether a problem is a $\hat{y}$ problem or a $\hat{\beta}$ problem is the first step in choosing the right tool.

## 2.3 Applications of Prediction in Research

A wide range of research papers leverage prediction for novel insights. We organize representative examples by domain: macro predictions, demand forecasting, recommendation systems, and a selection of other high-impact prediction domains.

### 2.3.1 Macro Predictions

In the realm of macroeconomics, Jean et al. (2016) demonstrate the power of combining satellite imagery with machine learning to predict poverty. By using nighttime light intensity as a proxy for economic activity, their convolutional neural network (CNN)[9] model can explain up to 75% of the variation in local-level economic outcomes in developing countries. Similarly, van Binsbergen et al. (2023) construct a 170-year time series of economic sentiment by applying machine learning to a massive corpus of 200 million pages from U.S. local newspapers. This sentiment index proves to be a powerful predictor of macroeconomic fundamentals such as GDP, consumption, and employment growth.

### 2.3.2 Demand Forecasting

Demand forecasting has been revolutionized by the availability of more and wider data, as well as by the shift from linear models to more sophisticated machine learning techniques. Bajari et al. (2015) explore the use of machine learning methods for demand estimation, showing their superiority over traditional linear models. Cui et al. (2018) find that incorporating social media information can improve demand forecast accuracy by a significant margin of 12.85% to 23.23%. Cohen et al. (2022) introduce a novel Data Aggregation with Clustering (DAC) method to improve demand prediction by effectively aggregating data from various sources.

### 2.3.3 Recommendation Systems

Recommendation systems are a prime example of prediction in a business context. Farias and Li (2019) develop a tensor-recovery approach that incorporates side information to learn user preferences more effectively, leading to better personalization. Peng and Liang (2021) investigate the differences between view-based (view-also-view) and purchase-based (purchase-also-purchase) recommender systems. Their findings indicate that while view-based systems are more effective at generating views and sales for a wider range of products, purchase-based systems are more effective for cheaper products.

From a computer science perspective, Zhan et al. (2022) address the issue of duration bias in watch-time prediction for video recommendations on platforms such as

[9] A convolutional neural network is a neural-network architecture specialized for grid-structured inputs such as images. Rather than connecting every input pixel to every hidden unit, it slides small, *shared* filters across the image to detect local patterns (edges, textures, then higher-level structures), which drastically reduces the number of free parameters and builds in translation invariance. We introduce neural networks and convolutional architectures from scratch in the deep-learning chapter; for now it suffices to read "CNN" as "a flexible nonlinear predictor that takes an image as input."

Kuaishou. They propose a Duration-Deconfounded Quantile-based (D2Q) framework to mitigate this bias and improve recommendation quality. Covington et al. (2016) provide insights into the architecture of YouTube's massive recommendation system, which employs a two-stage process of candidate generation followed by deep ranking, as illustrated in Figure 2.3.1.

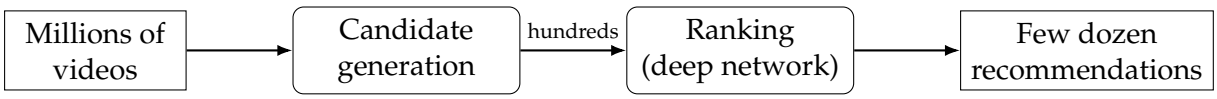


**Figure 2.3.1.** The two-stage YouTube recommendation pipeline of Covington et al. (2016): a high-recall candidate-generation network narrows millions of videos to hundreds, after which a high-precision deep ranking network orders the final shortlist.

The two-stage design is driven by a computational constraint familiar from screening and search problems: scoring *every* candidate with an expensive deep model for *every* user request is infeasible at web scale (millions of videos times hundreds of millions of users). A cheap, high-*recall* first stage therefore narrows the full corpus to a manageable shortlist (aiming to lose few of the truly relevant items), and an expensive, high-*precision* second stage then carefully ranks only that shortlist (aiming to order the survivors well). This is the same coarse-screen-then-fine-rank logic that appears in two-stage testing procedures and in funnel models of consumer search.

### 2.3.4 Other Prediction Domains

The application of prediction extends to numerous other fields. In finance, Gu et al. (2020) show that machine learning models, particularly trees and neural networks, can roughly double the performance of traditional regression-based strategies for empirical asset pricing. In medicine, a deep learning system called PANDA has demonstrated remarkable accuracy in detecting pancreatic cancer from non-contrast CT scans, achieving an AUC[10] of 0.986-0.996 and outperforming human radiologists (An et al., 2023). The development of AlphaFold 3 represents a monumental leap in predicting the three-dimensional structure of proteins, with profound implications for drug discovery and biology.[11]

[10] AUC denotes the area under the receiver operating characteristic (ROC) curve. It equals the probability that the model assigns a higher risk score to a randomly chosen positive case (here, a true cancer patient) than to a randomly chosen negative case; equivalently it is (a rescaling of) the Mann–Whitney–Wilcoxon rank statistic. An AUC of 0.5 is no better than random guessing and 1 is perfect ranking, so values near 0.99 indicate near-perfect discrimination between cases and non-cases, independent of any particular decision threshold.

[11] https://www.deepmind.com/alphafold

## 2.4 From Prediction to Decision

A crucial application of prediction is in informing high-stakes decisions. Kleinberg et al. (2018) study the use of machine predictions in judicial bail decisions. Their research shows that using a machine learning model to predict a defendant's flight risk could lead to a reduction in crime rates of up to 24.7% without changing the jailing rate, or a reduction in the jailing rate of up to 41.9% without an increase in crime. This highlights the potential for algorithmic decision support to improve outcomes in complex social systems.

## 2.5 When Do Predictions Make No Sense?

Despite its power, prediction is not a panacea. There are several scenarios where prediction models may be ineffective or even misleading.

1. **Lack of Importance:** The prediction target is not a sufficiently important macroeconomic, political, or natural outcome.
2. **Inaccuracy or Lack of Causality:** The prediction is neither accurate enough nor causally relevant for decision-making. As the decomposition in (2.2.1) shows, if the prediction of $Y$ is inaccurate, or if the causal link between the policy and the outcome is not well-identified, the resulting policy decision will be flawed.
3. **Ungrounded Counterfactuals:** The predictions of counterfactual outcomes are unreliable due to the violation of key assumptions for causal inference:
    - **Unconfoundedness (Conditional Independence Assumption, CIA):** The treatment assignment is independent of the potential outcomes, conditional on a set of observed covariates.
    - **Common Support (Overlapping Condition):** For any given set of covariates, there is a non-zero probability of being both treated and untreated.

We state these two requirements formally, as they recur throughout the causal-inference portions of the course.

**Assumption 2.5.1** (Unconfoundedness / CIA)**.** Let $W \in \{0,1\}$ denote treatment assignment, let $(Y(0), Y(1))$ denote the potential outcomes, and let $X$ denote observed covariates. Treatment is unconfounded if

$$\big(Y(0), Y(1)\big) \perp W \,\big|\, X.$$

**Assumption 2.5.2** (Common Support / Overlap)**.** With the propensity score $e(x) = \Pr(W = 1 \mid X = x)$, overlap requires

$$0 < e(x) < 1 \quad \text{for all } x \text{ in the support of } X.$$

When either Assumption 2.5.1 or Assumption 2.5.2 fails, the predicted counterfactual is no longer grounded in the data, and prediction-driven policy conclusions become unreliable.

## 2.6 Prediction vs. Estimation: A Broader Perspective

It is essential to distinguish prediction from the related concept of estimation. Hofman et al. (2021) provide a useful framework for thinking about the interplay between explanation (causal inference) and prediction in computational social science. They propose a $2 \times 2$ matrix that categorizes different research goals based on (i) whether the analysis involves an intervention and (ii) whether the focus is on specific features/-effects or on outcome prediction. This framework is summarized in Table 2.6.1.

**Table 2.6.1.** A framework for integrating explanation and prediction in computational social science (adapted from Hofman et al., 2021).

| | **Without Intervention** | **With Intervention** |
|---|---|---|
| **Specific Features / Effects** | Descriptive analysis or constructing new measurements | Causal inference or applied micro |
| **Outcome Prediction** | Predictive modeling or forecasting | Structural estimation, counterfactual simulation, and world models |

This framework clarifies that **predictive modeling and forecasting** are concerned with predicting outcomes *without* an intervention. In contrast, when an intervention is involved, the goal shifts to **structural estimation, counterfactual simulation, and building world models** in order to understand the causal impact of the intervention on the outcome.

*Remark* (What "world models" add to structural estimation). The lower-right cell of Table 2.6.1 groups together three ideas that the target reader can anchor to familiar econometrics. *Structural estimation* recovers deep, policy-invariant primitives (preferences, technology, costs) precisely so that one can simulate outcomes under interventions never realized in the data — the Lucas-critique-robust counterfactual exercise at the heart of structural industrial organization and labor economics. A *world model*

is the machine-learning analogue: a learned model of how an environment transitions from one state to the next under each possible action, which can then be "rolled forward" to simulate the consequences of a candidate policy. Both answer "with intervention, predict the outcome" questions; the difference is mainly one of flexibility and scale. The world-model idea returns later in the course as the foundation of *model-based* reinforcement learning.

## 2.7 Conclusion

Prediction is a powerful and versatile tool that is transforming business research and practice. From forecasting macroeconomic trends to personalizing user experiences and informing critical policy decisions, the applications of prediction are vast and growing. However, it is crucial to understand the limitations of prediction and the assumptions that underpin its use, particularly when it is used to inform causal inferences and policy interventions. By integrating prediction with explanation, as proposed by Hofman et al. (2021), researchers can build more robust and reliable models of the world, leading to better decisions and a deeper understanding of complex social and economic phenomena.

# Chapter 3: Machine Learning Basics

This chapter develops the fundamental concepts of supervised machine learning with an eye toward business-research applications. We move beyond a "black box" view to examine the theoretical underpinnings of supervised learning, the central role of model evaluation, and the mechanics of core algorithms, K-Nearest Neighbors, Decision Trees, and Ensemble Methods, before closing with an advanced treatment of Causal Forests for estimating heterogeneous treatment effects.

## Abstract

This document serves as a comprehensive guide to the fundamental concepts of machine learning, tailored for business research applications. Moving beyond the "black box" perspective, we explore the theoretical underpinnings of supervised learning, the critical importance of model evaluation, and the mechanics of core algorithms such as K-Nearest Neighbors, Decision Trees, and Ensemble Methods. The goal is to provide a rigorous yet accessible framework for understanding how these tools can be applied to predict outcomes and infer relationships in complex datasets.

## 3.1 Introduction

In the era of big data, machine learning has emerged as a pivotal tool for extracting actionable insights from vast information repositories. For business researchers, the ability not only to apply these algorithms but also to understand their internal mechanics is crucial. This chapter focuses on the supervised learning paradigm, in which the objective is to learn a mapping from input features to target outputs.

We begin by establishing the mathematical framework of supervised learning and the dual goals of prediction and inference. We then study the bias-variance trade-off, a central concept that governs the performance of all machine learning models. Finally, we examine specific algorithms, ranging from simple non-parametric methods to sophisticated ensemble techniques, and discuss their practical implementation and evaluation. Throughout, our treatment follows the modern statistical-learning canon (James et al., 2013; Hastie et al., 2009).

### Key Learning Objectives

- **Understand the bias-variance trade-off**: grasp the tension between model simplicity and flexibility.
- **Master cross-validation**: learn robust methods for estimating model performance on unseen data.
- **Apply core algorithms**: gain proficiency in K-Nearest Neighbors, Decision Trees, and Ensemble Methods.

## 3.2 The Supervised Learning Framework

Supervised learning involves building a statistical model to predict or estimate an output based on one or more inputs. Let $X$ represent the vector of input features (predictors) and $Y$ represent the target variable (response). We assume there is a systematic relationship between $X$ and $Y$, which can be expressed as

$$Y = f(X) + \epsilon. \tag{3.2.1}$$

The terms in (3.2.1) are defined as follows:

- $Y$ **(Target Variable)**: the outcome we wish to predict (e.g., sales revenue, customer churn).
- $X$ **(Feature Vector)**: the set of input variables or predictors (e.g., advertising spend, customer age).

- $f(X)$ **(Systematic Function)**: the true underlying relationship between the inputs and the output. This represents the signal we aim to learn.
- $\epsilon$ **(Irreducible Error)**: the random error term that captures noise, measurement errors, and the influence of unmeasured variables. It is assumed to have a mean of zero ($E[\epsilon] = 0$) and to be independent of $X$.

*Remark* (Connection to the econometric regression model). Equation (3.2.1) is the familiar regression model of graduate econometrics, but read with a different emphasis. Taking the conditional expectation of both sides and using $E[\epsilon \mid X] = 0$ gives $f(X) = E[Y \mid X]$: the systematic part $f$ is exactly the *conditional expectation function* (CEF) of $Y$ given $X$. Ordinary least squares posits that this CEF is linear, $f(X) = X^\top \beta$, and then concentrates on the finite-dimensional parameter $\beta$ (its sign, magnitude, and standard error). Machine learning instead treats $f$ itself—a possibly highly nonlinear function—as the object to be estimated, and judges $\hat{f}$ by how well it predicts $Y$ at *new* inputs rather than by the interpretability of any coefficient. The irreducible error $\epsilon$ plays the role of the regression disturbance: even the true CEF cannot predict $Y$ perfectly, because $Y$ still varies across units that share the same $X$.

### 3.2.1 Prediction vs. Inference

The estimation of $f$, denoted $\hat{f}$, serves two primary purposes in business research:

1. **Prediction**: when the focus is on predicting the value of $Y$ for a new, unobserved input $X$. In this context, $\hat{f}$ is treated as a "black box," and the primary metric of success is the accuracy of the prediction $\hat{Y} = \hat{f}(X)$.
2. **Inference**: when the goal is to understand the relationship between $X$ and $Y$. Researchers ask questions such as: Which predictors are most significant? Is the relationship linear or non-linear? In this case, the interpretability of $\hat{f}$ is paramount.

*Remark* (Two cultures, and "prediction policy problems"). Most graduate training in the social sciences emphasizes *inference*: we want a consistent, interpretable estimate of a structural or causal parameter (a price elasticity, a treatment effect), and we worry about identification, endogeneity, and standard errors. Machine learning, by contrast, was built around *prediction*: it will happily exploit a thousand correlated, uninterpretable features if doing so lowers out-of-sample error. Neither goal dominates—they answer different questions. Kleinberg et al. (2015) call the policy settings in which an accurate prediction (rather than a causal estimate) is what a decision-maker actually needs "prediction policy problems"—for example, predicting which patients will not survive long enough to benefit from a costly joint-replacement surgery. Recognizing

which of the two goals a study requires is the first methodological decision a researcher makes, because it dictates which tools, and which validity concerns, are relevant.

### 3.2.2 Parametric vs. Non-Parametric Approaches

Methods for estimating $f$ generally fall into two categories.

- **Parametric Methods**: these methods make an explicit assumption about the functional form of $f$ (e.g., assuming it is linear). This simplifies the problem to estimating a set of parameters. While easier to interpret and requiring less data, parametric methods suffer from high bias if the assumed form does not match the true underlying relationship.
  - *Examples*: Linear Regression, Logistic Regression.
- **Non-Parametric Methods**: these methods do not assume a specific functional form for $f$. Instead, they seek to fit the data as closely as possible. This flexibility allows them to model complex, non-linear relationships but requires a larger number of observations to avoid overfitting.
  - *Examples*: K-Nearest Neighbors, Decision Trees.

*Remark* (Relation to parametric and nonparametric econometrics). This dichotomy mirrors one the reader already knows. Parametric methods (OLS, logit, probit, structural demand models) reduce estimation of the infinite-dimensional function $f$ to estimation of a fixed, finite parameter vector; they are efficient and interpretable *when the functional form is correct*, but a misspecified form leaves a systematic approximation error—a bias—that does not vanish even as $n \to \infty$. Nonparametric methods (kernel regression, series/sieve estimators, splines, and the k-NN and tree methods of this chapter) let the data choose the shape of $f$ and so are consistent for a far wider class of truth, but they "spend" data on flexibility and therefore converge more slowly and demand larger samples. The bias-variance trade-off developed in Section 3.3 is precisely the formal statement of this tension.

## 3.3 Model Evaluation and the Bias-Variance Trade-off

Evaluating the performance of a machine learning model is critical. For regression problems, the standard metric is the **Mean Squared Error (MSE)**, which measures the average squared difference between the estimated values and the actual values:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}(x_i)\right)^2. \tag{3.3.1}$$

In (3.3.1):

- $n$: the total number of observations in the dataset.
- $y_i$: the actual observed value of the target variable for the $i$-th observation.
- $\hat{f}(x_i)$: the predicted value generated by the model for the $i$-th observation.
- $(y_i - \hat{f}(x_i))^2$: the squared residual (error) for a single observation. Squaring ensures that positive and negative errors do not cancel each other out, and it penalizes larger errors more heavily.

While the training MSE (calculated on the data used to fit the model) is useful, the true test of a model is its performance on unseen data, known as the **Test MSE**.

### 3.3.1 The Bias-Variance Decomposition

The expected test MSE for a given input point $x_0$ can be decomposed into three distinct components:

$$E\left[\left(y_0 - \hat{f}(x_0)\right)^2\right] = \text{Var}\left(\hat{f}(x_0)\right) + \left[\text{Bias}\left(\hat{f}(x_0)\right)\right]^2 + \text{Var}(\epsilon). \tag{3.3.2}$$

Equation (3.3.2) breaks the expected test error into three components:

1. $\text{Var}(\hat{f}(x_0))$ **(Variance)**: measures how much the model's prediction $\hat{f}(x_0)$ would change if we trained it on a different dataset. High variance implies the model is unstable and overfits the training data.
2. $[\text{Bias}(\hat{f}(x_0))]^2$ **(Squared Bias)**: measures the error introduced by approximating a complex real-world problem with a simplified model. High bias implies the model is too simple (underfitting) and misses key relationships.
3. $\text{Var}(\epsilon)$ **(Irreducible Error)**: the inherent noise in the system that no model can eliminate.

This decomposition highlights the fundamental trade-off in machine learning:

- **Variance**: refers to the amount by which $\hat{f}$ would change if we estimated it using a different training dataset. High variance indicates that the model is overfitting the training data, capturing noise rather than the underlying signal.
- **Bias**: refers to the error introduced by approximating a real-life problem (which may be extremely complicated) by a much simpler model. High bias indicates underfitting, where the model fails to capture important relations.
- **Irreducible Error**: the variance of the error term $\epsilon$, which cannot be reduced by any model.

**Where the decomposition comes from.** The three terms in (3.3.2) are not an assumption but an algebraic identity, and the derivation is one the reader can reconstruct from the mean-variance bookkeeping used for any estimator. Write $y_0 = f(x_0) + \epsilon_0$ with $E[\epsilon_0] = 0$, $\text{Var}(\epsilon_0) = \text{Var}(\epsilon)$, and $\epsilon_0$ independent of the training sample (hence independent of $\hat{f}$). The expectation is taken over both the noise $\epsilon_0$ at the query point and the randomness of the training set that produced $\hat{f}$. Adding and subtracting $E[\hat{f}(x_0)]$ and expanding the square,

$$\begin{aligned} E\Big[\big(y_0 - \hat{f}(x_0)\big)^2\Big] &= E\Big[\big(\epsilon_0 + f(x_0) - \hat{f}(x_0)\big)^2\Big] \\ &= \underbrace{\text{Var}(\epsilon_0)}_{\text{irreducible}} + E\Big[\big(f(x_0) - \hat{f}(x_0)\big)^2\Big] \\ &= \text{Var}(\epsilon) + \underbrace{\big(f(x_0) - E[\hat{f}(x_0)]\big)^2}_{[\text{Bias}(\hat{f}(x_0))]^2} + \underbrace{E\Big[\big(\hat{f}(x_0) - E[\hat{f}(x_0)]\big)^2\Big]}_{\text{Var}(\hat{f}(x_0))}. \end{aligned}$$

The cross terms vanish for two reasons the reader will recognize: $\epsilon_0$ is mean-zero and independent of $\hat{f}$, killing the first cross term; and $f(x_0) - E[\hat{f}(x_0)]$ is a constant that multiplies the mean-zero deviation $\hat{f}(x_0) - E[\hat{f}(x_0)]$, killing the second. The middle two terms are just the classical identity $\text{MSE}(\hat{\theta}) = [\text{Bias}(\hat{\theta})]^2 + \text{Var}(\hat{\theta})$ for an estimator $\hat{\theta}$, applied pointwise to the function value $f(x_0)$; the extra term $\text{Var}(\epsilon)$ is the noise floor that even a perfect $\hat{f} = f$ would still face.

*Remark* (Key Insight). As model complexity increases, variance tends to increase while bias tends to decrease. The optimal model is found at the "sweet spot" where the sum of squared bias and variance is minimized.

Figure 3.3.1 sketches this trade-off: total expected error is U-shaped in model flexibility, decomposing into a decreasing (squared) bias curve and an increasing variance curve, atop the constant floor of irreducible error.

*Remark* (A modern caveat: double descent). The U-shaped curve of Figure 3.3.1 is the *classical* picture and remains the right intuition for the models in this chapter. It is, however, not the whole story for the heavily over-parameterized models of modern deep learning. Belkin et al. (2019) document a "double descent" phenomenon: as the number of parameters grows past the point of exactly interpolating the training data (zero training error), the test error—after first rising as the classical curve predicts—can begin to *fall* again, sometimes below the classical sweet spot. Reconciling this with the trade-off above (roughly, that among the infinitely many zero-training-error fits the learning procedure implicitly selects a low-norm, low-variance one) is an active research area we return to when we discuss deep learning.

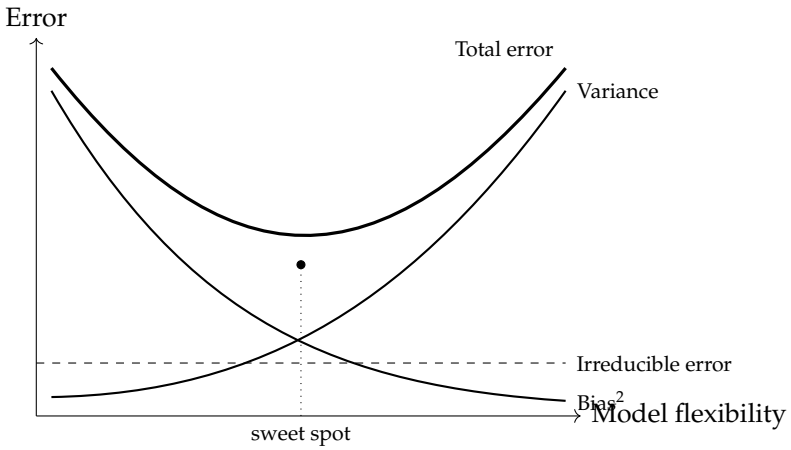


**Figure 3.3.1.** The bias-variance trade-off. Total expected test error (thick curve) is the sum of squared bias (decreasing), variance (increasing), and an irreducible-error floor. The optimal flexibility minimizes the sum.

### 3.3.2 Cross-Validation

To estimate the test error without setting aside a large portion of data solely for testing (which would deprive the model of valuable training data), we employ **cross-validation**. The most common variant is **K-Fold Cross-Validation**.

In this approach, the dataset is randomly divided into $K$ equal-sized folds. The model is trained on $K-1$ folds and validated on the remaining fold. This process is repeated $K$ times, and the results are averaged:

$$CV_{(K)} = \frac{1}{K}\sum_{k=1}^{K} \text{MSE}_k. \tag{3.3.3}$$

In (3.3.3):

- $K$: the number of folds (partitions) into which the data is split.
- $\text{MSE}_k$: the Mean Squared Error calculated on the $k$-th validation fold (the hold-out set) after training on the other $K-1$ folds.
- $CV_{(K)}$: the final cross-validation estimate, which is simply the average of the $K$ individual MSE scores.

**Choice of $K$.** Typically, $K = 5$ or $K = 10$ is chosen. This balances the bias-variance trade-off in the error estimate itself. A very large $K$ (such as Leave-One-Out Cross-Validation, where $K = n$) has low bias but high variance and, in general, high computational cost.[12]

[12]For ordinary least squares and other linear smoothers there is an important exception that connects directly to the econometric "hat matrix." Leave-one-out error then has the closed form $CV_{(n)} =$

**Why $K = n$ inflates variance.** The claim that leave-one-out CV has *higher* variance than 5- or 10-fold CV is initially counter-intuitive, since each leave-one-out fit uses almost all the data. The reason lies in correlation, not in any single fit. With $K = n$ the $n$ training sets are nearly identical—any two differ in only two observations—so the $n$ fitted models, and hence the $n$ hold-out errors being averaged, are very highly positively correlated. Averaging strongly correlated quantities removes little variance (the same fact that motivates *decorrelating* trees in Section 3.6). With $K = 5$ or 10 the training sets overlap less, the per-fold errors are less correlated, and their average is a more stable estimate of test error—at the cost of a little upward bias, because each model is now trained on noticeably fewer than $n$ observations.

*Remark* (Cross-validation as a data-driven information criterion). Cross-validation plays the role, for a flexible machine-learning model, that AIC and BIC play for a likelihood model: it is a tool for *model selection* and *hyperparameter tuning*—choosing $K$ in k-NN, the cost-complexity penalty $\alpha$ in a pruned tree, the number and depth of trees in a forest, or the regularization strength in a penalized regression. Whereas AIC and BIC approximate out-of-sample fit through an analytic complexity penalty that presumes a correctly specified likelihood, cross-validation estimates the out-of-sample loss directly by resampling and requires no distributional assumption. The price is computation: a $K$-fold search over a grid of hyperparameters refits the model many times.

The procedure is summarized in Algorithm 1.

**Algorithm 1** K-Fold Cross-Validation

1: **Input:** dataset $\{(x_i, y_i)\}_{i=1}^n$, number of folds $K$, learning procedure $\mathcal{A}$
2: Randomly partition the data into $K$ equal-sized folds $F_1, \ldots, F_K$
3: **for** $k = 1$ to $K$ **do**
4: Train model $\hat{f}_{-k} \leftarrow \mathcal{A}$ on all folds except $F_k$
5: Compute $\text{MSE}_k \leftarrow \frac{1}{|F_k|} \sum_{i \in F_k} \left(y_i - \hat{f}_{-k}(x_i)\right)^2$
6: **end for**
7: **Output:** $CV_{(K)} \leftarrow \frac{1}{K} \sum_{k=1}^K \text{MSE}_k$

#### 3.3.2.1 *Python Implementation: K-Fold CV*

Listing 1 performs 5-fold cross-validation with `scikit-learn`. Note that `scikit-learn` reports a *negative* MSE for scoring (so that higher is always better), which we negate to recover positive MSE values.

```
from sklearn.model_selection import cross_val_score
```

$\frac{1}{n} \sum_{i=1}^n \left((y_i - \hat{f}(x_i))/(1 - h_{ii})\right)^2$, where $h_{ii}$ is the leverage of observation $i$ (the $i$-th diagonal entry of the projection matrix $H = X(X^\top X)^{-1} X^\top$). The $n$ models therefore never have to be refit (James et al., 2013).

```python
from sklearn.linear_model import LinearRegression
from sklearn.datasets import make_regression

# Generate synthetic data
X, y = make_regression(n_samples=100, n_features=20, noise=0.1)

# Initialize model
model = LinearRegression()

# Perform 5-fold cross-validation
# Note: sklearn uses negative MSE for scoring, so we negate it to get positive
    MSE
scores = cross_val_score(model, X, y, cv=5, scoring='neg_mean_squared_error')
mse_scores = -scores

print(f"Mean MSE: {mse_scores.mean():.4f}")
```

**Listing 1.** Five-fold cross-validation for a linear-regression model.

## 3.4 K-Nearest Neighbors

The K-Nearest Neighbors (k-NN) algorithm is a simple yet powerful non-parametric method. It operates on the principle that similar data points likely have similar outcomes. Given a query point $x_0$ and a positive integer $K$, the algorithm identifies the $K$ points in the training data that are closest to $x_0$.

**For Classification.** The algorithm assigns $x_0$ to the class that is most common among its neighbors (a majority vote):

$$P(Y = j \mid X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j). \tag{3.4.1}$$

In (3.4.1):

- $P(Y = j \mid X = x_0)$: the estimated probability that the new point $x_0$ belongs to class $j$.
- $\mathcal{N}_0$: the set of the $K$ nearest neighbors to $x_0$ in the training data.
- $I(y_i = j)$: an indicator function that equals 1 if neighbor $y_i$ belongs to class $j$, and 0 otherwise. Essentially, this counts the "votes" for class $j$.

**For Regression.** The algorithm assigns $x_0$ the average value of its neighbors:

$$\hat{f}(x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} y_i. \tag{3.4.2}$$

Here, $\hat{f}(x_0)$ is simply the arithmetic mean of the target values $y_i$ of the $K$ nearest neighbors.

*Remark* (k-NN as local averaging / kernel regression). Both formulas say the same thing: to predict at $x_0$, average the training responses of the observations nearest $x_0$. This is the simplest instance of *local* estimation, and it is a close cousin of the kernel (Nadaraya–Watson) regression familiar from nonparametric econometrics. Kernel regression forms a *weighted* average $\hat{f}(x_0) = \sum_i w_i(x_0)\, y_i$ with weights $w_i(x_0)$ that decay smoothly with the distance $\|x_i - x_0\|$ through a bandwidth; k-NN is the special case of uniform weights $1/K$ on the $K$ closest points and zero on the rest. In both methods the tuning parameter—$K$ here, the bandwidth there—governs how local the average is, and hence the bias-variance balance: a narrow neighborhood (small $K$) tracks the data closely but is noisy, a wide one (large $K$) is smooth but biased. Seen this way, the self-attention mechanism at the heart of modern transformers is, formally, yet another kernel-weighted average—of learned "value" vectors rather than of raw responses.

### 3.4.1 Implementation Considerations

- **The Role of** $K$: the choice of $K$ controls the bias-variance trade-off. A small $K$ (e.g., $K = 1$) leads to a highly flexible model with low bias but high variance (a jagged decision boundary). A large $K$ smooths the decision boundary, increasing bias but reducing variance.
- **Feature Scaling**: because k-NN relies on distance metrics (typically Euclidean), it is critical to standardize features so that variables with large scales do not dominate the distance calculation.
- **Curse of Dimensionality**: k-NN suffers in high-dimensional spaces. As the number of features increases, data points become sparse, and the concept of "nearest" neighbor becomes less meaningful.

*Remark* (Quantifying the curse of dimensionality). A quick calculation makes the last point concrete. Suppose the features are spread uniformly through the unit hypercube $[0,1]^p$ and we want a neighborhood that contains a fraction $r$ of the data. A hypercubic neighborhood must then have side length $r^{1/p}$ along each dimension. To capture just $r = 1\%$ of the data, the side length is $0.01^{1/p}$: about 0.01 when $p = 1$, but 0.63 when $p = 10$ and 0.95 when $p = 100$. In high dimensions a "local" neighborhood holding

even a tiny fraction of the sample must span almost the entire range of every feature—so it is no longer local at all, and the premise that nearby points have similar outcomes loses its force. This is why purely distance-based methods such as k-NN degrade as $p$ grows, and why methods that adaptively ignore irrelevant directions (such as the trees and forests we turn to next) often fare better.

## 3.5 Decision Trees

Decision trees mimic human decision-making by learning a hierarchy of if/else questions. Structurally, a tree consists of a **root node** (containing all data), **internal nodes** (where splits occur based on features), and **leaf nodes** (where the final prediction is made). Figure 3.5.1 illustrates this structure.

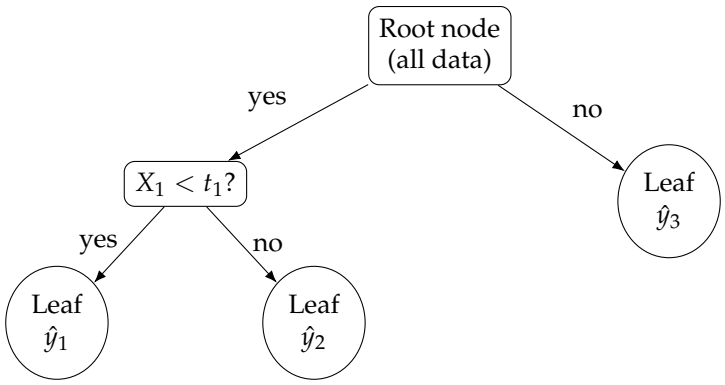


**Figure 3.5.1.** A decision tree recursively partitions the feature space. Internal nodes test feature thresholds; leaf nodes deliver predictions $\hat{y}_m$ (a class label or a regional mean).

### 3.5.1 The CART Algorithm

The Classification and Regression Tree (CART) algorithm builds the tree by recursively splitting the data into regions $R_1, \ldots, R_J$ to minimize impurity.

**How the splits are chosen.** Finding the single tree that globally minimizes error is computationally infeasible (the number of possible trees is astronomically large), so CART proceeds *greedily* by *recursive binary splitting*. Starting from the root, it searches over every feature $j$ and every candidate threshold $t$ and chooses the split $\{X_j < t\}$ versus $\{X_j \geq t\}$ that most reduces a node-impurity criterion; it then repeats the search separately within each child node, and so on, until a stopping rule (minimum leaf size, maximum depth) halts it. "Greedy" means each split is locally optimal given the splits already made and is never reconsidered—fast, but with no guarantee of global optimality. For a *regression* tree the impurity of a node is its within-node residual sum

of squares $\sum_{x_i \in R_m} (y_i - \hat{y}_{R_m})^2$, where $\hat{y}_{R_m}$ is the mean response in region $R_m$, and the prediction returned by a leaf is that regional mean. For *classification* the analogous node impurity is the Gini index or entropy defined next.

**Gini Index (Classification).** Measures the total variance across the $K$ classes. A small Gini index indicates a node that is predominantly one class (pure):

$$G = \sum_{k=1}^{K} \hat{p}_{mk} \, (1 - \hat{p}_{mk}). \tag{3.5.1}$$

In (3.5.1):

- $\hat{p}_{mk}$: the proportion of training observations in the $m$-th region (node) that belong to the $k$-th class.
- $G$: the Gini Index. A value of 0 indicates perfect purity (all observations in the node belong to a single class).

**Entropy (Classification).** An alternative measure of disorder or uncertainty:

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}. \tag{3.5.2}$$

Similar to the Gini index, the entropy $D$ is minimized (approaches 0) when the node is pure. It is maximized when the classes are perfectly mixed (e.g., a 50/50 split in a binary case).

*Remark* (Reading the two impurity measures). Both measures are largest when a node is a perfect mix of classes and zero when it is pure, so minimizing either pushes splits toward homogeneous leaves. They also have transparent statistical readings. For a single class $k$, the term $\hat{p}_{mk}(1 - \hat{p}_{mk})$ in the Gini index (3.5.1) is exactly the variance of a Bernoulli indicator for "belongs to class $k$"; the Gini index is the total such variance summed over the classes, which is why it is described as a measure of node "variance." Entropy (3.5.2) is the same quantity that appears in information theory and in the multinomial log-likelihood: up to a constant factor, the drop in entropy achieved by a split equals its *deviance reduction* (information gain), linking tree growing to the likelihood-based model comparison the reader already knows. In practice Gini and entropy select very similar trees.

### 3.5.2 Pruning to Prevent Overfitting

A major drawback of decision trees is their tendency to overfit by growing too deep and memorizing the training data. **Pruning** addresses this by reducing the size of the

tree. **Cost Complexity Pruning** introduces a penalty term $\alpha$ into the error function, penalizing the number of terminal nodes $|T|$:

$$\sum_{m=1}^{|T|} \sum_{x_i \in R_m} \left(y_i - \hat{y}_{R_m}\right)^2 + \alpha |T|. \tag{3.5.3}$$

This cost function balances two competing objectives:

1. **Model Fit**: the first term $\sum(y_i - \hat{y}_{R_m})^2$ measures the total squared error of the tree (how well it fits the training data).
2. **Model Complexity**: the second term $\alpha |T|$ penalizes the size of the tree, where:
   - $|T|$: the number of terminal nodes (leaves) in the tree.
   - $\alpha$ **(Alpha)**: a tuning parameter that controls the strength of the penalty. If $\alpha = 0$, we recover the full, unpruned tree. As $\alpha$ increases, we are forced to prune the tree to reduce $|T|$, leading to a simpler model.

*Remark* (Pruning as penalized model selection). The cost-complexity criterion (3.5.3) has exactly the shape of the penalized objectives the reader meets elsewhere: goodness-of-fit plus $\alpha$ times a complexity penalty. Here the penalty $\alpha |T|$ counts terminal nodes, just as AIC and BIC penalize the number of parameters and the Lasso (Tibshirani, 1996) penalizes the $\ell_1$ norm of the coefficients. Sweeping $\alpha$ upward from 0 generates a nested sequence of subtrees, from the full tree down to the root, and $\alpha$ itself is chosen by cross-validation (Section 3.3.2)—a concrete instance of the model-selection role described there. Larger $\alpha$ buys lower variance at the cost of higher bias, moving the fitted model leftward along the trade-off of Figure 3.3.1.

## 3.6 Ensemble Methods

Ensemble methods represent a paradigm shift from finding a single "best" model to combining multiple "weak learners" to create a "strong learner." These methods often produce state-of-the-art results in predictive tasks.

### 3.6.1 Bagging (Bootstrap Aggregating)

Bagging involves generating $B$ different bootstrapped training sets (sampling with replacement). A deep decision tree is trained on each set, and the predictions are aggregated (averaged for regression, majority vote for classification). By averaging multiple uncorrelated trees, bagging significantly reduces **variance**.

*Remark* (Why averaging reduces variance—and why correlation limits it). The resampling device here—drawing each of the $B$ training sets *with replacement* from the original sample—is exactly Efron's bootstrap, used by Breiman (2001), and the variance reduction has a one-line justification. Suppose each tree's prediction at a point has variance $\sigma^2$ and that the predictions of two different trees have pairwise correlation $\rho$. The average of $B$ such predictions then has variance

$$\operatorname{Var}\left(\frac{1}{B}\sum_{b=1}^{B}\hat{f}^{\,b}(x)\right) = \rho\,\sigma^2 + \frac{1-\rho}{B}\,\sigma^2.$$

The second term vanishes as $B \to \infty$, so adding trees only helps and carries essentially no overfitting risk; but the first term, $\rho\sigma^2$, does *not* depend on $B$. No matter how many trees we average, the variance floor is set by how *correlated* the trees are. Because each tree is grown deep (low bias) and the averaging does not raise bias, bagging trades a high-variance individual tree for a low-variance ensemble—but only down to that $\rho\sigma^2$ floor, which bagging itself leaves untouched.

### 3.6.2 Random Forests

Random Forests improve upon bagging by further decorrelating the trees (Breiman, 2001). When building each tree, at each split the algorithm is forced to consider only a **random subset of $m$ features** (typically $m \approx \sqrt{p}$, where $p$ is the total number of predictors). This prevents strong predictors from dominating every tree, ensuring diversity among the ensemble members and leading to better generalization.

*Remark* (Decorrelation: lowering the variance floor). Random forests attack precisely the $\rho\sigma^2$ floor identified in the previous remark. By restricting each split to a fresh random subset of $m$ of the $p$ features, the algorithm prevents the few strongest predictors from being chosen at the top of every tree; different trees are forced to rely on different variables, so their errors become less correlated and $\rho$ falls. The averaged forest therefore has lower variance than bagging, at the modest price of a slight increase in each individual tree's variance (each split now sees fewer candidate features). The subset size $m$ is itself a tuning parameter, trading off this decorrelation against per-tree strength.[13]

### 3.6.3 Boosting

Unlike bagging and Random Forests, which build trees in parallel, Boosting grows trees **sequentially**. Each new tree is trained to correct the errors (residuals) of the

[13]The default $m \approx \sqrt{p}$ quoted here is the usual choice for *classification*; for *regression* forests the common default is instead $m \approx p/3$ (Hastie et al., 2009). Both are tuned by cross-validation in practice.

previous tree.

1. Fit a tree to the data.
2. Calculate the residuals.
3. Fit a new tree to the residuals.
4. Update the model by adding a scaled version of the new tree.

The update step is

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \tag{3.6.1}$$

In (3.6.1):

- $\hat{f}(x)$: the current ensemble model before adding the new tree.
- $\hat{f}^b(x)$: the new decision tree fitted to the residuals (errors) of the current model.
- $\lambda$ **(Lambda)**: the **learning rate** (typically a small number such as 0.01 or 0.1). It shrinks the contribution of the new tree, preventing the model from learning too quickly and overfitting. This "slow learning" approach is key to boosting's success.

*Remark* (Boosting: bias reduction by forward stagewise fitting). It is worth contrasting boosting with bagging at the level of the bias-variance trade-off. Bagging and random forests average many low-bias, high-variance trees to drive down *variance*; boosting instead adds many high-bias, low-variance trees (each typically very shallow—a "stump" or a tree of depth two or three) to drive down *bias*. Fitting each new tree to the current residuals is an instance of *forward stagewise additive modeling*, and for a general differentiable loss $L$ the squared-error residual $y_i - \hat{f}(x_i)$ is replaced by the negative gradient $-\partial L/\partial \hat{f}(x_i)$ evaluated at the current fit. Each tree is thus one step of *gradient descent in function space*—hence the name *gradient boosting*—and the learning rate $\lambda$ in (3.6.1) is the step size: many small, well-regularized steps (small $\lambda$) typically generalize better than a few large ones. Because the steps are sequential and bias-reducing, boosting (unlike bagging) *can* overfit if too many trees are added, so the number of trees is itself tuned by cross-validation.

#### 3.6.3.1 *Python Example: Random Forest vs. Boosting*

Listing 2 contrasts a parallel ensemble (Random Forest) with a sequential one (Gradient Boosting) on a synthetic classification task.

```
from sklearn.ensemble import RandomForestClassifier,
    GradientBoostingClassifier
```

```python
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split

# Generate synthetic classification data
X, y = make_classification(n_samples=1000, n_features=20, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)

# Random Forest
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
print(f"Random Forest Accuracy: {rf.score(X_test, y_test):.4f}")

# Gradient Boosting
gb = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1,
    random_state=42)
gb.fit(X_train, y_train)
print(f"Gradient Boosting Accuracy: {gb.score(X_test, y_test):.4f}")
```

**Listing 2.** Random Forest versus Gradient Boosting on a synthetic classification dataset.

## 3.7 Advanced Topic: Causal Forests

While standard Random Forests predict an outcome $Y$ given covariates $X$, **Causal Forests** are designed to estimate the **Heterogeneous Treatment Effect (HTE)** (Wager and Athey, 2018; Athey and Imbens, 2016). The goal is to estimate

$$\tau(x) = E\left[Y^{(1)} - Y^{(0)} \mid X = x\right], \tag{3.7.1}$$

which represents the causal effect of a treatment (e.g., a marketing campaign) on an individual with characteristics $x$.

To understand this, we rely on the **Potential Outcomes Framework** (also known as the Rubin Causal Model). For any individual, there are two potential outcomes:

- $Y^{(1)}$ **(Treatment Outcome)**: the outcome that *would* be observed if the individual received the treatment (e.g., the customer buys the product after seeing an ad).
- $Y^{(0)}$ **(Control Outcome)**: the outcome that *would* be observed if the individual did *not* receive the treatment (e.g., the customer's behavior without seeing the ad).

*Remark* (Fundamental Problem of Causal Inference). The fundamental challenge of causal inference is that we can never observe both $Y^{(1)}$ and $Y^{(0)}$ for the same individual simultaneously. Causal Forests address this by estimating the *expected* difference between these potential outcomes for individuals with similar features $X$.

**What makes $\tau(x)$ identified.** The estimand $\tau(x)$ in (3.7.1)—the *conditional average treatment effect* (CATE), which when averaged over the distribution of $x$ returns the average treatment effect (ATE)—is defined through potential outcomes we never both observe, so it is not automatically recoverable from data on the treatment indicator $W \in \{0,1\}$, the covariates $X$, and the realized outcome $Y = W\,Y^{(1)} + (1-W)\,Y^{(0)}$. Identification requires the standard assumptions of the selection-on-observables paradigm the reader knows from program evaluation.

**Assumption 3.7.1** (Unconfoundedness and overlap)**.** *Unconfoundedness* (conditional ignorability): given the covariates, treatment assignment is as good as random, $\{Y^{(0)}, Y^{(1)}\} \perp W \mid X$. *Overlap* (positivity): every type of unit has a positive chance of either condition, $0 < P(W = 1 \mid X = x) < 1$ for all $x$ in the support.

Under Assumption 3.7.1, conditional means of $Y$ within treatment arms recover the conditional means of the potential outcomes, so $\tau(x) = E[Y \mid X = x, W = 1] - E[Y \mid X = x, W = 0]$ becomes estimable; in a randomized experiment—a marketing A/B test, say—unconfoundedness holds by design. A causal forest can be read as a nonparametric, adaptive way of forming the matched "similar units" comparison that these assumptions license, letting the data decide which dimensions of $x$ the treatment effect actually varies along.

To ensure valid inference, Causal Forests employ *honesty*: one half of the data is used to build the tree structure (deciding splits), and the other half is used to estimate the treatment effect within the leaves. This separation prevents overfitting and allows for accurate estimation of personalized causal effects.

*Remark* (Honesty as sample-splitting, and valid confidence intervals). The "honest" device of using one subsample to choose the splits and a disjoint subsample to estimate the leaf-level effects is the same sample-splitting logic that underlies cross-fitting in double/debiased machine learning. Its purpose is inferential. If the same data were used both to decide *where* treatment effects look large and to estimate *how* large they are, the estimates would be optimistically biased—the splits would chase noise—and ordinary standard errors would be invalid. By divorcing structure-learning from effect-estimation, Wager and Athey (2018) show that the causal-forest estimate $\hat{\tau}(x)$ is pointwise asymptotically normal, which yields valid confidence intervals for the personalized effect $\tau(x)$—a property ordinary predictive forests do not enjoy, and a

prerequisite for using these methods to make causal rather than merely predictive claims.

## References

The material in this chapter draws on the standard statistical-learning references (James et al., 2013; Hastie et al., 2009), the original Random Forests paper (Breiman, 2001), and foundational work on causal forests and recursive partitioning for heterogeneous causal effects (Wager and Athey, 2018; Athey and Imbens, 2016).

# Chapter 4: Introduction to Deep Learning

This chapter provides a comprehensive overview of the foundational concepts of deep learning, tailored for doctoral students in business. The objective is to offer a clear and intuitive understanding of the mathematical underpinnings of deep learning models and their practical applications in research. We begin by revisiting the principles of supervised learning and model training, with a focus on gradient-based optimization. We then study the architecture of deep neural networks, their expressive power and the mechanisms behind their learning capabilities, before turning to the computational realities of training large-scale models, including the hardware, resources, and efficiency considerations that govern modern practice.

## Abstract

This chapter provides a comprehensive overview of the foundational concepts of deep learning, tailored for doctoral students in business. The content is based on the lecture materials from the course "DOTE 6635: Artificial Intelligence for Business Research" and is supplemented with additional explanations, code examples, and references to seminal literature. The objective is to offer a clear and intuitive understanding of the mathematical underpinnings of deep learning models and their practical applications in research. We begin by revisiting the principles of supervised learning and model training, with a focus on gradient-based optimization. Subsequently, we delve into the architecture of deep neural networks, exploring their expressive power and the mechanisms behind their learning capabilities. Finally, we discuss the computational aspects of deep learning, including the hardware and resources required to train large-scale models.

## 4.1 The Framework of Supervised Learning and Optimization

Supervised learning constitutes a dominant paradigm in machine learning, wherein the primary objective is to learn a functional mapping from an input space to an output space. This is achieved by leveraging a dataset of labeled input-output pairs. Within the domain of deep learning, this mapping function is represented by a deep neural network. The process of discerning the optimal parameters for this function is termed *model training*, an optimization problem aimed at minimizing a predefined loss function that quantifies the disparity between the model's predictions and the observed ground truth.

### 4.1.1 Gradient Descent: The Engine of Optimization

At the heart of training for most contemporary machine learning models lies the **gradient descent** algorithm. As a first-order iterative optimization method, its purpose is to identify a local minimum of a differentiable function. The core principle is intuitive: to minimize a function, one should take steps in the direction of the function's steepest descent. This direction is precisely the negative of the function's gradient. The parameter update rule for gradient descent is thus formulated as:

$$\theta_{t+1} = \theta_t - \alpha \nabla L(\theta_t), \tag{4.1.1}$$

where $\theta_t$ represents the vector of model parameters at iteration $t$, $\alpha$ is the learning rate, a critical hyperparameter that governs the magnitude of each step, and $\nabla L(\theta_t)$ is the gradient of the loss function $L$ with respect to the parameters $\theta$ at the current iteration.

*Remark* (Connection to iterative estimation in econometrics). Gradient descent should feel familiar: it is the same kind of *iterative* numerical optimizer that econometric software invokes whenever a likelihood or moment objective has no closed form. Maximum-likelihood and GMM estimators are routinely computed by Newton–Raphson, Fisher scoring, or BHHH updates of the form $\theta_{t+1} = \theta_t - H_t^{-1} \nabla L(\theta_t)$, where $H_t$ is the Hessian (or an approximation to it). Gradient descent is the *first-order* special case in which the matrix $H_t^{-1}$ is replaced by a single scalar step size $\alpha$. It thus throws away curvature information, which makes each iteration far cheaper—there is no Hessian to form or invert, decisive when $\theta$ has millions of coordinates—at the price of needing many more iterations. This trade-off is precisely why first-order methods, rather than the Newton-type methods favored in classical econometrics, dominate deep learning, where the parameter vector can be astronomically large.

The convergence properties of gradient descent are contingent upon the charac-

teristics of the loss function. For convex functions possessing a Lipschitz continuous gradient, the algorithm is guaranteed to converge to a global minimum. However, the rate of this convergence varies.

**Theorem 4.1.1** (Convergence of Gradient Descent)**.** *Let $f$ be a convex and differentiable function with a Lipschitz continuous gradient, meaning there exists a constant $L > 0$ such that $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$ for all $x, y$. If the learning rate $\alpha$ is chosen such that $0 < \alpha \leq 1/L$, then gradient descent converges. Moreover:*

1. *For a general convex function, the convergence rate is sublinear, requiring $O(1/\epsilon)$ iterations to achieve an accuracy of $\epsilon$.*
2. *If $f$ is also strongly convex, the convergence rate becomes linear (or geometric), requiring only $O(\log(1/\epsilon))$ iterations, which is substantially faster (Tibshirani, 2016).*

**Reading the theorem.** A few words unpack what this guarantees, and—just as importantly—what it does not. The *Lipschitz-gradient* condition $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$ bounds how fast the gradient can change; $L$ is an upper bound on the curvature (the largest eigenvalue of the Hessian, when $f$ is twice differentiable). It is exactly this bound that licenses the step-size rule $\alpha \leq 1/L$: if the surface can bend sharply ($L$ large) one must take small steps, lest the update overshoot the minimum and diverge. "Sublinear" convergence ($O(1/\epsilon)$ iterations) means the error shrinks like $1/t$, so each additional digit of accuracy costs geometrically more work; "linear" or geometric convergence ($O(\log(1/\epsilon))$ iterations) means the error is multiplied by a fixed factor below 1 at every step, so accuracy improves *exponentially* fast. The catch for deep learning is that the loss surfaces of neural networks are emphatically *non-convex*, so none of these global guarantees strictly apply: in practice gradient descent settles into a *local* minimum or a low-loss region, and—remarkably—such solutions usually generalize well, for reasons still actively studied. This gap between the clean convex theory and the messy non-convex practice motivates the more robust optimizers of the next subsection.

To provide a tangible example, consider the simple quadratic function $f(x) = x^2 - 4x + 4$. The gradient is $f'(x) = 2x - 4$. A basic implementation of gradient descent in Python would be:

```
# A simple demonstration of gradient descent

# Define the function and its gradient
def f(x):
    return x**2 - 4*x + 4

def grad_f(x):
```

```
    return 2*x - 4

# Hyperparameters
learning_rate = 0.1
iterations = 50
current_x = 0.0 # Initial guess

print(f"Starting gradient descent at x = {current_x}")

for i in range(iterations):
    gradient = grad_f(current_x)
    current_x = current_x - learning_rate * gradient
    if (i+1) % 10 == 0:
        print(f"Iteration {i+1}: x = {current_x:.4f}, f(x) = {f(current_x):.4f}
            ")

print(f"Minimum found at x = {current_x:.4f}")
```

**Listing 3.** A simple demonstration of gradient descent.

### 4.1.2 Foundational Models as Illustrations

The principles of gradient-based optimization can be clearly observed in classical statistical models. In **Ordinary Least Squares (OLS)** regression, the objective is to minimize the sum of squared residuals, $L(\beta) = \sum_{i=1}^{n}(y_i - x_i^T\beta)^2$. While OLS benefits from a closed-form analytical solution ($\hat{\beta} = (X^TX)^{-1}X^Ty$), it serves as a useful pedagogical tool for demonstrating gradient descent. In contrast, **logistic regression**, used for binary classification, lacks a closed-form solution. Its parameters are estimated by minimizing the cross-entropy loss, a task for which iterative methods like gradient descent are indispensable.

**Cross-entropy is negative log-likelihood.** The phrase "cross-entropy loss" can obscure a fact every econometrician already knows: for logistic regression it is *exactly* the negative log-likelihood of the Bernoulli model, so minimizing it is maximum-likelihood estimation under another name. Writing $p_i = 1/(1 + e^{-x_i^T\beta})$ for the predicted probability that $y_i = 1$, the loss is

$$L(\beta) = -\sum_{i=1}^{n}\Big[\, y_i \log p_i + (1 - y_i)\log(1 - p_i)\,\Big],$$

which is $-1$ times the familiar logit log-likelihood. More generally, when a network outputs a probability distribution over $K$ classes through a softmax, the cross-entropy $-\sum_i \log p_{i,y_i}$ is the negative multinomial log-likelihood—the very object underlying the multinomial-logit discrete-choice models used throughout marketing and economics. Reading deep-learning losses through this likelihood lens demystifies them: squared-error loss corresponds to a Gaussian likelihood (OLS), and cross-entropy to a Bernoulli or multinomial likelihood (logit). For OLS itself the gradient is $\nabla L(\beta) = -2X^T(y - X\beta)$, so a gradient-descent step nudges $\beta$ in the direction that most rapidly reduces the residual sum of squares; setting this gradient to zero recovers the normal equations, and hence the closed form $\hat{\beta} = (X^T X)^{-1} X^T y$.

### 4.1.3 Enhancements to Gradient-Based Optimization

The vanilla gradient descent algorithm, while foundational, is often too slow or unreliable for the high-dimensional, non-convex landscapes typical of deep learning. Consequently, several more sophisticated variants have been developed.

**Stochastic Gradient Descent (SGD)** addresses the computational bottleneck of large datasets by approximating the true gradient using only a small, random subset of the data (a "mini-batch") at each iteration. This introduces noise into the optimization process, which can help the algorithm escape shallow local minima, but it also results in a more erratic convergence path.

*Remark* (SGD as stochastic approximation). The mini-batch gradient is an *unbiased estimator* of the full-sample gradient: when a mini-batch is drawn uniformly at random, the expectation of its gradient equals the true gradient $\nabla L(\theta)$, just as a sample average is an unbiased estimator of a population mean. SGD therefore replaces the exact descent direction by a noisy but unbiased one—precisely the logic of the Robbins–Monro stochastic-approximation scheme that also underlies recursive estimators and online-learning procedures.[14] Classical stochastic-approximation theory shows that, with step sizes satisfying $\sum_t \alpha_t = \infty$ and $\sum_t \alpha_t^2 < \infty$ (decaying neither too quickly nor too slowly), the iterates still converge despite the noise. Beyond merely tolerating this noise, deep-learning practice often *welcomes* it: the fluctuations act like a mild form of annealing that helps the iterates slip out of sharp, poorly generalizing minima and away from saddle points.

To smooth out this path and accelerate convergence, the **Momentum** method was introduced. It accumulates an exponentially decaying moving average of past gradients, analogous to a ball rolling down a hill that gathers momentum. This allows the

[14] H. Robbins and S. Monro, "A stochastic approximation method," *Annals of Mathematical Statistics*, 22(3):400–407, 1951.

optimizer to navigate ravines more effectively and dampen oscillations in directions of high curvature.

Perhaps the most ubiquitous optimizer in modern practice is **Adam (Adaptive Moment Estimation)** (Kingma and Ba, 2014). Adam synergistically combines the concept of momentum with adaptive learning rates. It maintains separate decaying averages of both the past gradients (the first moment, like momentum) and the past squared gradients (the second moment, which captures the variance). By using these estimates, Adam computes individualized, adaptive learning rates for each parameter, making it robust and often effective with minimal hyperparameter tuning.

*Remark* (The Adam update, explicitly). For the mathematically inclined reader it helps to see Adam in symbols. Let $g_t = \nabla L(\theta_t)$ denote the (mini-batch) gradient. Adam maintains exponentially weighted averages of the gradient and of its element-wise square,

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\, g_t, \qquad v_t = \beta_2 v_{t-1} + (1-\beta_2)\, g_t \odot g_t,$$

with decay rates typically $\beta_1 \approx 0.9$ and $\beta_2 \approx 0.999$. Because initializing $m_0 = v_0 = 0$ biases these averages toward zero in early iterations, Adam applies the bias corrections $\hat{m}_t = m_t/(1-\beta_1^t)$ and $\hat{v}_t = v_t/(1-\beta_2^t)$, and then updates each coordinate by

$$\theta_{t+1} = \theta_t - \alpha \, \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon},$$

where the division is element-wise. A coordinate whose gradients have been large and volatile (large $\hat{v}_t$) receives a *smaller* effective step, while one with small, consistent gradients receives a larger one. Here $\hat{m}_t$ supplies the momentum—a smoothed search direction—and $\sqrt{\hat{v}_t}$ supplies a per-coordinate, data-adaptive rescaling reminiscent of standardizing a regressor by its standard deviation; the constant $\varepsilon \approx 10^{-8}$ merely guards against division by zero.

## 4.2 The Architecture and Theory of Deep Neural Networks

Having established the optimization framework, we now turn to the models themselves: **Deep Neural Networks (DNNs)**. This section explores their architecture, the theoretical guarantees of their expressive power, and the practical considerations for their construction and training.

**Definition 4.2.1** (Feedforward neural network / multilayer perceptron)**.** A feedforward neural network with $L$ layers is a composition of alternating affine maps and

element-wise nonlinearities. Writing $h^{(0)} = x$ for the input, each layer computes

$$z^{(\ell)} = W^{(\ell)} h^{(\ell-1)} + b^{(\ell)}, \qquad h^{(\ell)} = \sigma\big(z^{(\ell)}\big), \qquad \ell = 1, \dots, L,$$

where $W^{(\ell)}$ and $b^{(\ell)}$ are the *weight matrix* and *bias vector* of layer $\ell$ (collectively the parameters $\theta$), and $\sigma(\cdot)$ is a fixed nonlinear *activation function* applied coordinate-wise. The output is $h^{(L)}$, with the final activation chosen to match the task (the identity for regression, a softmax for classification). A single "unit" or "neuron" therefore computes a familiar object: a linear index $w^T h + b$—the analogue of a regression linear predictor—passed through a nonlinearity.

*Remark* (Why the nonlinearity is essential). Without the activation $\sigma$, stacking layers would be pointless: a composition of affine maps is itself affine, so any "deep" purely linear network collapses to a single linear regression $x \mapsto Wx + b$. It is the nonlinearity—historically the sigmoid, today almost always the *rectified linear unit* $\mathrm{ReLU}(z) = \max(0, z)$—that lets each additional layer build genuinely richer, hierarchical features. ReLU has become the default because it is trivial to evaluate and its derivative is either 0 or 1, which sidesteps the "vanishing gradient" problem that saturating sigmoids cause in deep stacks (where repeated multiplication of small derivatives during backpropagation drives the gradient signal toward zero).

### 4.2.1 The Connectionist Paradigm

The genesis of neural networks lies in the **connectionist** movement, which sought to model intelligence by drawing inspiration from the parallel, distributed processing architecture of the human brain. This stands in contrast to the classical **symbolic AI** approach, which is predicated on rule-based manipulation of symbols. The 2024 Nobel Prize in Physics, awarded to John J. Hopfield and Geoffrey E. Hinton, celebrated their pioneering work on **Hopfield networks** and **Boltzmann machines**, which connected concepts from statistical mechanics to the learning dynamics of neural networks, laying a crucial theoretical foundation for the field.[15]

### 4.2.2 The Power of Representation: Universal Approximation

A cornerstone of neural network theory is the **Universal Approximation Theorem**. This result provides a powerful guarantee about the expressive capacity of even simple network architectures.

[15] https://www.nobelprize.org/prizes/physics/2024/press-release/; see also Wang et al. (2024), *The Innovation*, 5(6).

**Theorem 4.2.2** (Universal Approximation Theorem; Cybenko, 1989; Hornik et al., 1989)**.** *Let $\sigma(\cdot)$ be a non-constant, bounded, and monotonically-increasing continuous function (a "squashing" function, like the sigmoid). Let $I_m$ denote the m-dimensional unit hypercube $[0,1]^m$, and let $C(I_m)$ denote the space of continuous functions on $I_m$. Given any function $f \in C(I_m)$ and any $\epsilon > 0$, there exist an integer $N$, real constants $v_i, b_i \in \mathbb{R}$, and vectors $w_i \in \mathbb{R}^m$ for $i = 1, \ldots, N$, such that the function*

$$F(x) = \sum_{i=1}^{N} v_i \, \sigma(w_i^T x + b_i) \tag{4.2.1}$$

*provides an approximation of $f$ with arbitrary accuracy; that is, $|F(x) - f(x)| < \epsilon$ for all $x \in I_m$.*

In essence, the theorem states that a single hidden layer in a neural network is sufficient to approximate any continuous function to any desired degree of precision. However, it is an existence theorem: it does not prescribe how to find the network's parameters (weights and biases), nor does it suggest that a single-layer network is the most efficient or learnable representation for a given problem. The "deep" aspect of deep learning, the use of multiple hidden layers, is motivated by the empirical finding that hierarchical representations can learn complex features more efficiently and with fewer parameters than their shallow counterparts.

*Remark* (Universal approximation as a sieve/series argument). A reader trained in nonparametric econometrics has already met the idea behind this theorem. Approximating an unknown function by a finite linear combination of flexible basis functions—and letting the number of bases grow with the sample—is the method of *sieves*, or series estimation, and classical results establish that polynomials, splines, and Fourier bases are each dense in $C(I_m)$ (a Stone–Weierstrass-type property). Theorem 4.2.2 simply adds one more such basis: the "ridge" functions $x \mapsto \sigma(w_i^T x + b_i)$. The hidden layer thus plays the role of an adaptive basis expansion, with the crucial twist that the basis functions themselves—through the learned $w_i, b_i$—are estimated from data rather than fixed in advance. Two caveats temper the result. First, it is purely an existence statement, akin to knowing that a consistent series estimator exists without being told its convergence rate or how to compute it. Second, the required width $N$ can grow explosively with the input dimension $m$ (a curse of dimensionality); the empirical case for *depth* is precisely that composing many narrow layers can represent many functions of interest with far fewer parameters than a single impossibly wide layer.

### 4.2.3 Are Simple Multilayer Perceptrons Outdated? Applications in Business Research

Given the rapid advancements in deep learning architectures, from convolutional neural networks (CNNs) to transformers and large language models (LLMs), one might question whether the simple **Multilayer Perceptron (MLP)**, the most basic form of a feedforward neural network, has become obsolete. The answer, perhaps surprisingly, is a resounding "no." The key to leveraging MLPs effectively lies not in their architectural complexity, but in identifying **interesting and impactful applications** where their simplicity is a virtue.

For business researchers, the MLP remains a powerful and highly relevant tool. Its relative simplicity offers advantages in terms of interpretability, computational efficiency, and theoretical tractability, qualities that are often paramount in academic research where understanding *why* a model works is as important as its predictive accuracy. Several recent publications in top-tier management and operations research journals demonstrate the continued vitality of MLP-based approaches.

**Example 4.2.3** (AI for Scientific Discovery)**.** A landmark paper published in *Nature* by Davies et al. (2021), titled "Advancing mathematics by guiding human intuition with AI," demonstrated how machine learning, including neural network models, can assist mathematicians in discovering new patterns and formulating conjectures. This work, a collaboration between DeepMind and leading mathematicians, showcases how AI can augment human intuition in pure mathematics, a domain far removed from the typical "big data" applications of deep learning. The success of this project underscores that the value of neural networks often lies in their application to novel, high-impact problems rather than in the sheer scale of the model.

**Example 4.2.4** (Causal Inference in Large-Scale Experiments)**.** In the domain of operations and marketing, a paper by Ye et al. (2024), published in *Management Science* and titled "Deep Learning-Based Causal Inference for Large-Scale Combinatorial Experiments," addresses a critical challenge faced by online platforms. These platforms run thousands of A/B tests daily, but the sheer number of treatment combinations makes it infeasible to test every possibility. The authors develop a novel framework called "Debiased Deep Learning" (DeDL) that combines deep learning with doubly robust estimation to infer the causal effect of any treatment combination, even those that were never directly observed. This work highlights how neural networks can be integrated with established econometric techniques to solve practical business problems at scale.

**Example 4.2.5** (Deep Learning in Asset Pricing)**.** In finance, Chen et al. (2024) published "Deep Learning in Asset Pricing" in *Management Science*. They use deep neural networks to estimate an asset pricing model for individual stock returns. The model

takes advantage of a vast amount of conditioning information, maintains a flexible functional form, and accounts for time variation. Their approach outperforms traditional benchmark models in terms of Sharpe ratio, explained variation, and pricing errors, demonstrating the power of neural networks to capture complex, non-linear relationships in financial data.

**Example 4.2.6** (Neural Networks for Choice Modeling)**.** Another compelling example comes from the field of operations research. Wang et al. (2023b) have developed a "Neural-Network Mixed Logit Choice Model" that provides statistical and optimality guarantees. The mixed logit model is a workhorse in marketing and econometrics for modeling consumer choice. The authors show that a single-hidden-layer neural network can effectively approximate the mixture distribution in the mixed logit model. Crucially, they provide theoretical guarantees that the approximation error does not suffer from the curse of dimensionality, and that stochastic gradient descent can find the global optimum of the regularized problem. This work is a prime example of how rigorous theoretical analysis can validate the use of simple neural network architectures in established econometric frameworks.

These examples collectively illustrate a vital point: the frontier of impactful research is not solely defined by the complexity of the model, but by the novelty and significance of the problem being addressed. For business school researchers, this means that mastering the fundamentals of MLPs and understanding their theoretical properties can open doors to a wide range of high-impact research opportunities.

### 4.2.4 The Mechanics of Learning: Backpropagation and Regularization

The training of a DNN is accomplished via the **backpropagation** algorithm, which is a highly efficient method for computing the gradients of the loss function with respect to all network parameters. It works by first performing a **forward pass**, where the input signal propagates through the network to produce a prediction, followed by a **backward pass**, where the error signal is propagated backward from the output layer. At each layer, the chain rule of calculus is applied to compute the local gradients, which are then used by an optimizer like Adam to update the weights.

**Backpropagation is the chain rule, organized.** There is nothing mysterious in backpropagation beyond the multivariate chain rule the reader already knows; its content is purely *computational efficiency*. A deep network is a long composition $L = \ell\big(f_L(\cdots f_2(f_1(x))\cdots)\big)$, and the derivative of the loss with respect to an early-layer weight is, by the chain rule, a product of the per-layer Jacobians lying between that

weight and the output. The naive way to obtain every parameter's derivative would be to recompute such a product separately for each of the millions of parameters or—worse—to perturb each parameter and re-run the whole network (finite differences), costing one forward pass *per parameter*. Backpropagation instead sweeps *backward* exactly once, from output to input, reusing the shared sub-products: it caches the activations from the forward pass and propagates a single "error signal" $\partial L/\partial z^{(\ell)}$ through the layers. The payoff is that the gradient with respect to *all* parameters is obtained in one backward pass whose cost is a small constant multiple of one forward pass—independent of the number of parameters. This is the *reverse mode* of automatic differentiation, and it is the reason that training networks with billions of parameters is feasible at all.

Given their immense number of parameters, DNNs are highly susceptible to **overfitting**, a phenomenon where the model memorizes the training data, including its idiosyncrasies and noise, at the expense of its ability to generalize to unseen data. This is a manifestation of the classic **bias-variance tradeoff** (Belkin et al., 2019).[16] To mitigate this, **regularization** techniques are essential.

Standard methods include **L1 (Lasso)** and **L2 (Ridge)** regularization, which add a penalty to the loss function based on the magnitude of the network's weights (Tibshirani, 1996). Concretely, one augments the loss with a penalty $\lambda \sum_j |w_j|$ (L1) or $\lambda \sum_j w_j^2$ (L2)—exactly the Lasso and Ridge penalties familiar from econometrics, with $\lambda > 0$ controlling the strength of shrinkage. The intuition transfers verbatim: L2 shrinks all weights smoothly toward zero (and, under the name "weight decay," is the most common regularizer in deep learning), while L1 induces sparsity by driving some weights exactly to zero. Penalizing weight magnitude keeps the fitted function smooth and limits the model's effective flexibility, so it cannot chase noise in the training data. A more modern and highly effective technique specific to neural networks is **Dropout** (Srivastava et al., 2014). During training, Dropout randomly sets the activations of a fraction of neurons to zero at each forward pass. This prevents complex co-adaptations between neurons and forces the network to learn more robust and redundant representations, akin to training an ensemble of many smaller networks.

Implementing dropout in a modern framework like Keras is straightforward:

```
# Example of a simple Keras model with Dropout and L2 regularization

# This is a conceptual example and requires a full environment to run.
```

[16] The classic tradeoff predicts a U-shaped test-error curve: error first falls as a model grows more flexible (lower bias) and then rises (higher variance). Modern deep networks complicate this picture. Belkin et al. (2019) document *double descent*, in which pushing capacity past the "interpolation threshold"—enough to fit the training data exactly—causes test error to fall *again*. Heavily over-parameterized networks thus often generalize well despite memorizing the training set, which is why the cited work *reconciles* the classical tradeoff with modern practice rather than simply illustrating it.

```
# from tensorflow.keras.models import Sequential
# from tensorflow.keras.layers import Dense, Dropout
# from tensorflow.keras.regularizers import l2

# model = Sequential([
# Dense(128, activation='relu', input_shape=(784,), kernel_regularizer=l2
    (0.001)),
# Dropout(0.5), # Apply dropout with a 50% rate
# Dense(64, activation='relu', kernel_regularizer=l2(0.001)),
# Dropout(0.5),
# Dense(10, activation='softmax')
# ])

# model.summary()
```

**Listing 4.** A simple Keras model with Dropout and L2 regularization.

## 4.3 The Computational Landscape of Deep Learning

The practical realization of deep learning is as much a story of computational engineering as it is of theoretical advances. The training of state-of-the-art models is a computationally demanding process that relies on a specialized ecosystem of software and hardware. This section provides a comprehensive overview of the tools, infrastructure, and economic considerations that underpin modern deep learning research.

### 4.3.1 Deep Learning Software Libraries

The good news for researchers entering the field is that you will likely never need to implement backpropagation from scratch. A rich ecosystem of software libraries has emerged to handle the mathematical and computational complexities of deep learning, allowing researchers to focus on model design and application.

At the foundation are **general parallel computing libraries**, primarily **TensorFlow** (developed by Google) and **PyTorch** (developed by Meta/Facebook). These libraries are conceptually similar to NumPy, the ubiquitous Python library for numerical computing, but with two critical advantages: they provide much better support for parallel computing on GPUs, and they automatically compute and store derivatives, a feature known as **automatic differentiation**. This latter capability is the core function of any deep learning library: to take and store the derivatives of complex computational graphs in an automated and efficient fashion, which is essential for the backpropagation algorithm.

Built on top of these foundational libraries are higher-level **deep learning frameworks** such as **Keras** (which provides a user-friendly API for TensorFlow) and **Hugging Face** (which offers a vast repository of pre-trained models and tools for natural language processing). These frameworks allow researchers to define, train, and deploy complex models with just a few lines of code.

Table 4.3.1 illustrates the correspondence between common NumPy operations and their PyTorch equivalents, highlighting the ease of transition for researchers already familiar with numerical computing in Python.

**Table 4.3.1.** Correspondence between common NumPy operations and their PyTorch equivalents.

| Operation | NumPy | PyTorch |
|---|---|---|
| Array/Tensor Creation | `numpy.array()` | `torch.tensor()` |
| Dimensions | `array.ndim` | `tensor.dim()` |
| Shape | `array.shape` | `tensor.size()` |
| Sum over all elements | `numpy.sum(array)` | `torch.sum(tensor)` |
| Mean | `numpy.mean(array)` | `torch.mean(tensor)` |
| Standard Deviation | `numpy.std(array)` | `torch.std(tensor)` |
| Element-wise Sum | `array1 + array2` | `tensor1 + tensor2` |
| Element-wise Product | `array1 * array2` | `tensor1 * tensor2` |
| Matrix Multiplication | `numpy.dot(a, b)` | `torch.matmul(a, b)` |
| Reshape | `array.reshape()` | `tensor.view()` |
| Transpose | `array.T` | `tensor.t()` |

A simple example of defining and training a neural network in PyTorch demonstrates the elegance of these modern tools:

```
import torch
import torch.nn as nn
import torch.optim as optim

# Define a simple 2-layer neural network
class SimpleNet(nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim):
        super(SimpleNet, self).__init__()
        self.fc1 = nn.Linear(input_dim, hidden_dim)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(hidden_dim, output_dim)

    def forward(self, x):
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
```

```
        return x

# Instantiate model, loss function, and optimizer
model = SimpleNet(input_dim=10, hidden_dim=64, output_dim=1)
criterion = nn.MSELoss() # Mean Squared Error for regression
optimizer = optim.Adam(model.parameters(), lr=0.001)

# Training loop (conceptual)
# for epoch in range(num_epochs):
# for inputs, targets in dataloader:
# optimizer.zero_grad() # Clear previous gradients
# outputs = model(inputs) # Forward pass
# loss = criterion(outputs, targets)
# loss.backward() # Backward pass (computes gradients)
# optimizer.step() # Update parameters
```

**Listing 5.** Defining and training a simple neural network in PyTorch.

### 4.3.2 Computing Hardware: GPUs and Beyond

The computational demands of deep learning necessitate specialized hardware. While **Central Processing Units (CPUs)** are general-purpose processors optimized for sequential tasks, **Graphics Processing Units (GPUs)** feature a massively parallel architecture with thousands of smaller cores designed for simultaneous operations. This makes GPUs exceptionally well-suited for the matrix multiplications that dominate neural network computations.

Researchers have two primary options for accessing GPU compute.

**Self-Made Workstations and Servers.** Building a custom workstation with a high-end consumer GPU, such as the NVIDIA RTX 4090 (approximately US$1,500, though often difficult to procure), is a cost-effective option for individual researchers. For larger-scale needs, institutions may invest in dedicated High-Performance Computing (HPC) clusters. For example, the CUHK Business School AI Lab has deployed two HPC clusters equipped with 16 NVIDIA 4090s, 4 NVIDIA 3090s, and 24 NVIDIA H100s.

**Cloud Computing.** Major cloud providers offer on-demand access to powerful GPU instances, providing scalability and flexibility without the need for upfront capital investment. Pricing for high-end instances is substantial, as summarized in Table 4.3.2.

The performance difference between GPU generations is significant. Benchmarks on the ResNet-50 model (a standard computer vision architecture) show that the

**Table 4.3.2.** Approximate on-demand pricing for high-end cloud GPU instances.

| Provider | Instance Type | Approximate Cost |
|---|---|---|
| Google Cloud Platform (GCP) | 8× NVIDIA H100 | ∼US$38.84 per hour |
| Amazon Web Services (AWS) | 8× NVIDIA H100 | ∼US$39.33 per hour |

NVIDIA H100 outperforms the A100, which in turn significantly outperforms the consumer-grade RTX 4090. Crucially, performance scales near-linearly when using multiple GPUs in parallel (e.g., 4 or 8 GPUs), enabling the training of very large models; see Table 4.3.3.

**Table 4.3.3.** GPU benchmark scores on ResNet-50 (FP16). Higher is better.

| Configuration | NVIDIA A100 80GB | NVIDIA H100 | NVIDIA RTX 4090 |
|---|---|---|---|
| 1 GPU | 2,535 points | 3,042 points | 1,720 points |
| 4 GPUs | 9,366 points | 11,989 points | 5,934 points |
| 8 GPUs | 21,479 points | 30,070 points | N/A |

### 4.3.3 Model Size, Training Time, and Computational Costs

The cost of training a deep learning model is directly proportional to two factors: the **size of the network** (i.e., the number of parameters) and the **size of the training data**. Understanding the relationship between model scale and training time is crucial for planning research projects.

The following examples provide concrete benchmarks for training well-known models on a DGX server (a high-end NVIDIA system with 8 A100 GPUs).

**ResNet-50 (approximately 25.6 million parameters).** This classic computer vision model, trained on the ImageNet dataset using TF32 precision, takes approximately **30 minutes** on a DGX server. On a single consumer-grade RTX 4090, the same task would take roughly 6-10 hours.

**BERT (110 million parameters).** This foundational Natural Language Processing (NLP) model, pre-trained on approximately 16GB of text from BooksCorpus and Wikipedia, takes approximately **5 hours** on a DGX server. Fine-tuning BERT on a smaller, task-specific dataset like the Stanford Question Answering Dataset (SQuAD) takes only 3-5 minutes.

**GPT-3 (175 billion parameters).** Training this landmark large language model required an estimated $3.15 \times 10^{23}$ FLOPs (floating-point operations),[17] trained on 300 billion tokens. To understand the scale, consider the calculation

$$\text{Total FLOPs} \approx \text{Parameters} \times \text{Tokens} \times 6, \tag{4.3.1}$$

where the factor of 6 represents the approximate compute per parameter per token in transformer architectures. Using 128 DGX servers (each with 8 A100 GPUs running at 80 TFLOP/s), the estimated training time is

$$\text{Training Time} = \frac{3.15 \times 10^{23}}{128 \times 8 \times 80 \times 10^{12} \times 24 \times 3600} \approx 51 \text{ to } 100 \text{ days}, \tag{4.3.2}$$

which translates to roughly **3 months** of continuous training.[18]

### 4.3.4 Scaling Laws and the Economics of Large Models

Recent research has focused on understanding the **scaling laws** of neural networks, which describe the empirical relationship between model performance, model size, dataset size, and the amount of compute used for training (Kaplan et al., 2020). These laws, pioneered by researchers at OpenAI, suggest that performance (measured by loss) improves predictably as a power law of these factors:

$$L(N, D, C) \propto N^{-\alpha_N} + D^{-\alpha_D} + C^{-\alpha_C}, \tag{4.3.3}$$

where $L$ is the loss, $N$ is the number of parameters, $D$ is the dataset size, $C$ is the compute budget, and the $\alpha$ values are empirically determined exponents. This finding has profound implications: it suggests that, within certain regimes, simply scaling up models, data, and compute will yield predictable performance gains.

*Remark* (Scaling laws as log-log power laws). The power-law form in (4.3.3) is best read on a log scale, where it becomes the kind of linear relationship an applied economist estimates every day. Holding the other terms fixed, $L \propto N^{-\alpha_N}$ implies $\log L \approx \text{const} - \alpha_N \log N$, so plotting log-loss against log-parameters yields a straight line whose slope

[17] A FLOP is a single floating-point arithmetic operation (one add or one multiply). The rule of thumb in (4.3.1)—total training compute $\approx 6ND$ for a model with $N$ parameters trained on $D$ tokens—comes from counting these operations in a transformer: a forward pass costs about 2 FLOPs per parameter per token (each weight enters one multiply and one add), and the backward pass that computes the gradients costs roughly twice as much again, for a total of $2 + 4 = 6$.

[18] The bare fraction in (4.3.2) evaluates to about 44 days—the wall-clock time one would obtain *if* the 1,024 GPUs sustained their full 80 TFLOP/s peak every second. They never do: communication, memory traffic, and pipeline stalls keep the realized *model FLOPs utilization* (MFU) well below 100% (often only around one-half), which is why practical estimates land in the higher 51–100 day range quoted here.

is the exponent $\alpha_N$—structurally identical to estimating a constant-elasticity (returns-to-scale) relationship by regressing logs on logs. The empirical content of Kaplan et al. (2020) is that these lines stay strikingly straight over many orders of magnitude, which is what lets one forecast the performance of a not-yet-trained model from small-scale pilot runs. The additive form $N^{-\alpha_N} + D^{-\alpha_D} + C^{-\alpha_C}$ is schematic shorthand: the three inputs are not independent (compute $C$ is itself roughly $6ND$), and a central practical question is how to split a fixed compute budget between a bigger model (larger $N$) and more data (larger $D$).

The economics of training frontier models are staggering. Table 4.3.4 compares the training costs of two recent large language models.

**Table 4.3.4.** Estimated training costs of two recent large language models.

| Model | Parameters | Training Cost (USD) |
|---|---|---|
| DeepSeek-V3 | 671 Billion (MoE) | ∼$5.576 Million |
| Llama 3.1 | 405 Billion | ∼$160-200 Million |

The dramatic cost difference highlights the importance of **architectural innovation**. DeepSeek-V3 employs a **Mixture-of-Experts (MoE)** architecture, which activates only a sparse subset of its 671 billion parameters (approximately 37 billion) for each input token (DeepSeek-AI, 2024). This allows it to achieve the performance of a much larger dense model while significantly reducing training and inference costs. DeepSeek-V3 was trained on 14.8 trillion tokens, requiring a total of approximately $3.3 \times 10^{24}$ FLOPs and 2,664,000 H800 GPU hours (approximately $5.328 million at $2/GPU-hour).

### 4.3.5 Geopolitical Considerations: GPU Export Controls

The strategic importance of advanced AI has led to significant geopolitical considerations surrounding the hardware that powers it. Beginning in October 2022, the United States government implemented export controls restricting the sale of high-performance GPUs (such as the NVIDIA A100 and H100) to certain countries, most notably China. These controls were further tightened in January 2025, expanding the range of restricted chips.

These restrictions have spurred the development of alternative chips (like the NVIDIA H800, a variant of the H100 designed to comply with export rules) and have accelerated domestic chip development efforts in affected countries. For business researchers, understanding this geopolitical landscape is important, as it affects the global distribution of AI research capabilities and the competitive dynamics of the technology industry.

## 4.4 Conclusion

This chapter has traversed the foundational landscape of deep learning, from the mathematical principles of gradient-based optimization to the architectural and theoretical underpinnings of deep neural networks, and finally to the computational realities of their implementation. For the business researcher, a solid grasp of these concepts is indispensable. Understanding the mechanics of gradient descent and its variants provides the intuition for how models learn. Knowledge of network architecture, the Universal Approximation Theorem, and regularization techniques informs how models are designed and controlled. Finally, an appreciation for the computational costs and scaling laws provides a pragmatic perspective on the feasibility and scope of applying these powerful tools to substantive research questions.

## 4.5 Compute-Efficient Training with GPUs

While Section 4.3 highlights the macro-level cost structure of deep learning, effective research practice also depends on **micro-level training efficiency**. In modern workflows, the difference between a feasible experiment and an infeasible one often comes down to GPU-aware implementation details. The following principles, distilled from recent lecture materials and practitioner guidance,[19] summarize how to train models efficiently without compromising rigor.

### 4.5.1 Hardware-Aware Implementation and Parallelism

Deep learning models should be implemented in a framework that is optimized for GPU execution (e.g., PyTorch). This is not merely a convenience; it is foundational to performance. The most important first step is to **keep computation on GPU** and avoid unnecessary data transfers between CPU and GPU. When multiple GPUs are available, the default strategy for scaling training is **data parallelism**, typically implemented via **Distributed Data Parallel (DDP)**. In DDP, each GPU processes a different mini-batch, gradients are synchronized across devices, and model parameters are updated collectively.

An additional practical rule is to **choose "nice" tensor dimensions**, especially batch sizes and sequence lengths. GPUs operate on **CUDA kernels** that are optimized for **power-of-two block sizes**. As a result, choosing batch sizes and sequence lengths that are multiples of $2^k$ (e.g., 64, 128, 256) often yields noticeably better throughput.

[19] See Karpathy (2024), *Compute Efficient Training with GPUs* (lecture); Stanford CS336 (2025), *Language Modeling from Scratch*; Dao et al. (2022); and MIT 6.5940 (2024), *Efficient Deep Learning Computing*.

### 4.5.2 Batch Size, Gradient Accumulation, and Learning-Rate Scaling

For GPU utilization, **larger batch sizes are typically more efficient**, but GPU memory imposes a hard ceiling. When memory constraints prevent the desired batch size, **gradient accumulation** is a principled workaround. The idea is to split a large batch into several smaller mini-batches, accumulate gradients across them, and update parameters only after the full effective batch has been processed.

For example, if the GPU can only fit a mini-batch of 8, but the target batch size is 32, then set

$$\text{mini-batch size} = 8, \qquad \text{accumulation steps} = 4.$$

The optimizer updates parameters once every 4 mini-batches, achieving the same gradient effect as a batch of 32. A key corollary is **learning-rate scaling**: when the batch size is multiplied by $k$, it is common (and often empirically effective) to multiply the learning rate by $k$ as well.[20]

### 4.5.3 Precision Reduction and Efficient Attention

Modern GPU architectures support reduced-precision arithmetic (e.g., **BF16** and **FP8**).[21] For many deep learning models, training in reduced precision yields minimal accuracy loss while substantially improving throughput and lowering memory usage. These gains can enable larger batch sizes or larger models within the same hardware budget.

Another major performance bottleneck in large language models is attention computation. Recent advances, such as **FlashAttention** (Dao et al., 2022), reformulate attention to reduce memory overhead and improve GPU utilization, making it a de facto standard in high-performance transformer implementations.

### 4.5.4 Optimization Hygiene: Schedules and Initialization

Compute efficiency also depends on *convergence efficiency*. A poorly initialized model or an unstable learning-rate schedule can waste GPU cycles. Empirically, the following

[20] The rationale follows from the variance-of-a-mean argument used above for SGD: a $k$-times larger batch produces a gradient estimate whose variance is roughly $k$ times smaller, i.e. a more reliable descent direction, so one can afford a proportionally larger step without the update being swamped by noise. This "linear scaling rule" was popularized by P. Goyal et al., "Accurate, large minibatch SGD: training ImageNet in 1 hour," 2017, https://arxiv.org/abs/1706.02677; a $\sqrt{k}$ rule is sometimes preferred instead, and very large batches eventually break the heuristic.

[21] These labels indicate how many bits encode each number. Standard "full precision" is FP32 (32-bit floating point); FP16 and BF16 use 16 bits and FP8 uses 8. Fewer bits mean less memory and faster arithmetic but coarser rounding. BF16 ("brain float") keeps the same exponent range as FP32 while sacrificing mantissa precision, which makes it more numerically stable for training than ordinary FP16; TF32, mentioned earlier in this chapter, is an NVIDIA format internally wider than FP16 but narrower than FP32.

practices are widely used:

- **He initialization** for ReLU-based networks to stabilize the variance of activations.
- **Learning-rate schedules** (e.g., cosine decay, warmup) to accelerate early learning and avoid late-stage oscillations.
- **AdamW** as the default optimizer for transformer-style architectures, due to its stable convergence and decoupled weight decay.

Taken together, these practices highlight a central lesson: **efficient training is a system-level problem**, not a single algorithmic trick. Researchers who design experiments with hardware constraints in mind can iterate faster, explore larger design spaces, and produce more reproducible computational results.

# Chapter 5: Introduction to Reinforcement Learning

This chapter develops the foundations of reinforcement learning (RL), beginning with the simplest decision problem under uncertainty, the multi-armed bandit, and building up to the full framework of Markov decision processes (MDPs), their value functions, and the classical dynamic-programming and linear-programming methods for solving them. We close by connecting these tools to the structural estimation of dynamic discrete-choice models in econometrics.

## 5.1 Introduction: The New Era of Artificial Intelligence

The field of Artificial Intelligence (AI) is undergoing a significant transformation. As noted by pioneers like Richard Sutton and David Silver, we are entering an "Era of Experience."[22] This new era is marked by a shift away from reliance on human-provided data and towards systems that learn and improve through direct interaction with their environment. This paradigm shift is encapsulated in the ancient proverb, "Knowledge from books is shallow; to truly understand, one must practice." Reinforcement Learning (RL) is at the heart of this evolution, providing the theoretical and algorithmic framework for creating agents that learn from experience.

Shunyu Yao has characterized the current moment as "AI's halftime."[23] The first half of AI saw remarkable successes in solving well-defined problems with clear rules and objectives, such as chess and Go. These achievements were driven by innovations in search algorithms, deep learning, and large-scale computation. However, the

[22] R. S. Sutton and D. Silver, *Welcome to the Era of Experience*, 2025. https://www.youtube.com/watch?v=b_h3QO33-8A

[23] S. Yao, *The Second Half of AI*, 2025. https://xbench.org/

second half of AI, which is beginning now, will be defined by a move from *solving* problems to *defining* them. In this new landscape, the ability of an agent to learn and adapt in complex, dynamic environments becomes paramount. This is where RL truly shines, as it provides a mechanism for agents to learn to make good sequences of decisions, even in the absence of complete information or a perfect model of the world.

### 5.1.1 Why Reinforcement Learning?

The fundamental advantage of RL over other machine learning paradigms, such as Supervised Learning (SL), lies in its ability to generalize and discover novel strategies. As demonstrated by the performance of systems like AlphaGo Zero (Silver et al., 2017) and recent research on language models,[24] an agent that learns through RL can surpass the performance of its teachers. While Supervised Fine-Tuning (SFT) is effective at memorizing patterns from a given dataset, RL enables an agent to generalize its knowledge to new, unseen situations. This is because RL is an active learning process: the agent is not a passive recipient of data but an active participant that explores its environment, makes decisions, and learns from the consequences of its actions.

This ability to learn through trial and error is what allows RL agents to achieve superhuman performance. For example, DeepSeek-R1, a large language model trained with RL, has demonstrated the ability to reason and solve complex mathematical problems, even discovering novel solutions and correcting its own mistakes along the way (DeepSeek-AI, 2025). This self-evolving intelligence is a hallmark of RL and a key differentiator from other machine learning approaches.

## 5.2 Notation and Conventions

Before developing the theory, we fix the notation used throughout the reinforcement-learning chapters of these notes, from multi-armed bandits through deep reinforcement learning. The notation follows Sutton and Barto (2018).

**Value functions.** We write the true value functions under a policy as $v_\pi(s)$ and $q_\pi(s,a)$, reserving the upper-case $V(s)$, $Q(s,a)$ (or the hatted $\hat{v}$, $\hat{q}$) for their *estimates*, and $v_*$, $q_*$ for the optimal values. Some chapters and much of the deep reinforcement-learning literature write the same true value functions with an upper-case superscript, $V^\pi(s)$ and $Q^\pi(s,a)$; these denote exactly the same objects as $v_\pi(s)$ and $q_\pi(s,a)$, and the two forms should be read interchangeably.

[24] L. Gao et al., *SFT Memorizes, RL Generalizes*, arXiv:2501.17161, 2025. https://arxiv.org/abs/2501.17161

| Symbol | Meaning |
|---|---|
| $s \in \mathcal{S}$ | state; $\mathcal{S}$ is the state space |
| $a \in \mathcal{A}$ | action; $\mathcal{A}$ (or $\mathcal{A}(s)$) is the action space |
| $r$, $R(s,a)$ | reward; $R_{t+1}$ is the reward received after acting in $s_t$ |
| $p(s' \mid s,a)$ | transition kernel (environment dynamics) |
| $\gamma \in [0,1]$ | discount factor |
| $\pi(a \mid s)$, $\pi_\theta(a \mid s)$ | (parameterized) policy: a distribution over actions given a state |
| $G_t$ | return (cumulative, discounted reward from time $t$) |
| $v_\pi(s)$, $q_\pi(s,a)$ | state- and action-value functions under policy $\pi$ |
| $v_*(s)$, $q_*(s,a)$ | optimal value functions |
| $V(s)$, $Q(s,a)$, $\hat{v}$, $\hat{q}$ | *estimates* of the value functions (tabular or approximate) |
| $A_\pi(s,a) = q_\pi(s,a) - v_\pi(s)$ | advantage function |
| $\delta_t$ | temporal-difference (TD) error |
| $\alpha$, $\epsilon$ | step size (learning rate) and exploration rate |
| $\tau$ | a trajectory $(s_0, a_0, r_1, \dots)$ |
| $J(\theta)$ | policy performance objective |

**Table 5.2.1.** Core reinforcement-learning notation used throughout these notes.

**Reward indexing.** In the foundational chapters (bandits, MDPs, dynamic programming, and model-free methods) the return uses the standard time-shifted convention $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$, so that $R_{t+1}$ is the reward earned for acting in state $s_t$. In the policy-gradient and deep reinforcement-learning material we follow the convention prevailing in that literature and write the (often undiscounted) *reward-to-go* as $G_t = \sum_{t'=t}^{T-1} r_{t'}$, with the per-step reward written $r_t$. The two conventions describe the same quantity up to the index shift and are used only to match the primary sources of each topic.

## 5.3 The Simplest Form of Reinforcement Learning: Multi-Armed Bandits

To understand the core principles of RL, we can start with the simplest possible setting: the Multi-Armed Bandit (MAB) problem. Imagine a gambler in a casino facing a row of slot machines, each with a different, unknown probability of paying out a reward. The gambler has a limited number of pulls and wants to maximize their total winnings. This is the essence of the MAB problem: how to allocate a limited set of resources among several competing choices to maximize the expected gain, when the

properties of these choices are not known in advance.

In the MAB framework, each slot machine is an "arm," and the decision of which arm to pull at each time step is the central challenge. Formally, there are $K$ arms; pulling arm $a$ yields a stochastic reward drawn from a fixed but unknown distribution with mean $\mu_a$. At each round $t = 1, 2, \ldots, T$ the agent selects an arm $a_t \in \{1, \ldots, K\}$ and observes a reward $r_t$. The objective is to maximize the cumulative reward $\sum_{t=1}^{T} r_t$, or equivalently to minimize the *regret*

$$\text{Regret}(T) \;=\; T\mu^* - \mathbb{E}\left[\sum_{t=1}^{T} r_t\right], \qquad \mu^* = \max_a \mu_a. \tag{5.3.1}$$

This problem encapsulates the fundamental trade-off in RL: the **exploitation-exploration dilemma**.

- **Exploitation** involves choosing the arm that currently appears to be the best, based on past experience. This is the strategy of maximizing the immediate, known reward.
- **Exploration** involves trying out other arms to gather more information about their potential rewards. This may lead to a lower immediate reward but could result in discovering a better arm in the long run.

### 5.3.1 Naive Approaches and the Need for a Strategy

A simple, yet flawed, approach is the **greedy algorithm**. This strategy involves always choosing the arm with the highest estimated reward $\hat{\mu}_a$ based on the data collected so far. The problem with a purely greedy approach is that it can get stuck in a suboptimal strategy. If, due to random chance, a suboptimal arm initially appears to be the best, the greedy algorithm will continue to choose it, never exploring other arms that might be better. This leads to a situation where the agent is locked into a wrong decision forever.

To overcome this, we need a more sophisticated strategy that balances exploitation and exploration. The $\epsilon$**-greedy** algorithm is a simple yet effective way to achieve this. With a small probability $\epsilon$, the agent chooses an arm at random (exploration), and with probability $1 - \epsilon$, it chooses the best-known arm (exploitation):

$$a_t = \begin{cases} \arg\max_a \hat{\mu}_a & \text{with probability } 1 - \epsilon, \\ \text{a uniformly random arm} & \text{with probability } \epsilon. \end{cases} \tag{5.3.2}$$

The value of $\epsilon$ can be decreased over time, as the agent becomes more confident in its estimates of the rewards.

### 5.3.2 Advanced Strategies: Optimism in the Face of Uncertainty

More advanced MAB algorithms are based on the principle of **Optimism in the Face of Uncertainty (OFU)**. The core idea is that the more uncertain we are about the reward of an arm, the higher the priority we should give to exploring it. Two popular algorithms that implement this principle are the **Upper Confidence Bound (UCB)** and **Thompson Sampling (TS)**.

- **UCB** works by calculating an upper confidence bound for the reward of each arm. This bound is a combination of the current estimated reward and an uncertainty term that is larger for arms that have been tried less frequently. A canonical version, UCB1 (Auer et al., 2002), selects at round $t$ the arm

$$a_t \;=\; \arg\max_a \left[\, \hat{\mu}_a \;+\; \sqrt{\tfrac{2\ln t}{N_a}} \,\right], \tag{5.3.3}$$

  where $\hat{\mu}_a$ is the empirical mean reward of arm $a$ and $N_a$ is the number of times arm $a$ has been pulled so far. The first term encourages exploitation while the second "exploration bonus" shrinks as an arm is pulled more often, so the algorithm chooses the arm with the highest *optimistic* estimate.

- **Thompson Sampling** is a Bayesian approach where we maintain a probability distribution for the reward of each arm. At each step, we sample a reward value from each arm's distribution and then choose the arm with the highest sampled value. This naturally balances exploration and exploitation, as arms with higher uncertainty will have a wider distribution, giving them a chance to be selected even if their current mean estimate is not the highest.

**The exploration bonus is a confidence half-width.** The UCB rule will look familiar to anyone who has constructed confidence intervals. The bonus $\sqrt{2\ln t / N_a}$ in (5.3.3) is, up to constants, the half-width of a confidence interval for the unknown mean $\mu_a$: by a Hoeffding concentration inequality, the empirical mean $\hat{\mu}_a$ of $N_a$ bounded, i.i.d. draws deviates from the true mean $\mu_a$ by more than $\sqrt{2\ln t / N_a}$ with probability at most $t^{-4}$. Thus $\hat{\mu}_a + \sqrt{2\ln t / N_a}$ is a high-probability *upper* confidence limit, and "optimism in the face of uncertainty" means: act as if each arm is as good as its data plausibly allow. Arms pulled rarely have small $N_a$ and hence wide intervals (a large bonus), so they keep being selected until either their estimate falls or enough data accumulate to rule them out—the same logic by which a large standard error invites further sampling. Thompson sampling implements the same instinct in a Bayesian register: maintaining a posterior over each $\mu_a$ and picking the arm with the largest posterior *draw* selects an arm with probability equal to the posterior probability that it

is best, so posterior uncertainty (a wide posterior) automatically generates exploration and shrinks as evidence accumulates.

Both UCB and TS have been shown to be highly effective in practice and have strong theoretical guarantees on their performance, as measured by **regret** (the difference between the reward obtained by the algorithm and the reward that would have been obtained by an optimal strategy, as defined in Equation (5.3.1)). The foundational analysis of asymptotically efficient allocation rules for this problem is due to Lai and Robbins (1985), who established the logarithmic lower bound on regret that algorithms like UCB attain.

### 5.3.3 MAB in Action: Real-World Applications

The MAB framework has found numerous applications in business and technology. It is widely used in online advertising to dynamically allocate ad impressions to different ad creatives to maximize click-through or conversion rates (Schwartz et al., 2017; Ye et al., 2023). Other applications include dynamic pricing (Misra et al., 2019), personalized recommendations (Aramayo et al., 2022), and A/B testing. These applications demonstrate the power of MABs to solve real-world decision-making problems under uncertainty.

*Remark* (Bandits as adaptive experimentation). For an empirical researcher, the multi-armed bandit is best understood as *adaptive* (sequential) experimentation, and it is illuminating to contrast it with the fixed-design A/B test. A classical A/B test allocates a pre-committed fraction of traffic to each arm, runs to a planned sample size, and only then picks a winner; it is optimized for a clean hypothesis test about the treatment effect. A bandit instead updates the allocation as data arrive, steering traffic toward arms that currently look better. The two designs optimize different objectives: the A/B test minimizes the variance of the estimated effect (statistical power), whereas the bandit minimizes cumulative regret (5.3.1)—the opportunity cost, in lost reward, of every suboptimal pull made while learning. This is the exploration–exploitation tension seen from the experimenter's side: pure exploitation is a myopic "always ship the current leader" rule, pure exploration is a balanced A/B test, and UCB or Thompson sampling interpolate between the two. The trade-off is real—adaptive allocation can complicate the post-hoc inference that fixed randomized designs make easy—which is why bandit and experimentation methods are an active topic at the interface of marketing, econometrics, and machine learning.

## 5.4 Sequential Decision Making: Markov Decision Processes

While MABs are a useful model for single-step decision problems, many real-world problems involve a sequence of decisions where the action taken at one step can influence the state of the world and the available actions at future steps. To model these more complex scenarios, we use the framework of **Markov Decision Processes (MDPs)**.

An MDP is a mathematical framework for modeling decision-making in situations where outcomes are partly random and partly under the control of a decision-maker. The core assumption of an MDP is the **Markov property**, which states that the future is independent of the past, given the present. In other words, the current state of the system contains all the information needed to make a decision, and the history of how the system arrived at that state is irrelevant. Formally, the Markov property requires

$$\Pr\big(s_{t+1} \mid s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_0, a_0\big) \;=\; \Pr\big(s_{t+1} \mid s_t, a_t\big). \tag{5.4.1}$$

**The state as a sufficient statistic.** The Markov property (5.4.1) is the dynamic analogue of a notion familiar from econometrics: the current state $s_t$ is a *sufficient statistic* for the future of the process. Given $s_t$ (and the action $a_t$), the entire past history $(s_{t-1}, a_{t-1}, \ldots, s_0, a_0)$ carries no additional predictive information about $s_{t+1}$ or any later state. This is what makes the problem tractable: rather than conditioning on an ever-growing history, the agent need only track a fixed-dimensional state. When a raw observation is *not* Markov—for instance, a single video frame does not reveal velocity, and a customer's current basket may not reveal their purchase history—one enlarges the state (stacking several frames, or appending the relevant summary of the past) until the Markov property is restored. Choosing such a state is the modeling counterpart of selecting the right conditioning set in a regression.

The agent and environment interact in a closed loop: at each time step the agent observes the current state $s_t$, selects an action $a_t$, and the environment responds with a reward $r_{t+1}$ and a next state $s_{t+1}$. This interaction loop is depicted in Figure 5.4.1.

### 5.4.1 The Components of an MDP

An MDP is formally defined by a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where:

- $\mathcal{S}$ is a set of states.
- $\mathcal{A}$ is a set of actions.

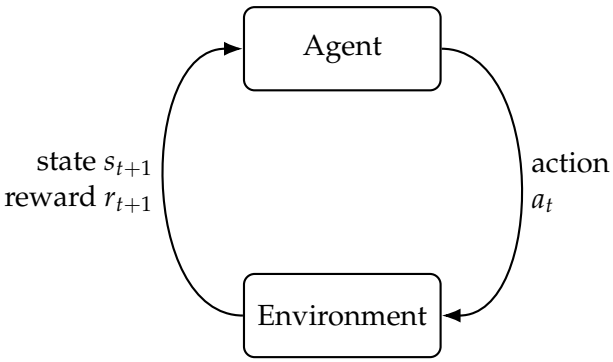


**Figure 5.4.1.** The agent-environment interaction loop in a Markov decision process. At each step the agent takes an action $a_t$ in state $s_t$; the environment returns a reward $r_{t+1}$ and a new state $s_{t+1}$.

- $P$ is the state transition probability function, $P(s' \mid s, a)$, which gives the probability of transitioning to state $s'$ from state $s$ after taking action $a$.
- $R$ is the reward function, $R(s, a, s')$, which gives the reward received after transitioning from state $s$ to state $s'$ as a result of taking action $a$.
- $\gamma$ (gamma) is the discount factor, a number $\gamma \in [0, 1]$ that represents the preference for immediate rewards over future rewards.

**Definition 5.4.1** (Markov Decision Process)**.** A *Markov decision process* is a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, transition kernel $P(s' \mid s, a)$, reward function $R(s, a, s')$, and discount factor $\gamma \in [0, 1]$, in which transitions satisfy the Markov property (5.4.1).

### 5.4.2 From Markov Processes to MDPs

To understand MDPs, it is helpful to build up from simpler concepts:

1. **Markov Process (or Markov Chain):** This is the simplest model, consisting of a set of states and a transition probability matrix that defines the probability of moving from one state to another. There are no actions or rewards in a Markov Process.
2. **Markov Reward Process (MRP):** An MRP adds a reward function and a discount factor to a Markov Process. This allows us to calculate the **value** of being in a particular state, which is defined as the expected total discounted reward from that state onwards.
3. **Markov Decision Process (MDP):** An MDP adds a set of actions to an MRP, giving the agent control over the transitions between states. The goal of the

agent in an MDP is to learn a **policy** that maximizes the expected total discounted reward.

In each case the quantity of interest is the discounted **return** accumulated from time $t$ onwards,

$$G_t \;=\; r_{t+1} + \gamma\, r_{t+2} + \gamma^2 r_{t+3} + \cdots \;=\; \sum_{k=0}^{\infty} \gamma^k\, r_{t+k+1}. \tag{5.4.2}$$

*Remark* (Discounting as present value). The discount factor $\gamma$ plays the same role here as the subjective discount factor $\beta = 1/(1+\rho)$ familiar from dynamic economic models, where $\rho$ is a per-period rate of time preference (or interest rate): a reward of one unit received $k$ periods from now is worth only $\gamma^k$ units today, so the return $G_t$ is literally the present value of the future reward stream. Two consequences are worth noting. First, discounting makes the infinite sum well behaved: if rewards are bounded, $|r| \le R_{\max}$, then the geometric series gives $|G_t| \le R_{\max} \sum_{k=0}^{\infty} \gamma^k = R_{\max}/(1-\gamma) < \infty$ whenever $\gamma < 1$. Second, the quantity $1/(1-\gamma)$ acts as an *effective horizon*: with $\gamma = 0.99$ the agent effectively plans about 100 steps ahead, whereas $\gamma \to 1$ pushes the horizon toward infinity and, as we will see, slows every solution method.

### 5.4.3 Policies and Value Functions

A **policy**, denoted by $\pi$, is a mapping from states to actions (in general, a distribution $\pi(a \mid s)$ over actions in each state). It specifies what action the agent should take in each state. The goal of RL is to find an optimal policy, $\pi^*$, that maximizes the expected return.

To evaluate a policy, we use **value functions**. There are two main types of value functions:

- **State-value function** $V^\pi(s)$**:** the expected return starting from state $s$ and following policy $\pi$,

$$V^\pi(s) \;=\; \mathbb{E}_\pi[\, G_t \mid s_t = s \,] \;=\; \mathbb{E}_\pi\left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \;\middle|\; s_t = s \right]. \tag{5.4.3}$$

- **Action-value function** $Q^\pi(s,a)$**:** the expected return starting from state $s$, taking action $a$, and then following policy $\pi$,

$$Q^\pi(s,a) \;=\; \mathbb{E}_\pi[\, G_t \mid s_t = s,\, a_t = a \,]. \tag{5.4.4}$$

These value functions can be calculated using the **Bellman equations**, which ex-

press the value of a state in terms of the values of its successor states:

$$V^{\pi}(s) \;=\; \sum_{a} \pi(a \mid s) \sum_{s'} P(s' \mid s, a) \Big[ R(s, a, s') + \gamma\, V^{\pi}(s') \Big]. \tag{5.4.5}$$

The Bellman equations form the basis for many RL algorithms.

**Where the Bellman equation comes from.** Equation (5.4.5) is not a new assumption but a direct consequence of the recursive structure of the return together with the law of iterated expectations. Split the return into its first term and the rest, $G_t = r_{t+1} + \gamma G_{t+1}$, take the conditional expectation under $\pi$ given $s_t = s$, and use linearity:

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}_{\pi}[\, r_{t+1} + \gamma G_{t+1} \mid s_t = s \,] \\ &= \mathbb{E}_{\pi}[\, r_{t+1} \mid s_t = s \,] + \gamma\, \mathbb{E}_{\pi}[\, \mathbb{E}_{\pi}[G_{t+1} \mid s_{t+1}] \mid s_t = s \,]. \end{aligned}$$

The inner expectation is just $V^{\pi}(s_{t+1})$ by the definition (5.4.3), and averaging over the action $a \sim \pi(\cdot \mid s)$ and the next state $s' \sim P(\cdot \mid s, a)$ reproduces (5.4.5). Economists will recognize this as exactly the recursive form of a value function in dynamic programming: it is the same "today's value = expected current payoff + discounted continuation value" logic that underlies the value function of an optimal-stopping or dynamic-investment problem, and the maximizing version (5.4.8) is the counterpart of the agent's first-order (Euler) optimality condition.

### 5.4.4 Finding the Optimal Policy

The ultimate goal in an MDP is to find the optimal policy, $\pi^*$. The optimal policy is the one that achieves the highest possible expected return from every state. The corresponding value functions are called the optimal value functions, $V^*(s)$ and $Q^*(s, a)$, defined by

$$V^*(s) = \max_{\pi} V^{\pi}(s), \qquad Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a). \tag{5.4.6}$$

**Theorem 5.4.2** (Existence of an optimal policy)**.** *For any finite MDP there exists at least one optimal (deterministic) policy $\pi^*$ that is simultaneously optimal in every state. Once we have the optimal action-value function $Q^*(s, a)$, the optimal policy is simply to choose the action that maximizes $Q^*(s, a)$ in each state:*

$$\pi^*(s) \;=\; \arg\max_{a} Q^*(s, a). \tag{5.4.7}$$

The optimal value functions can be found by solving the **Bellman optimality equations**. These equations are similar to the Bellman equations for a given policy, but they

include a maximization step over all possible actions:

$$V^*(s) \;=\; \max_a \sum_{s'} P(s' \mid s,a)\Big[\, R(s,a,s') + \gamma\, V^*(s') \,\Big], \tag{5.4.8}$$

and, equivalently, for the action-value function,

$$Q^*(s,a) \;=\; \sum_{s'} P(s' \mid s,a)\Big[\, R(s,a,s') + \gamma\, \max_{a'} Q^*(s',a') \,\Big]. \tag{5.4.9}$$

## 5.5 Solving MDPs with Dynamic Programming

When the full model of the MDP is known (i.e., we know the transition probabilities and reward function), we can use dynamic programming methods to find the optimal policy. The foundational treatment of these methods is due to Bellman (1957); a modern, comprehensive reference is Sutton and Barto (2018). The two main dynamic programming algorithms for solving MDPs are **Value Iteration** and **Policy Iteration**.

### 5.5.1 Value Iteration

Value Iteration is an iterative algorithm that starts with an arbitrary value function and repeatedly updates it using the Bellman optimality equation. The algorithm converges to the optimal value function, from which the optimal policy can be easily extracted. The update rule for Value Iteration is

$$V_{k+1}(s) \;=\; \max_a \sum_{s'} P(s' \mid s,a)\Big[\, R(s,a,s') + \gamma\, V_k(s') \,\Big]. \tag{5.5.1}$$

This process is repeated until the value function converges. Algorithm 2 states the procedure precisely.

**Algorithm 2** Value Iteration

Initialize $V_0(s)$ arbitrarily for all $s \in \mathcal{S}$; choose tolerance $\theta > 0$
**repeat**
  $\Delta \leftarrow 0$
  **for** each state $s \in \mathcal{S}$ **do**
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s'} P(s' \mid s,a)\big[R(s,a,s') + \gamma V(s')\big]$
    $\Delta \leftarrow \max\big(\Delta,\, |v - V(s)|\big)$
  **end for**
**until** $\Delta < \theta$
**return** the greedy policy $\pi(s) = \arg\max_a \sum_{s'} P(s' \mid s,a)\big[R(s,a,s') + \gamma V(s')\big]$

### 5.5.2 Policy Iteration

Policy Iteration is another iterative algorithm that alternates between two steps:

1. **Policy Evaluation:** Given a policy $\pi$, calculate the corresponding value function $V^\pi$. This is done by solving a system of linear equations or by iteratively applying the Bellman equation (5.4.5) for the given policy until convergence.

2. **Policy Improvement:** Improve the policy by acting greedily with respect to the current value function. For each state, we choose the action that maximizes the expected return based on the current value function:

$$\pi'(s) \;=\; \arg\max_a \sum_{s'} P(s' \mid s,a)\Big[\, R(s,a,s') + \gamma\, V^\pi(s') \,\Big]. \tag{5.5.2}$$

**Policy evaluation is a linear solve.** For a *fixed* policy $\pi$ the Bellman equation (5.4.5) drops the max and becomes *linear* in the unknown values. Stacking the states into a vector $V^\pi \in \mathbb{R}^{|\mathcal{S}|}$, writing $P_\pi$ for the $|\mathcal{S}| \times |\mathcal{S}|$ transition matrix induced by $\pi$ (with entries $\sum_a \pi(a \mid s)\, P(s' \mid s,a)$) and $R_\pi$ for the vector of expected one-step rewards under $\pi$, the equation reads $V^\pi = R_\pi + \gamma P_\pi V^\pi$, whose unique solution is

$$V^\pi = (I - \gamma P_\pi)^{-1} R_\pi.$$

The inverse exists because $\gamma P_\pi$ has spectral radius at most $\gamma < 1$. This is the "solving a system of linear equations" alluded to above—structurally the same kind of closed form as the OLS normal equations—and the Neumann-series expansion $(I - \gamma P_\pi)^{-1} = \sum_{k=0}^{\infty} (\gamma P_\pi)^k$ recovers the definition of the value as a discounted sum of expected future rewards. Iterating the Bellman equation instead of inverting the $|\mathcal{S}| \times |\mathcal{S}|$ matrix is the cheaper alternative used in practice when the state space is large.

This process of policy evaluation and improvement is repeated until the policy no longer changes, at which point we have found the optimal policy (Algorithm 3). The convergence of policy iteration is often visualized in a simple environment like a GridWorld, where the initial random policy quickly evolves into an optimal path-finding strategy within a few iterations.

### 5.5.3 A Comparison of Dynamic Programming Methods

Both Value Iteration and Policy Iteration are guaranteed to converge to the optimal policy, but they do so in different ways. The choice between them involves a trade-off between the number of iterations and the computational cost per iteration, summarized in Table 5.5.1.

**Algorithm 3** Policy Iteration

Initialize policy $\pi$ arbitrarily for all $s \in \mathcal{S}$
**repeat**
  **Policy Evaluation:** compute $V^\pi$ by solving, for all $s$,
    $V^\pi(s) = \sum_{s'} P(s' \mid s, \pi(s)) \big[R(s, \pi(s), s') + \gamma V^\pi(s')\big]$
  **Policy Improvement:** *stable* $\leftarrow$ **true**
  **for** each state $s \in \mathcal{S}$ **do**
    $a_{\text{old}} \leftarrow \pi(s)$
    $\pi(s) \leftarrow \arg\max_a \sum_{s'} P(s' \mid s, a) \big[R(s, a, s') + \gamma V^\pi(s')\big]$
    **if** $a_{\text{old}} \neq \pi(s)$ **then**
      *stable* $\leftarrow$ **false**
    **end if**
  **end for**
**until** *stable* is **true**
**return** $\pi$ and $V^\pi$

In practice, **Policy Iteration is often faster than Value Iteration** because it can converge in a much smaller number of iterations. However, it is important to note that if the discount factor $\gamma$ is close to 1, both methods can become slow.

*Remark* (Why these methods converge—and why $\gamma \to 1$ hurts). Both algorithms inherit their guarantees from a single fact: the Bellman optimality operator $(\mathcal{T}V)(s) = \max_a \sum_{s'} P(s' \mid s, a) \big[R(s, a, s') + \gamma V(s')\big]$ is a $\gamma$-contraction in the sup-norm, i.e. $\|\mathcal{T}V - \mathcal{T}V'\|_\infty \leq \gamma \|V - V'\|_\infty$ for any two value functions $V, V'$. Value iteration (5.5.1) is precisely the fixed-point iteration $V_{k+1} = \mathcal{T}V_k$, so by the Banach fixed-point theorem it converges geometrically to the unique fixed point $V^*$, with error shrinking like $\gamma^k$: $\|V_k - V^*\|_\infty \leq \gamma^k \|V_0 - V^*\|_\infty$. This is why convergence is "asymptotic"—the error decays geometrically but never reaches exactly zero in finitely many steps. Policy iteration, by contrast, terminates *exactly* after finitely many steps: each policy is a deterministic map from a finite state space to a finite action space, so there are only finitely many policies; the policy-improvement step (5.5.2) never decreases the value of any state and strictly improves at least one until no further change is possible, so the iteration cannot cycle and must reach $\pi^*$. (Policy iteration is Howard's algorithm and can be read as Newton's method applied to the Bellman optimality equation, which explains its characteristically fast, few-iteration convergence.) Finally, both methods slow down as $\gamma \to 1$ because the contraction factor $\gamma$ governs the per-iteration error reduction: the effective horizon $1/(1-\gamma)$ blows up, so many more iterations—or, in policy evaluation, a near-singular matrix $I - \gamma P_\pi$ to solve—are required.

**Table 5.5.1.** Comparison of Value Iteration and Policy Iteration.

| Algorithm | Pros | Cons |
|---|---|---|
| **Value Iteration** | Each iteration is computationally fast. | Convergence can be slow and is asymptotic, meaning it gets closer and closer to the optimum without necessarily reaching it in a finite number of steps. |
| **Policy Iteration** | Guaranteed to converge in a finite number of iterations, and often converges very quickly in practice. | The policy evaluation step in each iteration can be computationally expensive, as it involves solving a system of linear equations or running an iterative process until convergence. |

## 5.6 An Alternative Perspective: Linear Programming

Beyond dynamic programming, MDPs can also be solved using **Linear Programming (LP)**. This approach provides a different mathematical perspective on the problem of finding the optimal value function. The core idea is to formulate the Bellman optimality equation as a set of constraints in an LP problem.

The primal form of the LP is formulated as follows:

$$\begin{aligned} \text{minimize}_{\,V} \quad & \sum_{s} d_0(s)\, V(s) \\ \text{subject to} \quad & V(s) \;\geq\; R(s,a) + \gamma \sum_{s'} P(s' \mid s,a)\, V(s') \qquad \forall\, s \in \mathcal{S},\; a \in \mathcal{A}. \end{aligned} \tag{5.6.1}$$

Here, the $V(s)$ are the decision variables, representing the value of each state, and $d_0(s)$ is an initial state distribution. The constraints ensure that the value function satisfies the Bellman optimality condition. The optimal solution to this LP is the optimal value function of the MDP.

This LP has $|\mathcal{S}|$ decision variables and $|\mathcal{S}| \times |\mathcal{A}|$ linear constraints. By taking the dual of this primal LP, we can obtain a formulation where the variables correspond to state-action visitation frequencies, and the optimal solution to the dual LP can be used to directly derive the optimal policy.

**Reading the primal and the dual.** Two remarks make this LP less mysterious. First, the reward is written $R(s,a)$ here—the *expected* one-step reward $R(s,a) = \sum_{s'} P(s' \mid s,a)\, R(s,a,s')$—which is why it sits outside the inner sum over $s'$; it is the same re-

ward used throughout, just pre-averaged over next states. Second, the primal asks for the smallest value function that dominates every one-step Bellman "lookahead": the constraints force $V \geq \mathcal{T}V$ component-wise, and minimizing the positive combination $\sum_s d_0(s)\, V(s)$ (with $d_0(s) > 0$) pins $V$ down to exactly the optimal value $V^*$. The dual variables, one per constraint $(s, a)$, turn out to be the *discounted state–action occupancy measure*—the expected discounted number of visits to $(s, a)$, $\sum_{t \geq 0} \gamma^t \Pr(s_t = s,\, a_t = a)$. Economists can read the pair as a value/quantity duality: the primal prices each state, while the dual chooses visitation "flows" that maximize total reward subject to flow-conservation (Bellman-flow) constraints, and complementary slackness ties the two together so that the dual's positive flows trace out the optimal policy.

## 5.7 Connections to Econometrics: Structural Estimation

The concepts and methods developed for solving MDPs have found significant applications in the field of econometrics, particularly in the **structural estimation of dynamic discrete choice models**. These models are used to analyze the behavior of individuals or firms making sequential decisions over time, such as a consumer's decision to purchase a product or a firm's decision to invest in a new technology.

Estimating these structural models is often computationally challenging. A traditional approach is the **nested fixed-point (NFXP)** algorithm,[25] which involves repeatedly solving for the value function (the fixed point) within an optimization routine that searches for the model's parameters. This can be very slow.

In a seminal paper, Su and Judd (2012) proposed a more efficient approach based on **constrained optimization**. Instead of nesting the fixed-point calculation, they formulate the estimation problem as a single large optimization problem where the Bellman equation is included as a set of constraints. This approach, often referred to as **Mathematical Programming with Equilibrium Constraints (MPEC)**, can be significantly faster than NFXP because it avoids the repeated solving of the structural model for every guess of the parameters.

**What is really being estimated.** For an econometrician the setup is a maximum likelihood problem with an equilibrium (fixed-point) constraint. The structural parameters $\theta$—the utility/reward primitives and, where identified, the discount factor—enter the agent's dynamic program, whose solution is a value function (or, in discrete-choice form, a vector of conditional choice probabilities); the likelihood of the observed choice data is then a function of those model-implied probabilities. NFXP evaluates this like-

[25] The nested fixed-point estimator originates with J. Rust, *Optimal Replacement of GMC Bus Engines: An Empirical Model of Harold Zurcher*, *Econometrica* 55(5):999–1033, 1987. https://doi.org/10.2307/1911259

lihood by, at every trial value of $\theta$, fully solving the dynamic program (the "inner loop," a fixed point of exactly the Bellman type studied in this chapter) and feeding the resulting choice probabilities into the log-likelihood (the "outer loop" optimization over $\theta$). MPEC instead treats the value function $V$ as additional unknowns and maximizes the same likelihood over $(\theta, V)$ jointly, subject to the Bellman equation imposed as equality constraints—trading an expensive nested solve for a larger but smooth constrained optimization that modern solvers handle efficiently. The bridge to reinforcement learning is direct: the "inner loop" that structural econometrics must repeatedly solve is the very policy-evaluation / value-iteration computation of Section 5.5, and RL's model-free methods (next chapter) can be viewed as ways to carry out that step from data when $P$ and $R$ are unknown.

This connection highlights the deep interplay between reinforcement learning, optimization, and econometrics. The tools and techniques developed in one field can often be leveraged to solve important problems in another.

## 5.8 Conclusion

This lecture has provided an introduction to the fundamental concepts of Reinforcement Learning. We have seen how RL is driving a new era of AI, enabling agents to learn from experience and achieve superhuman performance. We have explored the core concepts of the exploitation-exploration trade-off, Markov Decision Processes, value functions, and policies. We have also discussed how to solve MDPs using dynamic programming methods when a model of the environment is available, and we have touched upon alternative solution methods like linear programming and the connections between RL and structural estimation in econometrics.

This is just the beginning of our journey into the fascinating world of RL. In future lectures, we will explore more advanced topics, including model-free RL methods that can be used when the model of the environment is unknown, as well as deep reinforcement learning, which combines the power of deep neural networks with RL to solve complex problems in high-dimensional state and action spaces.

# Chapter 6: Model-Free Reinforcement Learning

This chapter studies how an agent can learn to predict and to act *without* knowing the environment's transition dynamics or reward function. We progress from *prediction* (estimating the value of a fixed policy) to *control* (learning an optimal policy), and along the way we develop the Monte Carlo, temporal-difference, and $n$-step families of algorithms that form the backbone of modern reinforcement learning.

# Abstract

This chapter provides a comprehensive introduction to model-free reinforcement learning (RL), where agents learn from experience without knowing the transition dynamics or reward function. The content is based on the lecture slides from the course "DOTE 6635: Artificial Intelligence for Business Research" and is supplemented with additional explanations and references to the foundational literature (Sutton and Barto, 2018). We first discuss the core challenges of RL and the distinction between model-based and model-free approaches. We then cover **model-free prediction (policy evaluation)** via Monte Carlo (MC) methods, temporal-difference (TD) learning, and $n$**-step** methods that interpolate between them. Finally, we study **model-free control**, learning optimal behavior, via MC control with $\varepsilon$-greedy exploration (including GLIE schedules), and TD control algorithms such as **SARSA** (on-policy) and **Q-learning** (off-policy), highlighting the practical trade-offs behind their different targets.


## 6.1 Introduction to Model-Free Reinforcement Learning

### 6.1.1 Core Challenges of Reinforcement Learning

Before diving into specific algorithms, it is essential to understand the fundamental challenges that any RL system must address.

**Temporal Credit Assignment.** In sequential decision-making, rewards may arrive long after the actions that caused them. The challenge is determining which earlier actions deserve "credit" or "blame" for outcomes that occur later. For instance, in a game of chess, the winning move at the end of the game is a consequence of many strategic decisions made throughout the match.

**Exploration.** An agent must balance between exploiting known good actions and exploring potentially better alternatives. How can an agent efficiently explore to gain information about the environment without sacrificing too much reward?

**Generalization.** How can policies learned in one environment or set of states generalize to new, unseen situations? This is particularly important when the state space is vast or continuous.

### 6.1.2 Two Paradigms: RL for Planning vs. RL for Learning

Reinforcement learning manifests in two distinct paradigms, each with different assumptions and challenges.

**RL to Solve a Large Planning Problem.**

- Applications include AlphaGo, AlphaStar, simulated robotics, LLM coding, and LLM math reasoning.
- The transition dynamics are *known* (or can be simulated), but the state space is astronomically large.
- Data is collected by running a simulator.
- RL has proven highly successful in this domain.

**RL to Solve a Learning Problem.**

- Applications include adaptive medical treatment, deep research agents, and AI scientists.
- Transition dynamics are *unknown*, rewards are *unknown*, and there are too many states.
- Data must be collected by interacting with the real environment.
- This paradigm holds great potential but remains largely unrealized.

### 6.1.3 From Model-Based to Model-Free RL

In previous discussions of Markov Decision Processes (MDPs), we assumed complete knowledge of four critical components:

- **Action Space** $\mathcal{A}$: the set of all possible actions.
- **State Space** $\mathcal{S}$: the set of all possible states.
- **Transition Matrix** $P$: the probability of transitioning from one state to another given an action.
- **Reward Function** $R$: the immediate reward received after taking an action in a state.

With this complete information, algorithms like value iteration and policy iteration can compute the optimal policy through dynamic programming (Bellman, 1957). However, in many real-world scenarios, the transition matrix $P$ and reward function $R$ are unknown. **Model-free RL** addresses this challenge by learning directly from experience without explicitly modeling the environment dynamics.

## 6.2 Monte Carlo Policy Evaluation (Model-Free Prediction)

### 6.2.1 The Monte Carlo Approach

Monte Carlo (MC) methods represent the most intuitive approach to model-free learning. The core idea is elegantly simple: *the value of a state equals the expected cumulative reward starting from that state, so we can estimate it by averaging the actual returns observed from many sample episodes.*

Key characteristics of MC methods:

- **Model-free:** no knowledge of MDP transitions or rewards is required.
- **Experience-based:** learning occurs directly from episodes of interaction with the environment.
- **Episodic:** all episodes must terminate (reach a terminal state).
- **Simple estimation:** the value function is approximated by the empirical mean return.

### 6.2.2 MC Policy Evaluation

The objective of MC policy evaluation is to estimate the value function $v_\pi$ under a given policy $\pi$. Consider an episode generated by following policy $\pi$:

$$S_0, A_0, R_0, \ldots, S_T \sim \pi.$$

The return $G_t$ from time step $t$ is defined as

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{T-t-1} R_T.$$

The value function is then

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]\,. \tag{6.2.1}$$

The Monte Carlo idea is to use the **empirical mean return** to approximate the expected return in (6.2.1).

*Remark* (MC evaluation is just Monte Carlo integration / a sample mean). For a fixed policy $\pi$, the value $v_\pi(s)$ in (6.2.1) is an *expectation* of the random return $G_t$ over the randomness in the policy, the transitions, and the rewards. Estimating an expectation

by an average over independent draws is exactly Monte Carlo integration, and it is the same logic an econometrician uses when estimating a population mean $\mathbb{E}[Y]$ by the sample average $\frac{1}{N}\sum_i Y_i$. Each completed episode supplies one draw of the return for every state it visits; by the law of large numbers the running mean $V(s)$ converges to $v_\pi(s)$, and—for first-visit MC, whose per-episode returns to a given state are i.i.d. across episodes—a central limit theorem gives the usual $O(1/\sqrt{N(s)})$ standard error. The only thing that makes this "RL" rather than ordinary estimation is that the data are generated by simulating or interacting with the environment rather than handed to us as a fixed dataset.

### 6.2.3 First-Visit vs. Every-Visit MC

Two variants of MC policy evaluation exist, differing in how they handle states that appear multiple times within an episode.

**First-Visit MC Policy Evaluation.** The value $v_\pi(s)$ is estimated by averaging returns following only the *first* visit to state $s$ in each episode.

**Algorithm 4** First-Visit MC Policy Evaluation

**Initialize:** counter $N(s) \leftarrow 0$ for all $s \in \mathcal{S}$; Returns$(s) \leftarrow 0$ (running sum) for all $s \in \mathcal{S}$
**repeat**
  Generate an episode following policy $\pi$
  **for all** distinct states $s$ appearing in the episode **do**
    $G \leftarrow$ return following the *first* occurrence of $s$
    $N(s) \leftarrow N(s) + 1$
    Returns$(s) \leftarrow$ Returns$(s) + G$
  **end for**
**until** a stopping criterion is met
**Output:** for each distinct $s$, set $V(s) =$ Returns$(s)/N(s)$, so that $V(s) \to v_\pi(s)$ as $N(s) \to \infty$

*Remark* (First-visit consistency). First-visit MC evaluation is **unbiased**, and it converges to the true value function as the number of episodes approaches infinity.

**Every-Visit MC Policy Evaluation.** The value $v_\pi(s)$ is estimated by averaging returns following *every* visit to state $s$ across a set of episodes.

*Remark* (Every-visit consistency). Every-visit MC evaluation is **biased** (because returns from the same episode are correlated), but it still converges to the true value function. It typically achieves a **smaller mean squared error (MSE)** because the effective sample size is much larger.

**Algorithm 5** Every-Visit MC Policy Evaluation

**Initialize:** counter $N(s) \leftarrow 0$ for all $s \in \mathcal{S}$; Returns$(s) \leftarrow 0$ (running sum) for all $s \in \mathcal{S}$
**repeat**
  Generate an episode following policy $\pi$
  **for all** occurrences of each state $s$ in the episode **do**
    $G \leftarrow$ return following *this* occurrence of $s$
    $N(s) \leftarrow N(s) + 1$
    Returns$(s) \leftarrow$ Returns$(s) + G$
  **end for**
**until** a stopping criterion is met
**Output:** for each distinct $s$, set $V(s) =$ Returns$(s)/N(s)$, so that $V(s) \to v_\pi(s)$ as $N(s) \to \infty$

**Practical Note.** If there is an absorbing (terminal) state, the episode length is well-defined. Otherwise, one must specify a finite horizon $T$, which should be larger when the discount factor $\gamma$ is larger (to capture more of the discounted future rewards).

### 6.2.4 Incremental MC Policy Evaluation

Rather than storing all returns and computing averages at the end, we can update estimates incrementally. This approach reduces memory usage and is computationally more efficient.

The key insight comes from the incremental formula for computing means:

$$\mu_k = \frac{1}{k}\sum_{j=1}^{k} x_j = \frac{1}{k}\left(x_k + \sum_{j=1}^{k-1} x_j\right) = \frac{1}{k}\big(x_k + (k-1)\mu_{k-1}\big) = \mu_{k-1} + \frac{1}{k}\big(x_k - \mu_{k-1}\big). \tag{6.2.2}$$

This can be interpreted as: **new estimate** = **old estimate** + **step size** × **(observation** − **old estimate)**.

Applying this to MC policy evaluation, for each visited state $S_t$:

$$N(S_t) \leftarrow N(S_t) + 1, \tag{6.2.3}$$

$$V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)}\big(G_t - V(S_t)\big). \tag{6.2.4}$$

**Handling Non-Stationary Environments.** The incremental update enables adaptation to non-stationary environments where the underlying MDP may change over time. By replacing the decreasing step size $1/N(S_t)$ with a constant step size $\alpha$, we obtain

$$V(S_t) \leftarrow V(S_t) + \alpha\big(G_t - V(S_t)\big). \tag{6.2.5}$$

This formulation gives more weight to recent observations, allowing the algorithm to track changes in the environment. This is crucial for real-world applications where transition dynamics may evolve over time.

*Remark* (Running means, EWMA, and stochastic approximation). The update form "new estimate = old estimate + step size × (target − old estimate)" that we read off (6.2.2) is worth memorizing, because *every* algorithm in this chapter is an instance of it. With the decreasing step size $1/N(S_t)$ in (6.2.4), the recursion reproduces the *exact* running sample mean of the returns observed so far. With a *constant* step size $\alpha$ in (6.2.5), unrolling the recursion $V \leftarrow (1-\alpha)V + \alpha G$ shows that $V(S_t)$ becomes a geometrically weighted average of past returns—an **exponentially weighted moving average** (EWMA) that places weight $\alpha(1-\alpha)^k$ on the return seen $k$ updates ago. This is precisely the smoother used in time-series analysis to track a slowly drifting mean, which is why constant-$\alpha$ updates can follow a non-stationary environment. More abstractly, any update of this form is a **stochastic-approximation** (Robbins–Monro) scheme for solving a moment condition $\mathbb{E}[\,\text{target} - V\,] = 0$ one noisy draw at a time; the step size governs the trade-off between forgetting stale information quickly and averaging out sampling noise.

### 6.2.5 Example: Blackjack (Policy Evaluation)

**Example 6.2.1** (Blackjack policy evaluation)**.** A classic illustration of Monte Carlo policy evaluation is **Blackjack** (Sutton and Barto, 2018). One typically defines the state using compact summary statistics such as:

- the player's current sum,
- the dealer's showing card,
- whether the player has a usable ace.

Given a fixed policy, e.g., *stick if the player's sum is at least 20, otherwise hit*, we can simulate many games (episodes) and estimate $v_\pi(s)$ by averaging observed returns from visits to $s$ (first-visit or every-visit MC).

Why Blackjack is a useful teaching example:

- It is naturally **episodic** (each game ends).
- It is **stochastic** (random card draws), so averaging returns is meaningful.
- It is easy to **visualize** $v_\pi$ (e.g., heatmaps over player sum vs. dealer card) and to see how different policies change the value landscape.

## 6.3 Temporal-Difference Policy Evaluation (Model-Free Prediction)

### 6.3.1 The Central Idea of TD Learning

Temporal-difference (TD) learning is arguably the most important and novel contribution of reinforcement learning. As Sutton and Barto eloquently state (Sutton and Barto, 2018):

> "If one had to identify one idea as central and novel to reinforcement learning, it would undoubtedly be temporal-difference (TD) learning."

TD learning combines the advantages of Monte Carlo methods and dynamic programming (Sutton, 1988):

- Like MC, TD is **model-free** and learns directly from experience.
- Like dynamic programming, TD **bootstraps**[26], it updates estimates based on other estimates without waiting for a final outcome.
- TD can be applied to **both episodic and non-episodic (continuing)** settings.
- TD **updates a guess towards a guess**, using the Bellman equation structure.

### 6.3.2 From MC to TD: The Intuition

Recall the two equivalent formulations of the value function under policy $\pi$.

**Formulation 1 (MC perspective).**

$$V^{\pi}(s) = \mathbb{E}^{\pi}[G_t \mid S_t = s]\,, \tag{6.3.1}$$

where $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$ is the complete return.

**Formulation 2 (Bellman equation perspective).**

$$V^{\pi}(s) = \mathbb{E}^{\pi}[R_{t+1} + \gamma V^{\pi}(S_{t+1}) \mid S_t = s]\,. \tag{6.3.2}$$

[26] A terminology warning for readers coming from statistics: "bootstrapping" here has *nothing* to do with Efron's resampling bootstrap. In RL it means using one's own current value estimate of a future state as a stand-in for the (unknown) true future value inside the update target—pulling oneself up by one's own bootstraps. Monte Carlo does *not* bootstrap, precisely because it waits for the actual realized return.

The Bellman operator for policy evaluation is

$$B^{\pi}V(s) = r(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s, \pi(s))\, V(s'). \tag{6.3.3}$$

**Monte Carlo** uses Formulation (6.3.1): it waits until the end of an episode to observe $G_t$, then updates

$$V(S_t) \leftarrow V(S_t) + \alpha\big(G_t - V(S_t)\big). \tag{6.3.4}$$

**Temporal-difference** uses Formulation (6.3.2): it immediately uses the observed reward and the current estimate of the next state's value,

$$V(S_t) \leftarrow V(S_t) + \alpha\big(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\big). \tag{6.3.5}$$

TD can be thought of as a **bootstrap method** that updates the current estimate based on an existing estimate of the future.

*Remark* (TD as a stochastic-approximation solver for the Bellman equation). Formulation (6.3.2) is a fixed-point, or conditional-moment, restriction: the true $V^{\pi}$ is the unique function satisfying $V^{\pi}(s) = \mathbb{E}^{\pi}[\, R_{t+1} + \gamma V^{\pi}(S_{t+1}) \mid S_t = s\,]$ for all $s$. Dynamic programming solves this system directly because, knowing $p$ and $r$, it can compute the conditional expectation. Model-free TD cannot evaluate that expectation, so it performs the econometrician's substitution: it replaces the conditional expectation by a *single noisy draw* of the bracketed quantity, $R_{t+1} + \gamma V(S_{t+1})$, and takes a small step toward it. The TD error $\delta_t$ of (6.3.6) is exactly the sample analogue of the residual of this moment condition—its conditional expectation under $\pi$ is zero when $V = V^{\pi}$. Iterating the update is a Robbins–Monro stochastic-approximation scheme that drives the *average* residual to zero, which is why tabular TD(0) converges to $V^{\pi}$ (Sutton, 1988; Tsitsiklis and Van Roy, 1997). (Monte Carlo, by contrast, targets the unconditional moment $\mathbb{E}^{\pi}[G_t \mid S_t = s]$ directly and never exploits the Bellman structure—which is both why it has no bootstrap bias and why it is noisier.)

### 6.3.3 The TD(0) Algorithm

TD(0) is the simplest temporal-difference learning algorithm.

**Definition 6.3.1** (TD target and TD error)**.** The **TD target** is the estimated return using the immediate reward and the current value estimate,

$$R_{t+1} + \gamma V(S_{t+1}),$$

and the **TD error** is the difference between the TD target and the current estimate,

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t). \tag{6.3.6}$$

**Algorithm 6** TD(0) Policy Evaluation

**Require:** policy $\pi$ to be evaluated; step size $\alpha$
**Initialize:** $V$ arbitrarily (e.g., $V(s) = 0$ for all $s$)
**for** each episode **do**
  Initialize state $s$
  **repeat**
    $a \leftarrow$ action given by $\pi$ for $s$
    Take action $a$, observe reward $r$ and next state $s'$
    $V(s) \leftarrow V(s) + \alpha\big[r + \gamma V(s') - V(s)\big]$
    $s \leftarrow s'$
  **until** $s$ is a terminal state
**end for**

The update rule can be written compactly as

$$V(S_t) \leftarrow V(S_t) + \alpha \cdot \delta_t, \tag{6.3.7}$$

where $\delta_t$ is the TD error from (6.3.6).

### 6.3.4 MC vs. TD: A Comprehensive Comparison

The choice between MC and TD methods involves fundamental trade-offs in terms of bias, variance, and applicability. Table 6.3.1 summarizes the structural differences.

**Table 6.3.1.** Structural comparison of Monte Carlo and temporal-difference policy evaluation.

| Aspect | Monte Carlo | Temporal-Difference |
|---|---|---|
| Update timing | Must wait until episode ends | Updates after each step |
| Learning mode | From complete sequences | From incomplete sequences |
| Environment type | Episodic only | Episodic and continuing |
| Target | Actual return $G_t$ | Estimated return $R_{t+1} + \gamma V(S_{t+1})$ |

### 6.3.5 The Bias-Variance Trade-off

The distinction between MC and TD fundamentally reflects a bias-variance trade-off in estimation, summarized in Table 6.3.2.

**Monte Carlo.**

- MC updates using $G_t$, which is an **unbiased estimate** of $v_\pi(S_t)$.
- MC has **high variance, zero bias**.
- The return $G_t$ depends on many random actions, transitions, and rewards throughout the episode.
- **Advantages:**
  - Good convergence properties even with function approximation (e.g., using deep neural networks).
  - Not sensitive to initial value estimates.
  - Simple and intuitive, does not require knowledge of the Bellman equation.
  - More efficient in non-Markov environments.

**Temporal-Difference.**

- TD updates using $R_{t+1} + \gamma V(S_{t+1})$, not $R_{t+1} + \gamma v_\pi(S_{t+1})$, which is a **biased estimate** of $v_\pi(S_t)$ (because $V$ is itself an estimate).
- TD has **much lower variance, some bias**.
- The TD target depends on only one random action, transition, and reward.
- **Advantages:**
  - More data-efficient than MC because it exploits the Markov property and the Bellman equation structure.
  - TD(0) provably converges to the true value function (Sutton, 1988; Tsitsiklis and Van Roy, 1997) (though not always with function approximation).
- **Disadvantages:**
  - More sensitive to initial value estimates.
  - Convergence guarantees are weaker with function approximation (Tsitsiklis and Van Roy, 1997).

*Remark* (A familiar bias–variance trade-off). The MC-versus-TD contrast is the same bias–variance trade-off that governs estimator choice throughout statistics. The MC target $G_t$ behaves like an unbiased but noisy outcome measurement: correct on average, but with variance that compounds over every random action, transition, and

**Table 6.3.2.** Bias-variance characterization of the MC and TD targets.

| **Property** | **Monte Carlo ($G_t$)** | **TD(0) ($R_{t+1} + \gamma V(S_{t+1})$)** |
|---|---|---|
| Bias | Zero (unbiased) | Some (bootstraps on estimate $V$) |
| Variance | High | Much lower |
| Sources of noise | Many actions/transitions/rewards | One action/transition/reward |
| Sensitivity to $V_0$ | Low | Higher |

reward until the episode ends. The TD target $R_{t+1} + \gamma V(S_{t+1})$ behaves like a *plug-in* (or shrinkage) estimator: it substitutes the current model-implied guess $V(S_{t+1})$ for the long random tail of the trajectory, which sharply cuts variance at the cost of a bias that disappears only as $V \to v_\pi$. As in nonparametric estimation, neither extreme is uniformly best; the $n$-step returns of Section 6.3.7 expose the entire frontier between them as a tunable knob.

### 6.3.6 Empirical Comparison: Random Walk Example

The difference between MC and TD is vividly illustrated by the classic random-walk example (Sutton and Barto, 2018). Consider a Markov chain with states $A, B, C, D, E$ arranged linearly, with absorbing terminal states on each end, as depicted in Figure 6.3.1.

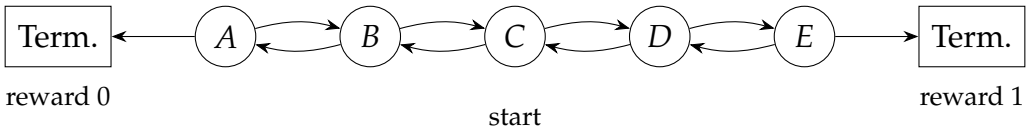


**Figure 6.3.1.** The five-state random walk. The agent starts at $C$ and moves left or right with equal probability; reaching the left terminal gives reward 0, the right terminal gives reward 1.

The agent starts at state $C$ and moves randomly left or right with equal probability. Reaching the left terminal yields reward 0; reaching the right terminal yields reward 1.

Experimental results show that:

- **TD converges faster** than MC across all learning rates.
- **TD achieves lower RMS error** after the same number of episodes.
- The optimal learning rate for TD is larger than for MC, reflecting TD's lower variance.

This empirical advantage of TD stems from its ability to learn incrementally and to leverage the Markov structure of the problem.

### 6.3.7 $n$-Step Prediction: Bridging MC and TD

TD(0) uses a one-step bootstrap target, while MC uses the full-episode return. **$n$-step prediction** creates a continuum between these extremes by looking ahead $n$ steps and then bootstrapping:

$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n}), \tag{6.3.8}$$

and updating

$$V(S_t) \leftarrow V(S_t) + \alpha\big(G_t^{(n)} - V(S_t)\big). \tag{6.3.9}$$

Special cases (episodic setting):

- $n = 1$ recovers **TD(0)**.
- $n = T - t$ (bootstrapping disappears) recovers the **MC return**.

Intuitively, larger $n$ tends to reduce bootstrapping bias but can increase variance; intermediate values of $n$ can often work well in practice.

## 6.4 Model-Free Control (Learning an Optimal Policy)

So far we focused on **prediction**: estimating $v_\pi$ (or $q_\pi$) for a fixed policy $\pi$. In **control**, the objective is to learn an *optimal* policy $\pi^*$ while interacting with an unknown environment.

### 6.4.1 The Control Objective and Why We Learn $Q(s, a)$

For control, the most convenient object is the **action-value function**

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s,\, A_t = a]\,. \tag{6.4.1}$$

If we know $q_\pi$, we can improve the policy by acting greedily:

$$\pi'(s) \in \arg\max_a q_\pi(s, a). \tag{6.4.2}$$

*Remark* (Why model-free control learns $Q$ rather than $V$). This is the crucial reason action-values dominate model-free control. With a *known* model one can improve a policy starting from a state-value function via the one-step lookahead $\pi'(s) \in \arg\max_a \big[\, r(s,a) + \gamma \sum_{s'} p(s' \mid s, a)\, V(s') \,\big]$—but this requires the very transition probabilities $p$ and rewards $r$ that model-free RL does not possess. The action-value function sidesteps the difficulty by *baking the one-step lookahead into the function itself*: greedy improvement becomes the purely model-free operation

$\pi'(s) \in \arg\max_a Q(s,a)$, a table lookup and a maximization over the (finite) action set that needs no model at all. This is why every model-free control algorithm below estimates $Q$ rather than $V$.

Model-free control algorithms implement a form of **generalized policy iteration (GPI)**:

- (approximately) evaluate the current policy by estimating $q_\pi$ from data;
- improve the policy using the current estimate $Q \approx q_\pi$;
- repeat until the policy stabilizes.

*Remark* (GPI generalizes policy iteration from dynamic programming). Generalized policy iteration is the model-free descendant of the *policy iteration* algorithm for known MDPs (the dynamic-programming material of the previous chapter): one alternates a policy-*evaluation* step (make $Q$ consistent with the current policy) with a policy-*improvement* step (make the policy greedy with respect to $Q$). Classical policy iteration runs each step to completion using the known model $p, r$; GPI relaxes both—it evaluates only *approximately* (from finitely many sampled returns or TD updates), and it may improve the policy after every episode, or even after every single step. The striking fact, which underlies essentially all of model-free control, is that these two interleaved and never-fully-completed processes still converge toward the same joint fixed point, the optimal pair $(Q^*, \pi^*)$.

### 6.4.2 MC Control and the Exploration Problem

MC control uses Monte Carlo returns to estimate $q_\pi(s,a)$ and improves the policy using $\varepsilon$-greedy exploration. The key obstacle is that under a deterministic policy, many state-action pairs may **never be visited**, making their values unidentifiable from data.

**$\varepsilon$-greedy exploration** addresses this by ensuring every action has a nonzero chance of being taken:

$$\pi(a \mid s) = \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}|, & \text{if } a \in \arg\max_{a'} Q(s,a'), \\ \varepsilon/|\mathcal{A}|, & \text{otherwise.} \end{cases} \tag{6.4.3}$$

*Remark* (Exploration is the bandit problem in disguise). The exploration–exploitation dilemma here is precisely the one studied in **multi-armed bandits** and, more broadly, in sequential experimentation and adaptive A/B testing: to learn whether an untried action is good, one must occasionally try it, paying a short-run reward cost for long-run information. $\varepsilon$-greedy is the crudest such rule—play the current best estimate with probability $1 - \varepsilon$ and "run an experiment" (sample uniformly) with probability

$\varepsilon$—which guarantees, exactly like a randomized trial, that every action keeps a positive sampling probability so its value remains identified (Lai and Robbins, 1985; Auer et al., 2002). RL control is harder than a pure bandit only because an action's payoff is delayed and routed through future states, so exploration must eventually cover state–action *pairs*, not merely actions in a single state.

*Remark* (Improvement with approximate $Q$). With *exact* $q_\pi$, greedy (or $\varepsilon$-greedy) improvement guarantees non-decreasing value. With **approximate** $Q$ from finite samples (as in model-free methods), monotonic improvement is not guaranteed at every iteration, yet the control loop often works well empirically.

### 6.4.3 GLIE (Greedy in the Limit with Infinite Exploration)

To reason about convergence in tabular settings, a common condition is **GLIE**:

1. every state-action pair is explored infinitely often, and
2. the policy becomes greedy in the limit (exploration decays to zero).

For $\varepsilon$-greedy policies, a typical GLIE schedule is $\varepsilon_k = 1/k$ on episode $k$.

*Remark* (GLIE and the Robbins–Monro step-size conditions). GLIE is the control-side analogue of the consistency conditions for a stochastic-approximation estimator. The "infinite exploration" clause ensures every $(s, a)$ is sampled infinitely often—without it, some $Q(s, a)$ could never be identified, just as a regressor with no variation cannot be estimated. The "greedy in the limit" clause ensures we ultimately evaluate the policy we actually care about (the greedy one) rather than a permanently randomized one. Convergence of tabular MC control and SARSA to $q^*$ then requires, in addition, the classical Robbins–Monro step-size conditions $\sum_k \alpha_k = \infty$ and $\sum_k \alpha_k^2 < \infty$: the first lets the iterate travel arbitrarily far to reach the truth, the second forces the steps to shrink fast enough to average out sampling noise. The harmonic schedules $\alpha_k = 1/N(s, a)$ and $\varepsilon_k = 1/k$ used here satisfy these requirements.

In practice, MC control is often demonstrated on Blackjack: by repeatedly simulating games and improving an $\varepsilon$-greedy policy, we can learn both an interpretable policy (e.g., stick/hit regions) and the corresponding action-value surfaces.

### 6.4.4 TD Control: Learning $Q$ Online

TD methods are often more **sample-efficient** than MC because they bootstrap and can learn online. For control, we apply TD updates to the action-value function and combine them with $\varepsilon$-greedy improvement. This leads to two canonical algorithms:

- **SARSA** (on-policy TD control) (Rummery and Niranjan, 1994), and
- **Q-learning** (off-policy TD control) (Watkins and Dayan, 1992).

**Algorithm 7** GLIE Monte Carlo Control (sketch)

**Initialize:** $Q(s,a)$ arbitrarily; $N(s,a) \leftarrow 0$ for all $(s,a)$
**for** episode $k = 1, 2, 3, \ldots$ **do**
  $\varepsilon \leftarrow 1/k$
  Generate an episode using the $\varepsilon$-greedy policy w.r.t. $Q$
  **for all** $(s,a)$ visited in the episode (first-visit or every-visit) **do**
    $G \leftarrow$ return following that $(s,a)$
    $N(s,a) \leftarrow N(s,a) + 1$
    $Q(s,a) \leftarrow Q(s,a) + \dfrac{1}{N(s,a)}\big(G - Q(s,a)\big)$
  **end for**
**end for**

### 6.4.5 SARSA (On-Policy TD Control)

SARSA is named after the data tuple it uses: **State-Action-Reward-State-Action**. It evaluates and improves the *same* (typically $\varepsilon$-greedy) policy it follows (Rummery and Niranjan, 1994).

**SARSA update.**

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\Big). \tag{6.4.4}$$

**Algorithm 8** SARSA (tabular, on-policy TD control)

Initialize $Q(s,a)$ arbitrarily
**for** each episode **do**
  Initialize $S$; choose $A$ using $\varepsilon$-greedy$(Q)$
  **repeat**
    Take action $A$, observe reward $R$ and next state $S'$
    Choose $A'$ using $\varepsilon$-greedy$(Q)$ at $S'$
    $Q(S,A) \leftarrow Q(S,A) + \alpha\,[\,R + \gamma Q(S',A') - Q(S,A)\,]$
    $S \leftarrow S'$; $\quad A \leftarrow A'$
  **until** $S$ is terminal
**end for**

In the tabular setting, SARSA converges to the optimal action-value function under standard step-size conditions and a GLIE exploration schedule.

### 6.4.6 $n$-Step SARSA and SARSA($\lambda$)

As with prediction, we can reduce variance and speed learning by looking ahead multiple steps before bootstrapping. The **$n$-step SARSA** return is

$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n Q(S_{t+n}, A_{t+n}), \tag{6.4.5}$$

leading to the update

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\big(G_t^{(n)} - Q(S_t, A_t)\big). \tag{6.4.6}$$

**SARSA($\lambda$)** mixes $n$-step returns with exponentially decaying weights. In the forward-view form (episodic),

$$G_t^{(\lambda)} = (1-\lambda)\sum_{n=1}^{T-t-1} \lambda^{n-1} G_t^{(n)} + \lambda^{T-t-1} G_t, \tag{6.4.7}$$

where $G_t$ is the full MC return. An equivalent and computationally efficient **backward view** uses eligibility traces to distribute TD errors over recently visited state-action pairs (Singh and Sutton, 1996).

*Remark* (The $\lambda$-return is an EWMA over horizons). The forward-view weights $(1-\lambda)\lambda^{n-1}$ in (6.4.7) form an **exponentially weighted average over all $n$-step returns**: they are nonnegative and sum to one (the leftover geometric mass $\lambda^{T-t-1}$ is assigned to the full MC return $G_t$), with $\lambda$ controlling how quickly longer-horizon targets are discounted. So $\lambda$ is a single dial spanning the same bias–variance continuum as the integer $n$: $\lambda = 0$ collapses to the one-step TD target (lowest variance, most bootstrap bias), and $\lambda = 1$ recovers the MC return (unbiased, highest variance). The backward-view eligibility-trace implementation (Singh and Sutton, 1996) delivers the *same* expected updates online—without waiting for the episode to end—by keeping, for each state–action pair, a scalar "eligibility" that records how recently it was visited and decays geometrically at rate $\gamma\lambda$, then scaling each pair's update by its current eligibility.

### 6.4.7 Off-Policy Learning and Q-Learning

In **off-policy** learning, data is generated by a **behavior policy** $\mu(a \mid s)$, while we evaluate or improve a different **target policy** $\pi(a \mid s)$. This matters when we want to:

- learn from logs, demonstrations, or other agents;
- reuse experience generated by older policies;
- learn about an optimal (greedy) policy while behaving exploratorily;
- or learn about multiple target policies from one data stream.

**Q-learning** is an off-policy TD control algorithm that learns the optimal $Q^*$ while typically behaving $\varepsilon$-greedily (Watkins and Dayan, 1992).

**Q-learning update.**

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big). \tag{6.4.8}$$

*Remark* (Off-policy learning as counterfactual evaluation, and why the max matters). "Off-policy" should resonate with anyone who has estimated the effect of one policy from data generated under a *different* policy: learning from logs, demonstrations, or a legacy system is the RL version of counterfactual or observational policy evaluation, and in general it requires reweighting (importance sampling) to correct for the mismatch between the behavior policy $\mu$ and the target policy $\pi$. Q-learning enjoys a fortunate special case in which *no* importance weights are needed: its bootstrap target $R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$ refers to the greedy target policy directly and does *not* depend on which action the behavior policy actually took at $S_{t+1}$. Contrast SARSA's target $R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$, which plugs in the action $A_{t+1}$ genuinely drawn from the behavior policy—making SARSA *on-policy*, able to evaluate only the (exploratory) policy it is actually running. That single substitution, $\max_a Q(S_{t+1}, a)$ in place of $Q(S_{t+1}, A_{t+1})$, is the entire difference between learning the optimal $Q^*$ off-policy and learning $q_\pi$ on-policy.

**Algorithm 9** Q-learning (tabular, off-policy TD control)

Initialize $Q(s, a)$ arbitrarily
**for** each episode **do**
  Initialize $S$
  **repeat**
    Choose $A$ using the behavior policy (e.g., $\varepsilon$-greedy w.r.t. $Q$)
    Take action $A$, observe reward $R$ and next state $S'$
    $Q(S, A) \leftarrow Q(S, A) + \alpha\,[\,R + \gamma \max_a Q(S', a) - Q(S, A)\,]$
    $S \leftarrow S'$
  **until** $S$ is terminal
**end for**

### 6.4.8 SARSA vs. Q-Learning: The Cliff-Walking Intuition

The difference between SARSA and Q-learning is the **target** they bootstrap toward:

- SARSA uses $Q(S_{t+1}, A_{t+1})$, where $A_{t+1}$ is drawn from the ($\varepsilon$-greedy) behavior policy. It therefore learns the value of the *actually executed* exploratory behavior.
- Q-learning uses $\max_a Q(S_{t+1}, a)$, which corresponds to the greedy target policy, even if the behavior is exploratory.

In the classic **cliff-walking** example (Sutton and Barto, 2018) (a gridworld with a large negative reward for stepping off the cliff), $\varepsilon$-greedy SARSA tends to learn a **safer**

**path** away from the cliff because it accounts for occasional exploratory moves. Q-learning tends to learn the **shortest path** along the cliff edge (optimal under a purely greedy policy) but can suffer more exploratory failures when $\varepsilon$ remains nonzero.

## 6.5 Conclusion

Model-free reinforcement learning can be viewed as two intertwined problems:

- **Prediction:** evaluate a policy (MC, TD, and $n$-step methods).
- **Control:** improve the policy toward optimality (MC control, SARSA, Q-learning, and their multi-step/trace variants).

Across both, the same design tensions recur: bias vs. variance (bootstrapping vs. full returns), online learning vs. episodic averaging, and exploration vs. exploitation ($\varepsilon$-greedy and GLIE schedules). For business and social-science researchers, these algorithms provide principled tools for sequential decision problems where accurate environment models are unavailable, ranging from dynamic pricing and inventory control to recommendations, experimentation, and adaptive interventions.

# Chapter 7: Deep Reinforcement Learning

This chapter introduces deep reinforcement learning (Deep RL), the combination of deep neural networks with reinforcement learning algorithms that has enabled agents to tackle problems with enormous state and action spaces. We begin by reviewing where we stand in the RL landscape, then build up value function approximation as the bridge from tabular RL to scalable methods. We cover Deep Q-Networks (DQN), the landmark algorithm that achieved human-level performance on Atari games, together with its extensions and business applications. We then turn to policy-based methods, which optimize the policy directly rather than deriving it from a value function, covering policy gradient theory, the REINFORCE algorithm, variance reduction techniques, and actor-critic methods including A3C. We close this part with an operations-management application of Deep RL to inventory control.

## 7.1 Where Are We?

Before proceeding to deep reinforcement learning, it is helpful to situate ourselves within the broader RL curriculum (see Sutton and Barto (2018) for a comprehensive textbook treatment, and Silver's lecture series[27] for an excellent video course).

[27] https://www.davidsilver.uk/teaching/

**Model-Based Methods.**

- We know the action space, the state space, the transition probability matrix, and the reward function.
- The Bellman equation is used to evaluate a policy, and the Bellman optimality operator is used to find the optimal policy.
- Solution algorithms include value iteration, policy iteration, and linear programming.

**Model-Free Methods.**

- Monte Carlo (MC) learns directly from episodes of experience: sample the entire reward process and use the empirical mean return to approximate the expected return.
- Temporal Difference (TD) combines MC and the Bellman operator: bootstrap to update the value function based on the existing estimate.
- MC has zero bias and high variance; TD has some bias and low variance.

**Model-Free Control.**

- MC policy iteration combines MC policy evaluation with $\varepsilon$-greedy policy improvement.
- SARSA is an on-policy method that updates $Q(s, a)$ using TD and $\varepsilon$-greedy policy improvement.
- Q-learning is an off-policy method that learns $Q(s, a)$ for the greedy target policy while using an $\varepsilon$-greedy behavior policy through the TD update.

All of the above methods, however, rely on **tabular representations**, maintaining a separate value for every state or state-action pair. This works well for small, discrete problems but becomes infeasible when the state or action space is large or continuous. The central question of this chapter is: **how do we scale reinforcement learning to real-world problems?**

## 7.2 Value Function Approximation

### 7.2.1 The Limits of Tabular RL

In tabular RL, we represent the value functions by a lookup table:

- Every state $s$ has an entry $V(s)$.
- Every state-action pair $(s, a)$ has an entry $Q(s, a)$.

This is sometimes visualized as a **Q-table**, with states along one axis and actions along the other. Each cell stores the estimated value of taking that action in that state.

**Challenges with large MDPs.**

- **Too many states or actions to store in memory.** For example, the game of Go has approximately $3^{19\times19} = 3^{361} \approx 10^{172}$ states,[28] far exceeding the estimated number of atoms in the observable universe ($\sim 10^{80}$). Self-driving cars operate in continuous state spaces. Dynamic pricing involves continuous action spaces.
- **Too slow to learn the value of each state individually.** Even if we could store the table, visiting every state often enough to obtain reliable estimates would take prohibitively long.

### 7.2.2 The Idea of Function Approximation

The solution is to **avoid explicitly learning or storing values for every single state**. Instead, we estimate approximate value and policy functions parameterized by a weight vector $\mathbf{w}$:

$$\hat{v}(s, \mathbf{w}) \approx v_\pi(s), \tag{7.2.1}$$

$$\hat{q}(s, a, \mathbf{w}) \approx q_\pi(s, a), \tag{7.2.2}$$

$$\hat{\pi}(a, s, \mathbf{w}) \approx \pi(a \mid s). \tag{7.2.3}$$

The key benefits of function approximation are:

1. **Generalization from seen to unseen states.** By learning a parameterized function, we can estimate values for states that have never been visited, as long as they share features with states that have been visited.
2. **Compact representation.** Instead of storing $|\mathcal{S}|$ (or $|\mathcal{S}| \times |\mathcal{A}|$) values, we store only the parameter vector $\mathbf{w}$, whose dimension can be vastly smaller.
3. **Scalability.** The parameter $\mathbf{w}$ is updated using MC or TD learning, making the approach compatible with the model-free methods we have already studied.

[28] Each of the $19 \times 19 = 361$ board points is empty, black, or white, giving $3^{361} \approx 1.7 \times 10^{172}$ raw configurations; the number of *legal* positions is somewhat smaller, about $2.1 \times 10^{170}$. Either way the count dwarfs anything that could ever be stored in a lookup table.

*Remark* (Function approximation as parametric regression). A reader without a machine-learning background can read this whole section through the lens of regression. Tabular RL is like estimating a separate "cell mean" for every state—a fully saturated, nonparametric specification with one free parameter per cell. Function approximation instead posits a *parametric* model $\hat{v}(s, \mathbf{w})$ and estimates the low-dimensional parameter $\mathbf{w}$, exactly as one replaces a saturated dummy-variable specification by a parsimonious regression on covariates $\mathbf{x}(s)$. Generalization to unseen states is then nothing more than out-of-sample prediction: states with similar features receive similar predicted values. The price, as in any parametric model, is approximation (specification) error if the chosen functional form cannot represent the true $v^\pi$.

There are several function design choices. The approximator can take:

- state $s$ as input and output $\hat{v}(s, \mathbf{w})$;
- state $s$ and action $a$ as inputs and output $\hat{q}(s, a, \mathbf{w})$;
- state $s$ as input and output $\hat{q}(s, a_1, \mathbf{w}), \hat{q}(s, a_2, \mathbf{w}), \ldots, \hat{q}(s, a_m, \mathbf{w})$ for all actions simultaneously.

We consider **differentiable functions** as the approximators, since they can deal with non-stationary and non-i.i.d. data, both hallmarks of RL. Two important families are:

- **Linear feature representations:** $\hat{v}(s, \mathbf{w}) = \mathbf{x}(s)^\top \mathbf{w}$;
- **Neural networks:** $\hat{v}(s, \mathbf{w}) = f_{\mathbf{w}}(s)$, where $f$ is a deep neural network.

### 7.2.3 Value Function Approximation with an Oracle

To build intuition, we first consider the idealized setting where an **oracle** provides the true value $v^\pi(s)$ for any given state $s$. The objective is to find the best approximate representation of $v^\pi(s)$.

**Loss function.** We use the mean squared error (MSE) between the true value and the approximation:

$$J(\mathbf{w}) = \mathbb{E}_\pi \left[ \left( v^\pi(s) - \hat{v}(s, \mathbf{w}) \right)^2 \right]. \tag{7.2.4}$$

**Gradient descent update.** We minimize this loss using gradient descent:

$$\Delta \mathbf{w} = -\tfrac{1}{2} \alpha \nabla_{\mathbf{w}} J(\mathbf{w}), \qquad \mathbf{w}_{t+1} = \mathbf{w}_t + \Delta \mathbf{w}, \tag{7.2.5}$$

where $\alpha$ is the learning rate (step size).

### 7.2.4 Linear Approximation with an Oracle

A natural starting point is to represent states using a finite feature vector with $n$ variables:

$$\mathbf{x}(s) = \begin{pmatrix} x_1(s) \\ x_2(s) \\ \vdots \\ x_n(s) \end{pmatrix}. \tag{7.2.6}$$

The value function is then approximated by a linear combination of features:

$$\hat{V}(s;\mathbf{w}) = \sum_{j=1}^{n} x_j(s) w_j = \mathbf{x}(s)^\top \mathbf{w}. \tag{7.2.7}$$

The loss function is

$$J(\mathbf{w}) = \mathbb{E}_\pi \left[ \left( V^\pi(s) - \hat{V}(s;\mathbf{w}) \right)^2 \right]. \tag{7.2.8}$$

Using stochastic gradient descent (SGD) to update the weights, $\Delta\mathbf{w} = -\frac{1}{2}\alpha \nabla_{\mathbf{w}} J(\mathbf{w})$. Since the function is linear and the loss is MSE, the gradient simplifies to

$$\Delta\mathbf{w} = \alpha \left( v^\pi(s) - \hat{v}(s,\mathbf{w}) \right) \mathbf{x}(s). \tag{7.2.9}$$

**Where the update comes from.** The simplification is just the chain rule applied to a squared loss, identical to deriving the OLS normal-equation gradient. With $\hat{v}(s,\mathbf{w}) = \mathbf{x}(s)^\top \mathbf{w}$ we have $\nabla_{\mathbf{w}} \hat{v}(s,\mathbf{w}) = \mathbf{x}(s)$, so

$$-\tfrac{1}{2} \nabla_{\mathbf{w}} \big( v^\pi(s) - \hat{v}(s,\mathbf{w}) \big)^2 = -\tfrac{1}{2} \cdot 2 \big( v^\pi(s) - \hat{v}(s,\mathbf{w}) \big) \cdot \big( -\mathbf{x}(s) \big) = \big( v^\pi(s) - \hat{v}(s,\mathbf{w}) \big) \mathbf{x}(s).$$

The factor $\frac{1}{2}$ in the loss is included precisely to cancel the 2 from differentiating the square. The update thus moves $\mathbf{w}$ in the direction of the feature vector $\mathbf{x}(s)$, scaled by the prediction error $v^\pi(s) - \hat{v}(s,\mathbf{w})$—the same "error times regressor" form as a single SGD step on a linear regression.

*Remark* (Convergence under linearity). Because the loss is convex in $\mathbf{w}$, SGD will converge to the **global minimum**.

For a general (nonlinear) approximation function, SGD will only converge to a local minimum. The general update rule is

$$\Delta\mathbf{w} = \alpha \left( v^\pi(s) - \hat{v}(s,\mathbf{w}) \right) \nabla_{\mathbf{w}} \hat{v}(s_t,\mathbf{w}). \tag{7.2.10}$$

### 7.2.5 Model-Free Evaluation with Approximation

In practice, we do not have oracles, we only have **rewards** collected by **interacting with the environment**. But we can obtain the **target** using samples.

**Monte Carlo with function approximation.** For MC, the target is simply the actual return $G_t$:

$$\Delta \mathbf{w} = \alpha \left( G_t - \hat{v}(s_t, \mathbf{w}) \right) \nabla_{\mathbf{w}} \hat{v}(s_t, \mathbf{w}). \tag{7.2.11}$$

Key properties:

- $G_t$ is an **unbiased estimate** of the oracle $v^\pi(s)$.
- MC prediction converges, in both linear and nonlinear value function approximation.

**TD(0) with function approximation.** For TD(0), the target is the TD target $R_{t+1} + \gamma \hat{v}(s_{t+1}, \mathbf{w})$:

$$\Delta \mathbf{w} = \alpha \left( R_{t+1} + \gamma \hat{v}(s_{t+1}, \mathbf{w}) - \hat{v}(s_t, \mathbf{w}) \right) \nabla_{\mathbf{w}} \hat{v}(s_t, \mathbf{w}). \tag{7.2.12}$$

Key properties:

- The TD target $R_{t+1} + \gamma \hat{v}(s_{t+1}, \mathbf{w})$ is a **biased estimate** of the oracle $v^\pi(s)$.
- This is called a **semi-gradient** method, because we ignore the effect of changing $\mathbf{w}$ on the target (i.e., we do not differentiate through $\hat{v}(s_{t+1}, \mathbf{w})$ in the target).
- Linear TD(0) converges close to the global minimum.

*Remark* (Why "semi"-gradient, and the link to stochastic approximation). Two ideas here are worth slowing down on. First, *semi-gradient*: the TD target $R_{t+1} + \gamma \hat{v}(s_{t+1}, \mathbf{w})$ itself depends on $\mathbf{w}$, but we deliberately treat it as a fixed regression target and do *not* differentiate through it. The update is therefore not the exact gradient of any single loss function—hence "semi"—which is why TD is biased and, off-policy, can even diverge (the deadly triad of §7.3.4). Second, these incremental updates are instances of the **Robbins–Monro stochastic-approximation** scheme familiar from econometrics: to solve an equation of the form $\mathbb{E}[\,\text{error}\,] = 0$ using only noisy samples, take small steps $\mathbf{w} \leftarrow \mathbf{w} + \alpha_t \,(\text{noisy error})\, \nabla_{\mathbf{w}} \hat{v}$ with step sizes satisfying $\sum_t \alpha_t = \infty$ and $\sum_t \alpha_t^2 < \infty$. Monte Carlo uses the unbiased return $G_t$ (zero bias, high variance); TD "bootstraps" by plugging in its own current estimate of the continuation value (some bias, lower variance), trading bias against variance just as in any estimator.

**Algorithms: MC + SGD vs. TD + semi-gradient SGD.** The MC algorithm samples a complete trajectory $\tau$ before computing the gradient, whereas the TD algorithm updates after each single transition. The two procedures are given in Algorithms 10 and 11.

**Algorithm 10** MC value-function evaluation with SGD

**while** not converged **do**
  Sample $s_0 \sim p_0$; set $t = 0$
  **while** $s_t \neq$ terminal **do**
    $a_t \sim \pi(\cdot \mid s_t)$
    $(r_t, s_{t+1}) \sim p(\cdot, \cdot \mid s_t, a_t)$
    $t \leftarrow t + 1$
  **end while**
  $T \leftarrow t$
  $g \leftarrow \left(V_\phi(s_0) - \sum_{t=0}^{T-1} \gamma^t r_t\right) \nabla_\phi V_\phi(s_0)$
  Update $\phi$ using $g$ with an optimizer
**end while**

**Algorithm 11** TD(0) value-function evaluation with semi-gradient SGD

**while** not converged **do**
  Sample $s_0 \sim p_0$, $a_0 \sim \pi(\cdot \mid s_0)$, $(r_0, s_1) \sim p(\cdot, \cdot \mid s_0, a_0)$
  **if** $s_1 \neq$ terminal **then**
    $y \leftarrow r_0 + \gamma V_\phi(s_1)$
  **else**
    $y \leftarrow r_0$
  **end if**
  $g \leftarrow \left(V_\phi(s_0) - y\right) \nabla_\phi V_\phi(s_0)$
  Update $\phi$ using $g$ with an optimizer
**end while**

The fundamental difference is clear: MC requires completing entire episodes before updating, while TD updates after a single step, making it suitable for continuing (non-episodic) tasks and faster learning in practice.

## 7.3 Generalized Policy Iteration with Value Function Approximation

### 7.3.1 The Framework

Just as in tabular RL, we can combine approximate policy evaluation with policy improvement to perform **generalized policy iteration (GPI)** with function approximation. The procedure alternates between two steps:

1. **Policy evaluation:** approximate policy evaluation, $\hat{q}(\cdot, \cdot, \mathbf{w}) \approx q^{\pi}$.
2. **Policy improvement:** $\varepsilon$-greedy policy improvement based on the approximate Q-function.

Starting from an initial weight vector $\mathbf{w}$, the process iteratively refines the Q-function approximation and improves the policy, converging toward the optimal Q-function $q_*$.

### 7.3.2 Model-Free Control with Linear Approximation

With a similar idea to value function approximation, we represent state-action pairs using a finite feature vector:

$$\mathbf{x}(s,a) = \begin{pmatrix} x_1(s,a) \\ x_2(s,a) \\ \vdots \\ x_n(s,a) \end{pmatrix}. \tag{7.3.1}$$

The Q-function is approximated by a linear combination of features:

$$\hat{Q}(s,a;\mathbf{w}) = \mathbf{x}(s,a)^{\top}\mathbf{w} = \sum_{j=1}^{n} x_j(s,a) w_j. \tag{7.3.2}$$

The objective is to minimize the MSE between the approximate Q-function and the oracle Q-function:

$$J(\mathbf{w}) = \mathbb{E}_{\pi}\left[\left(q^{\pi}(s,a) - \hat{q}(s,a,\mathbf{w})\right)^2\right]. \tag{7.3.3}$$

The stochastic gradient descent update is

$$\Delta\mathbf{w} = \alpha\left(q^{\pi}(s,a) - \hat{q}(s,a,\mathbf{w})\right)\mathbf{x}(s,a). \tag{7.3.4}$$

### 7.3.3 Incremental Control Algorithms

In practice, we do not have the oracle state-action function $q^{\pi}(s,a)$, we only have rewards collected by interacting with the environment. Different choices of target lead to different algorithms.

**MC control with approximation.** For MC, the target is the actual return $G_t$:

$$\Delta\mathbf{w} = \alpha\left(G_t - \hat{q}(s_t,a_t,\mathbf{w})\right)\nabla_{\mathbf{w}}\hat{q}(s_t,a_t,\mathbf{w}). \tag{7.3.5}$$

**SARSA with approximation.** For SARSA (Rummery and Niranjan, 1994), the target is the TD target $R_{t+1} + \gamma \hat{q}(s_{t+1}, a_{t+1}, \mathbf{w})$:

$$\Delta \mathbf{w} = \alpha \left( R_{t+1} + \gamma \hat{q}(s_{t+1}, a_{t+1}, \mathbf{w}) - \hat{q}(s_t, a_t, \mathbf{w}) \right) \nabla_{\mathbf{w}} \hat{q}(s_t, a_t, \mathbf{w}). \tag{7.3.6}$$

**Q-learning with approximation.** For Q-learning (Watkins and Dayan, 1992), the target is $R_{t+1} + \gamma \max_a \hat{q}(s_{t+1}, a, \mathbf{w})$:

$$\Delta \mathbf{w} = \alpha \left( R_{t+1} + \gamma \max_a \hat{q}(s_{t+1}, a, \mathbf{w}) - \hat{q}(s_t, a_t, \mathbf{w}) \right) \nabla_{\mathbf{w}} \hat{q}(s_t, a_t, \mathbf{w}). \tag{7.3.7}$$

### 7.3.4 Convergence of Control Algorithms

An important practical concern is whether these approximate control algorithms converge. The key insight is that **TD with value function approximation does NOT follow the gradient of any loss function.** The updates essentially involve an *approximate Bellman backup* combined with *fitting the underlying value function*, two operations that do not jointly minimize a single objective.

TD may diverge in **off-policy** (e.g., Q-learning) or **nonlinear approximation** settings (Tsitsiklis and Van Roy, 1997; Bertsekas and Tsitsiklis, 1996). This is the challenge for off-policy control: the behavior policy and the target policy are not identical, so the value function approximation may diverge.

Table 7.3.1 summarizes the convergence guarantees.

**Table 7.3.1.** Convergence of control algorithms under different value-function representations. The symbol (✓) indicates that the algorithm moves around the near-optimal value function (it oscillates near the optimum but does not converge exactly).

| Algorithm | Table Lookup | Linear | Non-Linear |
|---|---|---|---|
| Monte-Carlo Control | ✓ | (✓) | × |
| SARSA | ✓ | (✓) | × |
| Q-Learning | ✓ | × | × |

*Remark* (The Deadly Triad). The combination of (1) **bootstrapping**, (2) **function approximation**, and (3) **off-policy learning** is known as the "deadly triad" in RL. When all three are present simultaneously, training instability and divergence become significant risks. Q-learning with nonlinear function approximation is a prime example, as it combines all three elements.

## 7.4 Going Beyond Linear: Deep Reinforcement Learning

### 7.4.1 Motivation

So far, we have focused on linear approximators. Linear approximations work well given the right set of features, but they require **manual design** of the feature set, a process that demands significant domain expertise and may not capture complex non-linear relationships in the data.

We know that deep neural networks (DNNs) are much better representation tools when we have a large data set. They can automatically learn hierarchical feature representations from raw data, eliminating the need for hand-crafted features.

The central question is: **how can we leverage deep learning for function approximation and model-free control of MDPs?**

### 7.4.2 Deep Reinforcement Learning

Deep reinforcement learning leverages DNNs to represent:

- **value functions** ($Q$ and $V$);
- **policy functions** ($\pi$);
- **world models** (for model-based approaches).

The loss function is optimized by stochastic gradient descent (SGD), just as in supervised deep learning.

**Core challenges.**

- **Data inefficiency:** too many model parameters to optimize, requiring vast amounts of interaction data.
- **The Deadly Triad creates instability and divergence in training**, arising from:
  - nonlinear function approximation,
  - bootstrapping,
  - off-policy training.

Deep Q-learning (DQN) addresses two critical issues, **correlations between samples** and **non-stationary targets**, through two key innovations:

1. **Experience replay:** sample experience randomly from stored data to update weights, reducing correlations between data points.
2. **Fixed Q-targets:** fix the target network's weights while updating the main network, improving stability.

## 7.5 Deep Q-Network (DQN) for Atari

### 7.5.1 The Breakthrough

The landmark paper by Mnih et al. (2015), "Human-level control through deep reinforcement learning," published in *Nature*, demonstrated that a single DQN agent could achieve **professional human-level performance** across many Atari 2600 games using the **same network architecture and hyperparameters** for all games. This was a groundbreaking result: the same algorithm, without any game-specific tuning, could learn to play dozens of different video games from raw pixel inputs.

DQN represents the action-value function with a deep neural network (DNN) approximator. The performance comparison between linear approximators and deep networks across several Atari games (Table 7.5.1) illustrates the dramatic advantage of deep representations.

**Table 7.5.1.** Early performance comparison between linear approximators and deep networks across selected Atari games. Scores represent the percentage of human performance; deep networks eventually far surpass these early results with full DQN training.

| Game | Linear | Deep Network |
|---|---|---|
| Breakout | 3 | 3 |
| Enduro | 62 | 29 |
| River Raid | 2345 | 1453 |
| Seaquest | 656 | 275 |
| Space Invaders | 301 | 302 |

### 7.5.2 Architecture and Setup

The DQN architecture for Atari is an **end-to-end learning** system for $Q(s, a)$ directly from input pixel frames:

- **Input state** $s$: a stack of raw pixels from the latest 4 frames (to capture motion information). The input is preprocessed to $84 \times 84 \times 4$.
- **Output** of $Q(s, a)$: 18 values, one for each possible joystick action.
- **Reward**: the change in game score for that step.
- **Network architecture**: a convolutional neural network (CNN)[29] with

[29] A CNN is a neural network specialized for grid-structured inputs such as images (introduced in Chapter 4). A *filter* (or kernel) is a small weight matrix slid across the image; at each position it computes a weighted sum of the local pixels—much like a local regression or a moving-average smoother applied

  - $8 \times 8 \times 4$ filters with stride $4 \to 20 \times 20 \times 16$ feature maps;
  - $4 \times 4 \times 16$ filters with stride $2 \to 9 \times 9 \times 32$ feature maps;
  - a fully connected layer $\to$ 256 hidden units;
  - a fully connected output layer $\to$ 4-18 action values.

- **Network architecture and hyperparameters are fixed across all games.**

The architecture maps a stack of preprocessed frames through two convolutional layers and two fully connected layers to a vector of action values, as sketched in Figure 7.5.1.[30]

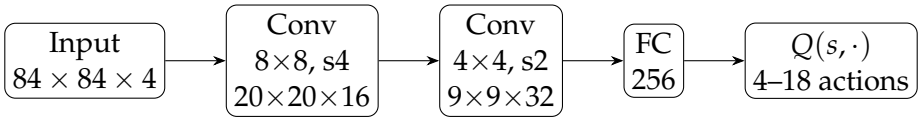


**Figure 7.5.1.** The DQN convolutional architecture for Atari: raw pixel frames are mapped end-to-end to action values $Q(s, a)$ for all actions at once.

The crucial design choice is to output Q-values for **all actions simultaneously** given a state, rather than taking both state and action as input. This allows efficient computation of $\max_a Q(s, a)$ in a single forward pass.

### 7.5.3 Experience Replay

A fundamental problem in training neural networks with RL data is that consecutive samples are **highly correlated**, the agent's trajectory through the environment produces a sequence of states where each state is very similar to the previous one. Standard SGD assumes i.i.d. samples, so training on correlated data leads to poor learning and instability.[31]

to a small patch. The *stride* is how many pixels the filter shifts between evaluations (stride 4 subsamples the image, shrinking its spatial size). Each filter produces one *feature map*, the 2D array of its responses across the image, so 16 filters yield 16 feature maps. Reusing one small filter at every position makes CNNs far more parameter-efficient than fully connected layers and builds in translation invariance.

[30] This compact two-convolution network (16 then 32 filters, a 256-unit hidden layer) is the architecture of the original 2013 workshop paper, *Playing Atari with Deep Reinforcement Learning*. The 2015 *Nature* version that reached professional human-level play on 49 games used a slightly larger network with three convolutional layers (32, 64, 64 filters) and a 512-unit hidden layer. The two are routinely shown interchangeably in lectures; the qualitative lessons (end-to-end learning of $Q$ from pixels, experience replay, fixed targets) are identical.

[31] The concern mirrors classical estimation: just as serially correlated errors violate the independence assumptions behind OLS inference and can mislead adaptive procedures, highly correlated gradient samples make SGD's noisy steps point the same way for long stretches, so the network overfits whatever small region of the state space it is currently visiting. Drawing random minibatches from a large replay buffer decorrelates the samples—in spirit like resampling from a pooled dataset rather than reading it strictly in time order.

**The solution.** Store the transition $(s_t, a_t, r_t, s_{t+1})$ in a **replay memory** $\mathcal{D}$. Then, instead of training on the most recent transition, sample random mini-batches from $\mathcal{D}$.

To perform experience replay, repeat the following:

1. **Sample** an experience tuple from the dataset: $(s, a, r, s') \sim \mathcal{D}$.
2. **Compute** the target value for the sampled tuple: $r + \gamma \max_{a'} \hat{Q}(s', a', \mathbf{w})$.
3. **Use SGD** to update the network weights $\mathbf{w}$:

$$\Delta \mathbf{w} = \alpha \left( r + \gamma \max_{a'} \hat{Q}(s', a', \mathbf{w}) - Q(s, a, \mathbf{w}) \right) \nabla_{\mathbf{w}} \hat{Q}(s, a, \mathbf{w}). \tag{7.5.1}$$

**Benefits of experience replay.**

- Breaks temporal correlations between consecutive samples.
- Each experience can be used for multiple weight updates, improving data efficiency.
- The replay buffer effectively smooths the training distribution over many past behaviors.

**Limitation.** Using the target as a scalar works for the current update, but the network weights will get updated on the next round, changing the target value. This introduces non-stationarity in the targets.

### 7.5.4 Fixed Q-Targets

A second source of instability is that the target used to compute the TD error depends on the **same weights w** that we are trying to update. In the original Q-learning with value function approximation, both the Q estimation and the Q target shift at each time step. This is analogous to a cat (Q estimation) chasing a mouse (Q target): if both the cat and the mouse are moving, the cat's trajectory will oscillate wildly, producing an **oscillated training history**.

**The solution.** Fix the target weights used in the target calculation for multiple updates.

Let $\mathbf{w}^-$ be a separate set of parameters used in the target (the **target network**), and let $\mathbf{w}$ be the weights that are being updated (the **online network**). The target network's weights are held fixed and only periodically synchronized with the online network.

To perform experience replay with fixed targets, repeat the following:

1. **Sample** an experience tuple from the dataset: $(s, a, r, s') \sim \mathcal{D}$.

2. **Compute** the target value for the sampled tuple: $r + \gamma \max_{a'} \hat{Q}(s', a', \mathbf{w}^-)$.

3. **Use stochastic gradient descent** to update the network weights:

$$\Delta \mathbf{w} = \alpha \left( r + \gamma \max_{a'} \hat{Q}(s', a', \mathbf{w}^-) - Q(s, a, \mathbf{w}) \right) \nabla_{\mathbf{w}} \hat{Q}(s, a, \mathbf{w}). \tag{7.5.2}$$

The key difference is that $\mathbf{w}^-$ in the target is **not** updated at every step. Instead, $\mathbf{w}^-$ is periodically copied from $\mathbf{w}$ every $C$ steps.

### 7.5.5 The Complete DQN Algorithm

Putting it all together, the full DQN algorithm is given in Algorithm 12.

**Algorithm 12** Deep Q-Network (DQN) with experience replay and fixed targets

**Require:** target-network update frequency $C$, learning rate $\alpha$, empty replay buffer $\mathcal{D} = \{\}$

Initialize online weights $\mathbf{w}$, target weights $\mathbf{w}^- = \mathbf{w}$, and $t = 0$
Get initial state $s_0$
**loop**
    Sample action $a_t$ via $\varepsilon$-greedy policy on the current $\hat{Q}(s_t, a; \mathbf{w})$
    Observe reward $r_t$ and next state $s_{t+1}$
    Store transition $(s_t, a_t, r_t, s_{t+1})$ in the replay buffer $\mathcal{D}$
    Sample a random minibatch of tuples $(s_j, a_j, r_j, s_{j+1})$ from $\mathcal{D}$
    **for** each $j$ in the minibatch **do**
        **if** the episode terminated at step $j + 1$ **then**
            $y_j \leftarrow r_j$
        **else**
            $y_j \leftarrow r_j + \gamma \max_{a'} \hat{Q}(s_{j+1}, a', \mathbf{w}^-)$
        **end if**
        Do a gradient descent step on $\left(y_j - \hat{Q}(s_j, a_j; \mathbf{w})\right)^2$:
        $\Delta \mathbf{w} = \alpha \left(y_j - \hat{Q}(s_j, a_j; \mathbf{w})\right) \nabla_{\mathbf{w}} \hat{Q}(s_j, a_j; \mathbf{w})$
    **end for**
    $t \leftarrow t + 1$
    **if** $\mathrm{mod}(t, C) = 0$ **then**
        $\mathbf{w}^- \leftarrow \mathbf{w}$
    **end if**
**end loop**

The algorithm elegantly combines three ideas: (1) Q-learning for off-policy control, (2) experience replay for breaking correlations, and (3) fixed targets for training stability.

### 7.5.6 Performance of DQN on Atari

The empirical results of DQN on Atari 2600 games were striking. DQN achieved scores comparable to or exceeding professional human testers across a diverse set of 49 games, all using the same architecture and hyperparameters.

An ablation study (Table 7.5.2) reveals the importance of each component.

**Table 7.5.2.** Ablation study of DQN components on selected Atari games.

| Game | Replay + Fixed-Q | Replay + Q-learning | No Replay + Fixed-Q | No Replay + Q-learning |
|---|---|---|---|---|
| Breakout | 316.81 | 240.73 | 10.16 | 3.17 |
| Enduro | 1006.3 | 831.25 | 141.89 | 29.1 |
| River Raid | 7446.62 | 4102.81 | 2867.66 | 1453.02 |
| Seaquest | 2894.4 | 822.55 | 1003 | 275.81 |
| Space Invaders | 1088.94 | 826.33 | 373.22 | 301.99 |

The results demonstrate that:

- **Experience replay** provides a massive performance boost across all games. Without it, performance drops dramatically.
- **Fixed Q-targets** further improve performance on top of experience replay.
- The combination of both innovations (**Replay + Fixed-Q**) consistently achieves the best performance.

## 7.6 Extensions to DQN

The success of DQN on Atari inspired a rich line of follow-up work improving deep reinforcement learning.

**Double DQN (Van Hasselt et al., 2016).** Standard Q-learning is known to overestimate action values because it uses the same network to both select and evaluate actions ($\max_a Q(s,a)$ selects the best action and simultaneously uses that Q-value as the estimate). Double DQN decouples these two operations: the online network selects the best action, but the target network evaluates its value. This significantly reduces overestimation bias and improves performance.

**Prioritized replay (Schaul et al., 2016).** Instead of sampling uniformly from the replay buffer, prioritized experience replay samples transitions with larger TD errors more frequently. The intuition is that transitions where the agent's prediction is most wrong are the most informative for learning. This approach stores the last encountered TD error along with each transition in the replay buffer and uses it to define sampling probabilities.

**Dueling DQN (Wang et al., 2016).** The dueling architecture (ICML 2016 Best Paper) decomposes the Q-function into two streams: a **value stream** $V(s)$ that estimates how good it is to be in state $s$, and an **advantage stream** $A(s,a)$ that estimates the relative advantage of each action. The Q-value is then reconstructed as $Q(s,a) = V(s) + A(s,a)$.[32] This decomposition allows the network to learn which states are valuable (or not) without having to learn the effect of each action in each state separately, leading to more efficient learning.

**Rainbow (Hessel et al., 2018).** A natural culmination of this line of work is Rainbow, which combines Double DQN, prioritized replay, dueling networks, and several other improvements (multi-step returns, distributional RL, and noisy networks) into a single integrated agent that outperforms each component in isolation on the Atari benchmark.

## 7.7 Applications: Deep RL for Business Decision Making

Deep reinforcement learning has found compelling applications across a range of business domains. We highlight several recent papers published in top journals that illustrate the breadth and depth of these applications.

*Remark* (How a business researcher should read these applications). In each case the firm faces a *sequential* decision under uncertainty—which coupon, ad, or recommendation to send now, anticipating how it changes the customer's future state and responses. This is the dynamic generalization of the A/B testing and bandit problems many readers already know: a one-shot randomized experiment estimates an average treatment effect; a multi-armed bandit balances exploration against exploitation for a single repeated decision; and reinforcement learning extends both to a *multi-period* policy that maximizes long-run (discounted) value while explicitly accounting for state transitions. Deep function approximation is what makes this tractable when the customer "state" is high-dimensional.

### 7.7.1 Dynamic Coupon Targeting

Liu (2023), in *Marketing Science*, applies batch deep reinforcement learning (BDRL) to dynamic coupon targeting in a livestream shopping context.

[32] Written this way, $V$ and $A$ are not separately identified: adding a constant to $V$ and subtracting it from every $A(s,a)$ leaves $Q$ unchanged. The paper resolves this by re-centering the advantage stream, e.g. $Q(s,a) = V(s) + \left(A(s,a) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s,a')\right)$, which forces the advantages to average to zero across actions and pins down the decomposition. The body's $Q = V + A$ is the simplified statement of the idea.

**Problem setting.**

- The firm must decide how to dynamically target coupons to consumers over time.
- The state space is high-dimensional (capturing consumer characteristics, browsing history, and purchase behavior).
- The action space involves selecting coupon types and discount levels.
- Transition dynamics and rewards are unknown and must be learned from observational data.

**Methodology.**

- The paper uses a batch (offline) variant of deep Q-learning, where the agent learns from a fixed dataset of historical interactions rather than through online exploration.
- Deep neural networks are used to represent the high-dimensional state space and alleviate the curse of dimensionality.
- The approach incorporates a doubly robust estimator for policy evaluation.[33]

**Key findings.**

- The BDRL solution increases the platform's revenue significantly compared to static targeting policies.
- Gains come from more effective and automatic targeting of consumers based on heterogeneity and dynamics, using counterfactually rich temporal differences among consumers over time.
- The approach demonstrates the practical value of deep RL for high-frequency, high-dimensional business decision-making.

### 7.7.2 Ensembling Experimentation Along the Customer Journey

Song and Sun (2024), in *Management Science*, propose using deep reinforcement learning to optimize interventions along the customer journey based on an ensemble of historical experiments.

[33]This is the same doubly robust idea as in causal inference and off-policy evaluation: it combines a direct (outcome-model) estimate of a policy's value with an importance-weighted correction, and remains consistent if *either* the outcome model *or* the propensity (behavior-policy) model is correctly specified. It matters offline because the firm must estimate the value of a counterfactual targeting policy from logged data it did not itself generate by experimentation.

**Problem setting.**

- Firms run multiple randomized experiments at different stages of the customer journey (e.g., ad exposure, landing-page design, promotional offers), but these experiments are typically analyzed in isolation.
- The challenge is to learn an optimal *sequence* of interventions across stages, accounting for intertemporal effects and heterogeneity.

**Methodology.**

- The paper proposes a Bayesian Recurrent Q-Network (BRQN) framework that combines Bayesian deep learning with recurrent neural networks to handle partial observability and sequential decision-making.
- The framework learns from an ensemble of historical experiments to guide future intervention trials, bridging the gap between RL and classical experimentation.

**Key findings.**

- Learning from multiple historical experiments jointly yields significantly higher average rewards than learning from any single experiment in isolation.
- The approach shows the promise of fusing RL with multiple experiments for optimizing multi-stage customer journeys.

### 7.7.3 Sequential Targeting

Wang et al. (2023a), in *Management Science*, design a DRL-based personalized targeting strategy in a sequential setting, addressing three important challenges: (1) *forward looking*, balancing exploration and exploitation; (2) *scalability*, coping with a high-dimensional state and policy space; and (3) *adaptivity*, learning while continuously interacting with consumers. The proposed DRL agent generates substantially more long-term revenue than the conventional bandit-based and greedy approaches can.

### 7.7.4 Career-Path Recommendations

Kokkodis and Ipeirotis (2021), in *Management Science*, apply reinforcement learning to provide demand-aware career-path recommendations for contractors on an online labor market. The framework combines reinforcement learning, Bayesian inference, and guided learning to promote future career-path recommendations while optimizing current market trends. It uses market information to identify current trends and

project future career paths, and recommends skills that contractors should learn to maximize long-term earnings.

## 7.8 Policy-Based Methods

For a thorough and accessible exposition of the material in this section, we recommend Weng's tutorial on policy gradient algorithms,[34] which provides detailed derivations and intuitive explanations.

### 7.8.1 From Value to Policy Approximations

So far, we have focused on approximating the value functions $v^\pi(s)$ and $q^\pi(s,a)$ given the policy $\pi$. An entirely different perspective is to **parameterize the policy function directly**,

$$\pi_\theta(a \mid s), \tag{7.8.1}$$

where $\theta$ is the parameter vector (e.g., the weights of a deep neural network) that determines the policy.

The central optimization problem becomes

$$\max_\theta \quad \mathcal{J}(\theta) = \mathbb{E}_{s_0\sim p_0}\left[V^{\pi_\theta}(s_0)\right] = \mathbb{E}^{\pi}_{s_0\sim p_0}\left[\sum_{t=0}^{T-1}\gamma^t r_t\right], \tag{7.8.2}$$

where $\pi_\theta$ is represented by a DNN with parameter $\theta$. The action and reward from the policy are all we need to optimize the policy directly, no value function estimation is required.

### 7.8.2 Value-Based RL vs. Policy-Based RL

There are three major paradigms for deep reinforcement learning:

- **Value-based RL** learns the value function and derives the policy from it (e.g., by acting greedily with respect to the Q-function). DQN is a prime example.
- **Policy-based RL** learns the optimal policy directly without explicitly computing a value function. The policy gradient methods we discuss below fall into this category.
- **Actor-critic** methods learn *both* a policy and a value function simultaneously. The policy (the "actor") decides which actions to take, while the value function (the "critic") evaluates how good those actions are.

[34] https://lilianweng.github.io/posts/2018-04-08-policy-gradient/

These three approaches can be visualized as a Venn diagram: value-based and policy-based methods occupy distinct regions, while actor-critic methods sit at their intersection, combining elements of both.

### 7.8.3 Pros and Cons of Policy-Based RL

**Advantages.**

- **Stochastic policies.** Policy-based methods can directly learn stochastic policies, which is important for games (e.g., rock-paper-scissors, where a deterministic policy is trivially exploitable) and for partially observable environments.
- **High-dimensional or continuous action spaces.** Policy-based RL is effective when the action space is high-dimensional or continuous (e.g., robotic control, LLM token generation), where computing $\max_a Q(s,a)$ is intractable.
- **Better convergence properties.** Policy gradient methods enjoy convergence guarantees similar to policy iteration, since the objective function is smooth in the policy parameters.

**Disadvantages.**

- **High variance.** Policy gradient estimators tend to have high variance, which can slow learning and make training unstable.
- **Local optima.** Policy gradient methods typically converge to a local optimum rather than the global optimum. The entire policy space is harder to explore and optimize than the value function space.

### 7.8.4 Policy Gradient for One-Step MDPs

To build intuition, consider a simple MDP with **one step**: start with $s \sim d(s)$, take one action, and terminate with reward $r = R(s,a)$.

The expected reward of policy $\pi_\theta(s,a)$ is

$$J(\theta) = \mathbb{E}_{\pi_\theta}[r] = \sum_{s\in\mathcal{S}} d(s) \sum_{a\in\mathcal{A}} \pi_\theta(s,a)\, r. \tag{7.8.3}$$

The gradient is

$$\nabla_\theta J(\theta) = \sum_{s\in\mathcal{S}} d(s) \sum_{a\in\mathcal{A}} \pi_\theta(s,a)\, \nabla_\theta \log \pi_\theta(s,a)\, r. \tag{7.8.4}$$

This uses the **log-likelihood trick** (also called the **score-function** trick or the **REINFORCE** trick):

$$\nabla_\theta \pi_\theta(s,a) = \pi_\theta(s,a)\frac{\nabla_\theta \pi_\theta(s,a)}{\pi_\theta(s,a)} = \pi_\theta(s,a)\nabla_\theta \log \pi_\theta(s,a). \tag{7.8.5}$$

The term $\nabla_\theta \log \pi_\theta(s,a)$ is called the **score function** or the **gradient of the log-likelihood ratio**. The gradient can thus be written as an expectation:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[r\,\nabla_\theta \log \pi_\theta(s,a)\right]. \tag{7.8.6}$$

This is remarkable: we can estimate the gradient by sampling trajectories from the policy, without needing to know the environment dynamics.

*Remark* (The score function is the MLE score; this is the likelihood-ratio estimator). The object $\nabla_\theta \log \pi_\theta(s,a)$ is literally the **score** in the maximum-likelihood sense: the gradient of a log-likelihood with respect to its parameters, where the "likelihood" is the probability the policy assigns to the action it took. An econometrician will recognize (7.8.5) as the **likelihood-ratio** (or score-function) trick for differentiating an expectation whose underlying distribution depends on the parameter, $\nabla_\theta \mathbb{E}_{p_\theta}[f] = \mathbb{E}_{p_\theta}[\,f\,\nabla_\theta \log p_\theta\,]$. It is the same device behind score-function gradient estimators in simulation-based estimation and behind the classical identity $\mathbb{E}_{p_\theta}[\nabla_\theta \log p_\theta] = 0$ (the expected score is zero). Its key virtue here is that the unknown environment dynamics $p(s' \mid s,a)$ do not depend on $\theta$, so they contribute nothing to the score and never have to be modeled—the basis for "model-free" policy optimization.

### 7.8.5 Policy Gradient for Multi-Step MDPs

Now consider a full multi-step MDP. A **state-action trajectory** from one episode is

$$\tau = (s_0, a_0, r_1, \ldots, s_{T-1}, a_{T-1}, r_T, s_T) \sim \big(\pi_\theta, P(s_{t+1} \mid s_t, a_t)\big). \tag{7.8.7}$$

Define $R(\tau) = \sum_{t=0}^{T} R(s_t, a_t)$ as the total reward for trajectory $\tau$ (assuming $\gamma = 1$ for simplicity). The policy value is

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1} R(s_t,a_t)\right] = \sum_{\tau} P(\tau;\theta)R(\tau), \tag{7.8.8}$$

where $P(\tau;\theta) = \mu(s_0)\prod_{t=0}^{T-1} \pi_\theta(a_t \mid s_t)\, p(s_{t+1} \mid s_t, a_t)$ is the probability of trajectory $\tau$ under policy $\pi_\theta$.

Our goal is to find the optimal policy parameter

$$\theta^* = \arg\max_\theta J(\theta) = \arg\max_\theta \sum_\tau P(\tau;\theta)R(\tau). \tag{7.8.9}$$

### 7.8.6 Taking the Gradient

To compute the gradient $\nabla_\theta J(\theta)$, we apply the log-likelihood trick at the trajectory level:

$$\nabla_\theta J(\theta) = \nabla_\theta \sum_\tau P(\tau;\theta)R(\tau) = \sum_\tau \nabla_\theta P(\tau;\theta)R(\tau) = \sum_\tau P(\tau;\theta)\nabla_\theta \log P(\tau;\theta)\, R(\tau). \tag{7.8.10}$$

Now we decompose $\nabla_\theta \log P(\tau;\theta)$. Since

$$P(\tau;\theta) = \mu(s_0) \prod_{t=0}^{T-1} \pi_\theta(a_t \mid s_t)\, p(s_{t+1} \mid s_t, a_t), \tag{7.8.11}$$

taking the log gives

$$\log P(\tau;\theta) = \log \mu(s_0) + \sum_{t=0}^{T-1} \Big[\log \pi_\theta(a_t \mid s_t) + \log p(s_{t+1} \mid s_t, a_t)\Big]. \tag{7.8.12}$$

The gradient with respect to $\theta$ eliminates terms that do not depend on $\theta$:

$$\nabla_\theta \log P(\tau;\theta) = \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t). \tag{7.8.13}$$

This is a crucial result: **the gradient does not require knowledge of the transition dynamics** $p(s_{t+1} \mid s_t, a_t)$. All we need is the ability to compute $\nabla_\theta \log \pi_\theta(a_t \mid s_t)$, which depends only on the parameterized policy.

### 7.8.7 The REINFORCE Algorithm

Combining the results above, the policy gradient can be estimated as

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[ \left( \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right) \left( \sum_{t=1}^{T} r(s_t, a_t) \right) \right]. \tag{7.8.14}$$

This expectation can be approximated by sampling $N$ trajectories:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left( \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t}) \right) \left( \sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right). \tag{7.8.15}$$

The full **REINFORCE** algorithm (Williams, 1992), also called vanilla policy gradient with Monte Carlo returns, is given in Algorithm 13.

**Algorithm 13** REINFORCE (vanilla policy gradient with Monte Carlo)

**repeat**
  Sample trajectories $\{\tau^i\}$ from $\pi_\theta(a_t \mid s_t)$
  Estimate the gradient:

$$\nabla_\theta J(\theta) \approx \sum_i \left( \sum_t \nabla_\theta \log \pi_\theta(a_t^i \mid s_t^i) \right) \left( \sum_t r(s_t^i, a_t^i) \right)$$

  Update: $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
**until** converged

The algorithm is conceptually simple: run the current policy to collect data, compute the gradient estimate, and take a gradient ascent step.

### 7.8.8 Policy Gradient vs. Maximum Likelihood

It is instructive to compare the policy gradient estimator with the maximum likelihood estimator used in supervised learning (also called **imitation learning** or **behavioral cloning**).

**Policy gradient estimator.**

$$\nabla_\theta J(\theta) \approx \frac{1}{M} \sum_{m=1}^{M} \left( \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{t,m} \mid s_{t,m}) \right) \left( \sum_{t=1}^{T} r(s_{t,m}, a_{t,m}) \right). \tag{7.8.16}$$

**Maximum likelihood estimator.**

$$\nabla_\theta J_{\mathrm{ML}}(\theta) \approx \frac{1}{M} \sum_{m=1}^{M} \left( \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{t,m} \mid s_{t,m}) \right). \tag{7.8.17}$$

The only difference is that the policy gradient **weights each trajectory's log-likelihood by its total reward**. The interpretation is intuitive: **good actions (those that lead to high rewards) are made more likely, and bad actions are made less likely.** In maximum likelihood (imitation learning), all demonstrated actions are treated equally, the agent simply mimics the observed behavior regardless of quality.

### 7.8.9 The Policy Gradient Theorem

The policy gradient theorem (Sutton et al., 2000) generalizes the likelihood-ratio approach to a broader set of objective functions.

**Theorem 7.8.1** (Policy Gradient Theorem)**.** *For any differentiable policy $\pi_\theta(s,a)$, and for any of the policy objective functions $J = J_1$ (episodic reward), $J_{avR}$ (average reward per time step), or $\frac{1}{1-\gamma}J_{avV}$ (average value), the policy gradient is*

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\, Q^{\pi_\theta}(s,a)\right]. \tag{7.8.18}$$

The policy gradient theorem says that the **policy gradient equals the Q-function following the policy, weighted by the score function of the policy**. This result is fundamental because it provides a unified expression for the gradient regardless of which objective function we choose.

*Remark* (REINFORCE is the Monte Carlo version of the theorem). Comparing with (7.8.15), REINFORCE is exactly Theorem 7.8.1 with the unknown $Q^{\pi_\theta}(s_t,a_t)$ replaced by a single-sample Monte Carlo estimate—the realized reward-to-go $G_t = \sum_{t'\geq t} r_{t'}$, which is unbiased for $Q^{\pi_\theta}(s_t,a_t)$. Read this way, everything that follows—reward-to-go, baselines, and the actor–critic methods below—is just a sequence of progressively lower-variance ways of estimating the same $Q^{\pi_\theta}$ that appears in the theorem.

### 7.8.10 Score Function

The score function $\nabla_\theta \log \pi_\theta(s,a)$ takes different forms depending on the policy parameterization.

**Discrete action policy (softmax).**

$$\pi_\theta(s,a) = \frac{\exp\left(\phi(s,a)^\top \theta\right)}{\sum_{a'} \exp\left(\phi(s,a')^\top \theta\right)}. \tag{7.8.19}$$

The score function is

$$\nabla_\theta \log \pi_\theta(s,a) = \phi(s,a) - \mathbb{E}_{\pi_\theta}[\phi(s,\cdot)]. \tag{7.8.20}$$

This is the feature vector for the chosen action minus the expected feature vector under the policy, it points in the direction that makes the chosen action more likely relative to the average.

*Remark* (The softmax policy is a conditional logit). This parameterization is identical to McFadden's **conditional logit** discrete-choice model: $\pi_\theta(s,a)$ is the choice probability of "alternative" $a$ in "choice situation" $s$, with linear index (utility) $\phi(s,a)^\top\theta$. The score $\phi(s,a) - \mathbb{E}_{\pi_\theta}[\phi(s,\cdot)]$ is exactly the familiar logit score: the attributes of the chosen alternative minus their choice-probability-weighted average across alternatives. Policy-gradient learning therefore differs from estimating a logit by maximum likeli-

hood only in *what* multiplies that score—a reward or advantage signal, rather than an indicator of the observed choice (compare §7.8.8).

**Continuous action policy (Gaussian).** The policy is Gaussian with mean linear in state features:

$$\mu(s) = \phi(s)^\top \theta, \qquad a \sim \mathcal{N}(\mu(s), \sigma^2). \tag{7.8.21}$$

The score function is

$$\nabla_\theta \log \pi_\theta(s, a) = \frac{(a - \mu(s))\,\phi(s)}{\sigma^2}. \tag{7.8.22}$$

This pushes the mean toward the selected action, proportional to how far the action is from the current mean.

*Remark* (The Gaussian policy is a linear-normal model). With $a \sim \mathcal{N}(\phi(s)^\top \theta, \sigma^2)$, the policy is simply a linear regression of the action on state features with Gaussian noise. Its score $\frac{(a-\mu(s))\,\phi(s)}{\sigma^2}$ is precisely the score of that regression's log-likelihood with respect to $\theta$—"residual times regressor, divided by the noise variance"—so the continuous-action case reuses the same Gaussian-MLE algebra the reader already knows.

### 7.8.11 Reducing the Variance of the Policy Gradient

The REINFORCE gradient estimator is **unbiased but very noisy** (high variance). The variance arises because the total reward $R(\tau)$ for an entire trajectory is used to weight every action's log-probability in that trajectory, even though some of that reward may have nothing to do with a particular action. Two key techniques reduce this variance.

**Temporal causality.** The principle of temporal causality states that a policy at time $t'$ cannot affect a reward at time $t$ for $t < t'$. Exploiting this:

$$\nabla_\theta J(\theta) = \mathbb{E}_\tau \left[ \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \sum_{t'=t}^{T-1} r_{t'} \right] = \mathbb{E}_\tau \left[ \sum_{t=0}^{T-1} G_t \cdot \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right], \tag{7.8.23}$$

where $G_t = \sum_{t'=t}^{T-1} r_{t'}$ is the **reward-to-go** from time step $t$ onward. This replaces the total trajectory reward with only the future rewards from each time step, significantly reducing variance.

The estimated gradient becomes

$$\nabla_\theta \mathbb{E}[R] \approx \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{T-1} G_t^{(i)} \cdot \nabla_\theta \log \pi_\theta(a_t^i \mid s_t^i). \tag{7.8.24}$$

**Subtracting a baseline.** We can subtract a **baseline** $b(s_t) = \mathbb{E}[r_t + r_{t+1} + \cdots + r_{T-1}]$ from the return without introducing bias:

$$\nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}[R] = \mathbb{E}_\tau \left[ \sum_{t=0}^{T-1} (G_t - b(s_t)) \cdot \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right]. \tag{7.8.25}$$

Why is this unbiased? Because $\mathbb{E}_\tau \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\, b(s_t)\right] = 0$ for any baseline that depends only on the state.

The key insight is that subtracting the baseline changes the gradient from asking "was this action good?" to asking "**was this action better than expected?**" This increases the log-probability of actions in proportion to how much their returns exceed the expected return, and is both **unbiased** and **variance-reducing**.

*Remark* (Baselines are control variates). Subtracting $b(s_t)$ is the classic **control-variate** technique for Monte Carlo variance reduction. We add to the estimator a term with *known zero mean*—here $\mathbb{E}_\tau[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\, b(s_t)] = 0$, because $\mathbb{E}_{a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a \mid s)] = \sum_a \nabla_\theta \pi_\theta(a \mid s) = \nabla_\theta \sum_a \pi_\theta(a \mid s) = \nabla_\theta 1 = 0$ for any function of the state alone—so the estimator stays unbiased while its variance falls. The variance-minimizing baseline is close to the state value $V^\pi(s_t) = \mathbb{E}[G_t \mid s_t]$, which is why the natural choice subtracts the expected return and works with the *advantage* $G_t - V^\pi(s_t)$.

### 7.8.12 Vanilla Policy Gradient with Baseline

The complete vanilla policy gradient algorithm with a learned baseline is given in Algorithm 14.

Note that vanilla policy gradient is **on-policy**: it uses only data collected from the current policy $\pi_\theta$ to update the parameters. This raises the question: how can we generalize to **off-policy** settings, where we reuse data from older policies?

### 7.8.13 Off-Policy Policy Gradient

What if we want to use samples collected from a different policy $\bar{\pi}$ to update our current policy $\pi_{\theta'}$? This is the off-policy setting, which is important for data efficiency since we can reuse past experience.

The idea is to use **importance sampling**. Given a proposal distribution $q(x)$ and a target distribution $p(x)$,

$$\mathbb{E}_{x\sim p(x)}[f(x)] = \int p(x) f(x)\, dx = \int q(x) \frac{p(x)}{q(x)} f(x)\, dx = \mathbb{E}_{x\sim q(x)} \left[ \frac{p(x)}{q(x)} f(x) \right]. \tag{7.8.26}$$

*Remark* (Importance sampling is inverse-probability reweighting). This is the same reweighting idea as **inverse-probability weighting (IPW)** in causal inference and **im-**

**Algorithm 14** Vanilla policy gradient with a learned baseline

**procedure** POLICYGRADIENT($\alpha$)
  Initialize policy parameters $\theta$ and baseline values $b(s)$ for all $s$ (e.g., to 0)
  **for** iteration $= 1, 2, \ldots$ **do**
    Collect a set of $m$ trajectories by executing the current policy $\pi_\theta$
    **for** each time step $t$ of each trajectory $\tau^{(i)}$ **do**
      Compute the return $G_t^{(i)} = \sum_{t'=t}^{T-1} r_{t'}$
      Compute the advantage estimate $\hat{A}_t^{(i)} = G_t^{(i)} - b(s_t)$
    **end for**
    Re-fit the baseline to the empirical returns by updating $\mathbf{w}$ to minimize
    $$\sum_{i=1}^{m} \sum_{t=0}^{T-1} \left\| b(s_t) - G_t^{(i)} \right\|^2$$
    Update the policy parameters $\theta$ using the policy gradient estimate
    $$\hat{g} = \sum_{i=1}^{m} \sum_{t=0}^{T-1} \hat{A}_t^{(i)} \nabla_\theta \log \pi_\theta(a_t^{(i)} \mid s_t^{(i)})$$
    with an optimizer such as SGD ($\theta \leftarrow \theta + \alpha \cdot \hat{g}$) or Adam
  **end for**
  **return** $\theta$ and the baseline values $b(s)$
**end procedure**

**portance sampling** in simulation: to evaluate an expectation under a target distribution $p$ using draws from a different "behavior" distribution $q$, reweight each draw by the ratio $p(x)/q(x)$. Here $q$ is the old/behavior policy that generated the data and $p$ is the new policy we wish to evaluate. As with IPW, the estimator is unbiased but its variance explodes when the two distributions are far apart and some ratios become huge—which, multiplied over a long trajectory, is exactly the instability that the per-timestep approximation and the trust-region constraint below are designed to control.

Applying this at the trajectory level, the importance weight is $\frac{p_\theta(\tau)}{p_{\bar{\theta}}(\tau)} = \prod_{t=1}^{T} \frac{\pi_\theta(a_t|s_t)}{\bar{\pi}(a_t|s_t)}$, which can become very small or very large for longer trajectories.

To mitigate this, we can instead apply importance sampling at the **per-timestep** level rather than the trajectory level, which is much less likely to explode or vanish:

$$\nabla_{\theta'} J(\theta') \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \frac{\pi_{\theta'}(s_{i,t}, a_{i,t})}{\pi_\theta(s_{i,t}, a_{i,t})} \nabla_{\theta'} \log \pi_{\theta'}(a_{i,t} \mid s_{i,t}) \left( \left( \sum_{t'=t}^{T} r(s_{i,t'}, a_{i,t'}) \right) - b \right). \tag{7.8.27}$$

In practice, the per-timestep importance ratio $\frac{\pi_{\theta'}(s_t,a_t)}{\pi_\theta(s_t,a_t)}$ is often approximated as 1, simplifying the computation.

**Key challenge.** If the policy changes too much before sampling new data, the data no longer reflects the states the updated policy would visit. To address this, we can

**constrain the policy not to change too much** between updates:

$$\mathbb{E}_{s\sim\pi_\theta}\left[D_{\mathrm{KL}}\big(\pi_{\theta'}(\cdot \mid s)\,\big\|\,\pi_\theta(\cdot \mid s)\big)\right] \le \delta. \tag{7.8.28}$$

This constraint is the foundation of trust-region methods such as TRPO (Trust Region Policy Optimization) and PPO (Proximal Policy Optimization), which have become widely used in practice, including in the training of large language models.

### 7.8.14 Reducing Variance with a Critic

Recall the policy gradient update with baseline:

$$\nabla_\theta \mathbb{E}[R] \approx \frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\,(G_t^{(i)} - b(s_t)). \tag{7.8.29}$$

The MC rollout return $G_t^{(i)}$ is an unbiased estimate of the Q-function, but it comes with high variance. We can instead use a **critic**, a learned approximation of the Q-function, to reduce this variance.

**The actor-critic architecture.**

- **Critic:** updates the Q-function approximation $Q_{\mathbf{w}}(s,a)$ with parameter $\mathbf{w}$, which is policy evaluation through function approximation.
- **Actor:** updates the policy function $\pi_\theta$ using the Q-function produced by the critic.
- **Baseline:** the average reward $V(s_t) = \mathbb{E}_{a_t\sim\pi_\theta(\cdot|s_t)}[Q(s_t,a_t)]$ serves as a natural baseline.

The interaction between the actor and the critic is depicted in Figure 7.8.1.

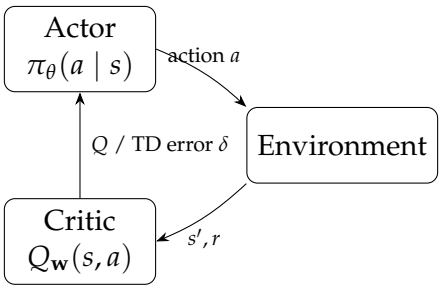


**Figure 7.8.1.** Actor-critic architecture: the actor selects actions in the environment, the critic evaluates them via a learned value function, and the critic's signal (the Q-value or TD error) drives the actor's policy gradient update.

The key quantity is the **advantage function**

$$A^{\pi_\theta}(s,a) = Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s). \tag{7.8.30}$$

The advantage function measures how much better action $a$ is compared to the average action under the current policy. It can be approximated by the **TD error**

$$\delta^{\pi_\theta} = r + \gamma V^{\pi_\theta}(s') - V^{\pi_\theta}(s). \tag{7.8.31}$$

In expectation, the TD error equals the advantage: $\mathbb{E}_{\pi_\theta}[\delta^{\pi_\theta} \mid s,a] = A^{\pi_\theta}(s,a)$.[35]

The policy gradient then becomes

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\, A^{\pi_\theta}(s,a)\right]. \tag{7.8.32}$$

### 7.8.15 Compatible Function Approximation Theorem

A natural concern arises: since the critic's Q-function approximation has errors, does this bias the policy gradient updates?

**Theorem 7.8.2** (Compatible Function Approximation)**.** *If the following two conditions are satisfied:*

1. *the value function approximator is* ***compatible*** *with the policy,* $\nabla_{\mathbf{w}} Q_{\mathbf{w}}(s,a) = \nabla_\theta \log \pi_\theta(s,a)$*; and*
2. *the value function parameters* $\mathbf{w}$ *minimize the mean-squared error,*

$$\varepsilon = \mathbb{E}_{\pi_\theta}\left[\left(Q^{\pi_\theta}(s,a) - Q_{\mathbf{w}}(s,a)\right)^2\right];$$

*then the policy gradient is* ***exact****:*

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\, Q_{\mathbf{w}}(s,a)\right]. \tag{7.8.33}$$

This theorem provides reassurance that if the critic satisfies certain compatibility conditions, the actor's gradient updates are not biased by the approximation error in the critic. In practice, the conditions need not hold exactly for the approach to work well.

[35] This holds by the definition of the value functions: $\mathbb{E}[\,r + \gamma V^{\pi_\theta}(s') \mid s,a\,] = Q^{\pi_\theta}(s,a)$, so subtracting $V^{\pi_\theta}(s)$ gives $Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s) = A^{\pi_\theta}(s,a)$. Thus a single one-step TD error is an unbiased estimate of the advantage, letting the critic supply the "better than average?" signal using only a learned $V$, with no separate $Q$-network.

### 7.8.16 Simple Action-Value Actor-Critic (QAC) Algorithm

A concrete actor-critic algorithm using linear value function approximation is the **Simple QAC** (Q Actor-Critic) (Konda and Tsitsiklis, 2000):

- **Critic:** use a linear value function approximation, $Q_{\mathbf{w}}(s,a) = \psi(s,a)^\top \mathbf{w}$, and update $\mathbf{w}$ by linear TD(0).
- **Actor:** update $\theta$ by policy gradient.

The procedure is given in Algorithm 15.

**Algorithm 15** Simple Action-Value Actor-Critic (QAC)

**for** each step **do**
  Generate a sample $s, a, r, s', a'$ following $\pi_\theta$
  $\delta \leftarrow r + \gamma Q_{\mathbf{w}}(s', a') - Q_{\mathbf{w}}(s,a)$ ▷ TD error
  $\mathbf{w} \leftarrow \mathbf{w} + \beta\, \delta\, \psi(s,a)$ ▷ Critic update
  $\theta \leftarrow \theta + \alpha\, \nabla_\theta \log \pi_\theta(s,a)\, Q_{\mathbf{w}}(s,a)$ ▷ Actor update
**end for**

### 7.8.17 Asynchronous Advantage Actor-Critic (A3C)

The **A3C** algorithm (Mnih et al., 2016) scales actor-critic methods through **parallel training of multiple actors**. It is a policy gradient method in which critics learn the value function while multiple actors are trained in parallel, each interacting with its own copy of the environment.

**Key design principles.**

1. **Global parameters** $\theta$ and $\mathbf{w}$ are shared, with thread-specific copies $\theta'$ and $\mathbf{w}'$.
2. Each thread runs independently: reset gradients, synchronize with global parameters, sample a trajectory, and compute local gradients.
3. Gradients are accumulated locally over a short trajectory segment and then used to **asynchronously update** the global parameters.

The algorithm outline is given in Algorithm 16.

**Reading the two updates.** The actor accumulation $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi_{\theta'}(a_i \mid s_i)\,(R - V_{\mathbf{w}'}(s_i))$ is the policy-gradient *ascent* direction, with the bootstrapped return $R$ playing the role of $G_i$ and $V_{\mathbf{w}'}(s_i)$ acting as the baseline, so $R - V_{\mathbf{w}'}(s_i)$ is an advantage estimate. The critic accumulation $d\mathbf{w} \leftarrow d\mathbf{w} + 2(R - V_{\mathbf{w}'}(s_i))\,\nabla_{\mathbf{w}'}(R - V_{\mathbf{w}'}(s_i))$ is the gradient of the squared prediction error $(R - V_{\mathbf{w}'}(s_i))^2$; the critic's global update therefore *descends*

**Algorithm 16** Asynchronous Advantage Actor-Critic (A3C), per-thread loop

**while** $T \leq T_{\max}$ **do**
  Reset gradients: $d\theta \leftarrow 0$ and $d\mathbf{w} \leftarrow 0$
  Synchronize thread-specific parameters: $\theta' \leftarrow \theta$ and $\mathbf{w}' \leftarrow \mathbf{w}$
  Sample a starting state $s_t$; set $t_{\text{start}} \leftarrow t$
  **while** $s_t \neq$ TERMINAL **and** $t - t_{\text{start}} \leq t_{\max}$ **do**
    Pick action $a_t \sim \pi_{\theta'}(a_t \mid s_t)$ and observe $r_t, s_{t+1}$
    $t \leftarrow t + 1$
  **end while**
  Initialize the return estimate: $R \leftarrow 0$ if $s_t$ is TERMINAL, else $R \leftarrow V_{\mathbf{w}'}(s_t)$
  **for** $i = t-1, \ldots, t_{\text{start}}$ **do**
    $R \leftarrow \gamma R + r_i$ ▷ MC estimate of $G_i$
    Accumulate $\theta'$ gradient: $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi_{\theta'}(a_i \mid s_i)\,(R - V_{\mathbf{w}'}(s_i))$
    Accumulate $\mathbf{w}'$ gradient: $d\mathbf{w} \leftarrow d\mathbf{w} + 2\,(R - V_{\mathbf{w}'}(s_i))\,\nabla_{\mathbf{w}'}(R - V_{\mathbf{w}'}(s_i))$
  **end for**
  Asynchronously update the global $\theta$ using $d\theta$, and the global $\mathbf{w}$ using $d\mathbf{w}$
**end while**

this error (it moves $\mathbf{w}$ toward fitting $V$ to the observed returns), while the actor's global update *ascends* $J(\theta)$. This is the same actor/critic division of labor as in QAC, now run in parallel across threads.

**Advantages of A3C.**

- **Parallelism for stability:** running multiple actors in parallel with different exploration patterns provides diverse experience, reducing the correlation between updates. This serves a similar purpose to experience replay in DQN, but without the memory overhead.
- **On-policy learning at scale:** unlike DQN, which is off-policy, A3C remains on-policy while achieving data efficiency through parallelism.
- **Lower memory requirements:** there is no need for a large replay buffer; each thread only needs to store a short trajectory segment.

## 7.9 Application: Deep RL for Inventory Control

Gijsbrechts et al. (2022), in *Manufacturing & Service Operations Management*, rigorously evaluate deep reinforcement learning for inventory management across three classic and intractable problem settings: lost sales, dual-sourcing, and multi-echelon inventory systems.

**Problem setting.**

- Some inventory control problems are **notoriously challenging**: lost-sales systems (where unmet demand is lost rather than backordered), dual-sourcing (choosing between fast and slow suppliers), and multi-echelon networks (coordinating inventory across supply chain tiers).
- These problems lack tractable closed-form solutions and have historically relied on hand-crafted heuristics.

**Methodology.**

- The authors formulate each inventory problem as a Markov decision process and apply the Asynchronous Advantage Actor-Critic (A3C) algorithm.
- The DRL agent learns inventory replenishment policies directly from simulated demand and cost data.

**Key findings.**

- The A3C algorithm **matches the performance of well-designed heuristics**, with limited changes to the tuning parameters across different problem structures.
- Although the initial tuning was computationally demanding and time-consuming, only small adjustments to the tuning parameters were needed for the other studied problems.
- However, **generating structural policy insight and specialized near-optimal policies remains desirable**, DRL provides a powerful numerical tool, but the learned policies can be difficult to interpret and do not replace the conceptual understanding offered by structural results.

## 7.10 Modern Deep RL: From TRPO to PPO

The policy gradient methods discussed in the previous sections, REINFORCE, vanilla policy gradient with baselines, and actor-critic algorithms, suffer from a critical practical limitation: **they are highly sensitive to the step size**. In supervised learning, an overly large learning rate leads to poor convergence but the model can typically recover. In RL, the consequences are far more severe: a bad policy update leads to bad data collection, which leads to an even worse policy, creating a vicious cycle that can **collapse overall performance irreversibly**.[36]

[36] See L. Weng (2018), *Policy Gradient Algorithms*, Lil'Log, https://lilianweng.github.io/posts/2018-04-08-policy-gradient/, for an accessible tutorial on the material in this section.

This section introduces two landmark algorithms, **Trust Region Policy Optimization (TRPO)** (Schulman et al., 2015) and **Proximal Policy Optimization (PPO)** (Schulman et al., 2017), that address this instability by constraining how much the policy can change in each update step.

### 7.10.1 Recap: The Policy Gradient Landscape

Before diving into modern methods, it is useful to recall the different forms the policy gradient can take:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)\, G_t] \quad \text{, REINFORCE} \tag{7.10.1}$$

$$= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)\, Q^{\mathbf{w}}(s,a)] \quad \text{, Q Actor-Critic} \tag{7.10.2}$$

$$= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)\, A^{\mathbf{w}}(s,a)] \quad \text{, Advantage Actor-Critic} \tag{7.10.3}$$

$$= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)\, \delta] \quad \text{, TD Actor-Critic} \tag{7.10.4}$$

The critic can use Monte Carlo (MC) or temporal-difference (TD) learning to estimate $Q^\pi(s,a)$, $A^\pi(s,a)$, or $V^\pi(s)$.

*Remark* (The common "score-function" skeleton). Every line above has the same shape: the gradient of the log-policy, $\nabla_\theta \log \pi_\theta(a \mid s)$, multiplied by a scalar "weight" (the return $G_t$, the action-value $Q^{\mathbf{w}}$, the advantage $A^{\mathbf{w}}$, or the TD error $\delta$). The term $\nabla_\theta \log \pi_\theta(a \mid s)$ is exactly the *score* familiar from maximum-likelihood estimation, so all four expressions are *likelihood-ratio* (score-function) gradient estimators: they reweight the score by "how good the action turned out to be." Crucially, the four variants share the same expectation and differ only in *which* weight is used—so the choice among them is purely a *variance-reduction* decision. Replacing the raw, high-variance return $G_t$ by a baseline-corrected and bootstrapped quantity (the advantage, or the TD error) lowers the estimator's variance without changing the gradient it is estimating.

### 7.10.2 Issues with Vanilla Policy Gradient

Vanilla policy gradient is **on-policy**, meaning it uses only data collected from the current policy $\pi_\theta$ to compute gradient updates. This has two major consequences:

1. **Poor sample efficiency.** Each batch of data is used for a single gradient update and then discarded. Collecting new data after every update is expensive.

2. **Catastrophic sensitivity to step size.** Unlike supervised learning where data and model are independent, in RL the data distribution depends on the policy. If the step is too large:

- The policy becomes bad → it collects bad data → the next update makes the policy even worse.
- The training may **never recover** from such a collapse.

The solution is to combine **off-policy learning** (via importance sampling) with a **trust region** constraint that limits how much the policy can change. This leads to Trust Region Policy Optimization (TRPO) and its simpler successor, Proximal Policy Optimization (PPO).

### 7.10.3 Relative Policy Performance

The key theoretical insight underlying TRPO begins with analyzing how policy performance changes when we move from one policy to another.

**Steepest ascent in parameter space (standard gradient).**

$$d^* = \nabla_\theta J(\theta) = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \arg\max_d J(\theta + d), \quad \text{s.t. } \|d\| \leq \epsilon \tag{7.10.5}$$

This characterizes the direction of steepest improvement using the **Euclidean metric** on parameter space.

**Steepest ascent in distribution space (natural gradient).**

$$d^* = \arg\max_d J(\theta + d), \quad \text{s.t. } \mathrm{KL}(\pi_\theta \,\|\, \pi_{\theta+d}) = c \tag{7.10.6}$$

This characterizes the direction of steepest improvement using the **KL divergence** to measure distances between policy distributions. The KL divergence is defined as:

$$\mathrm{KL}(\pi_\theta \,\|\, \pi_{\theta'}) = \mathbb{E}_{\pi_\theta}[\log \pi_\theta] - \mathbb{E}_{\pi_\theta}[\log \pi_{\theta'}] \tag{7.10.7}$$

The second-order Taylor expansion of the KL divergence gives:

$$\mathrm{KL}(\pi_\theta \,\|\, \pi_{\theta+d}) \approx \frac{1}{2} d^\top F d \tag{7.10.8}$$

where $F$ is the **Fisher Information Matrix**:

$$F = \mathbb{E}_{\pi_\theta}\left[\nabla \log \pi_\theta \, \nabla \log \pi_\theta^\top\right] \tag{7.10.9}$$

*Remark* (Why the Fisher information appears, and the link to MLE). The matrix $F$ is precisely the **Fisher information matrix** of the policy distribution—the same object an econometrician meets in maximum-likelihood theory, where it equals both

$\mathbb{E}[\nabla \log \pi \nabla \log \pi^\top]$ (the variance of the score) and $-\mathbb{E}[\nabla^2 \log \pi]$ (the expected negative Hessian of the log-likelihood), and where its inverse is the Cramér–Rao lower bound on the variance of an unbiased estimator. Here it plays a *geometric* role: it is the local quadratic form that turns the KL divergence into a squared distance, $\mathrm{KL}(\pi_\theta \,\|\, \pi_{\theta+d}) \approx \frac{1}{2} d^\top F d$. Measuring step size in this KL metric rather than in the raw Euclidean norm $\|d\|$ produces the *natural gradient* $F^{-1}\nabla_\theta J$: a step that is invariant to how the policy happens to be parameterized and that moves a fixed amount in *distribution space*. This is exactly the logic of Fisher scoring in MLE, where one preconditions the score by the inverse information matrix.[37]

**The performance difference between two policies** can be expressed exactly as:

$$J(\pi') - J(\pi) = \mathbb{E}_{\tau\sim\pi'}\left[\sum_{t=0}^{\infty} \gamma^t A^\pi(s_t, a_t)\right] = \frac{1}{1-\gamma}\mathbb{E}_{\substack{s\sim d^{\pi'} \\ a\sim\pi'}}\left[A^\pi(s,a)\right] \tag{7.10.10}$$

where $d^\pi(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t P(s_t = s \mid \pi)$ is the discounted state visitation distribution. This identity says: the improvement of $\pi'$ over $\pi$ equals the expected advantage of $\pi'$'s actions under $\pi$'s value function, weighted by $\pi'$'s state distribution.[38]

### 7.10.4 Importance Sampling for Policy Optimization

The performance difference formula (7.10.10) requires sampling states from the **new** policy $\pi'$, but we only have data from the **old** policy $\pi_{\theta_0}$. Applying importance sampling:

$$\mathcal{J}(\theta) - \mathcal{J}(\theta_0) = \mathbb{E}_{\tau\sim(p_0,\pi_\theta,p)}\left[\sum_{t=0}^{T-1}\gamma^t A^{\pi_{\theta_0}}(s_t, a_t)\right] \tag{7.10.11}$$

Using importance sampling to correct for the distribution mismatch:

$$= \mathbb{E}_{\tau\sim(p_0,\pi_{\theta_0},p)}\left[\sum_{t=0}^{T-1}\gamma^t \frac{\pi_\theta(a'_t \mid s_t)}{\pi_{\theta_0}(a'_t \mid s_t)} A^{\pi_{\theta_0}}(s_t, a'_t)\right] \tag{7.10.12}$$

where, in this estimator, *both* the states and the actions are drawn from the **original** policy $\pi_{\theta_0}$ (the only data we actually have), and the importance weight $\pi_\theta(a'_t \mid s_t)/\pi_{\theta_0}(a'_t \mid s_t)$ re-weights each sampled action so that, in expectation, the action distribution matches the **new** policy $\pi_\theta$. The exact performance-difference identity (7.10.10) actually calls for *states* drawn from the new policy's visitation distribution $d^{\pi_\theta}$; replac-

[37] The natural-gradient idea is due to S.-I. Amari, *Natural gradient works efficiently in learning*, *Neural Computation* 10(2):251–276, 1998.

[38] This is the *performance difference lemma* of S. Kakade and J. Langford, *Approximately optimal approximate reinforcement learning*, ICML 2002. The factor $1/(1-\gamma)$ is exactly the normalizing constant that makes the discounted visitation $d^{\pi'}$ a genuine probability distribution over states.

ing $d^{\pi_\theta}$ by the old policy's $d^{\pi_{\theta_0}}$ is a deliberate approximation that is accurate precisely when $\pi_\theta$ and $\pi_{\theta_0}$ are close—which is exactly what the trust-region constraint below will guarantee.[39]

### 7.10.5 The Surrogate Objective

If the new policy is **sufficiently close** to the original one, we can define a **surrogate objective** that can be estimated entirely from data collected under $\pi_{\theta_0}$:

$$\mathcal{K}(\theta;\theta_0) = \mathbb{E}_{\tau\sim(p_0,\pi_{\theta_0},p)}\left[\sum_{t=0}^{T-1}\gamma^t\frac{\pi_\theta(s_t,a_t)}{\pi_{\theta_0}(s_t,a_t)}A^{\pi_{\theta_0}}(s_t,a_t)\right]+C \tag{7.10.13}$$

Here both **states and actions** are sampled from the **original** policy, making this estimable from collected data. (The additive constant $C = \mathcal{J}(\theta_0)$ does not depend on $\theta$, so it has no effect on the maximizer and is dropped in practice.) The surrogate can be approximated as:

$$\mathcal{L}_{\theta_0}(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T^{(i)}-1}\gamma^t\frac{\pi_\theta(s_t^{(i)},a_t^{(i)})}{\pi_{\theta_0}(s_t^{(i)},a_t^{(i)})}\hat{A}_t^{(i)} \tag{7.10.14}$$

We then maximize this surrogate objective subject to a **trust-region constraint** that keeps $\theta$ close to $\theta_0$:

$$\max_{\theta\in\mathbb{R}^p}\frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T^{(i)}-1}\gamma^t\frac{\pi_\theta(s_t^{(i)},a_t^{(i)})}{\pi_{\theta_0}(s_t^{(i)},a_t^{(i)})}\hat{A}_t^{(i)}, \quad \text{subject to } \theta \text{ and } \theta_0 \text{ close} \tag{7.10.15}$$

### 7.10.6 Trust Region Policy Optimization (TRPO)

**TRPO** (Schulman et al., 2015) operationalizes the surrogate objective by constraining policy updates to stay within a **trust region** defined by KL divergence:

$$\mathrm{KL}\big(\pi_{\theta_{\text{old}}}(\cdot\mid s_t)\,\|\,\pi_\theta(\cdot\mid s_t)\big)\le\delta \tag{7.10.16}$$

### 7.10.7 Limitations of TRPO

While theoretically elegant, TRPO has significant **scalability issues**:

[39] This is the importance-sampling, or *change-of-measure*, trick familiar from Monte Carlo simulation and from inverse-probability weighting in causal inference: to estimate an expectation under a target distribution $p$ using draws from a proposal $q$, reweight each draw by the likelihood ratio $p/q$ (here $\pi_\theta/\pi_{\theta_0}$). As with inverse-probability weights, the estimator is unbiased but its variance explodes when the ratio is large, i.e. when the two policies disagree strongly—a further reason to keep them close.

**Algorithm 17** Trust Region Policy Optimization (TRPO)

**Input:** initial policy parameters $\theta_0$
**for** $k = 0, 1, 2, \ldots$ **do**
  Collect set of trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
  Estimate advantages $\hat{A}$ using any advantage estimation algorithm
  Form sample estimates for: the policy gradient $\hat{g}_k$ (using advantage estimates) and the KL-divergence Hessian-vector product function $f(v) = \hat{H}_k v$
  Use conjugate gradient (CG) with $n_{\text{cg}}$ iterations to obtain $x_k \approx \hat{H}_k^{-1} \hat{g}_k$
  Estimate proposed step $\Delta_k \approx \sqrt{2\delta / (x_k^\top \hat{H}_k x_k)} \cdot x_k$
  Perform backtracking line search with exponential decay to obtain the final update: $\theta_{k+1} = \theta_k + \alpha^j \Delta_k$
**end for**

1. **Expensive Fisher Information Matrix computation.** Computing the Fisher Information Matrix $H$ for the current policy requires a large batch of rollouts:

$$H = \nabla_\theta^2 \, \mathrm{KL}(\pi_{\theta_t} \,\|\, \pi_\theta) = \mathbb{E}_{a,s \sim \pi_{\theta_t}} \left[ \nabla_\theta \log \pi_\theta(a,s) \, \nabla_\theta \log \pi_\theta(a,s)^\top \right] \tag{7.10.17}$$

2. **Conjugate gradient solver.** Determining the step size involves an approximate Newton method, solving $H^{-1} g$ via a conjugate gradient (CG) solver, a second-order method that is complex and computationally expensive.

Using a first-order Taylor approximation of the objective and a second-order approximation of the KL constraint, the update becomes:

$$\theta_{t+1} = \arg\max_\theta \, g^\top (\theta - \theta_t) \quad \text{s.t. } \tfrac{1}{2} (\theta - \theta_t)^\top H (\theta - \theta_t) \leq \delta \tag{7.10.18}$$

This can be solved analytically:

$$\theta_{t+1} = \theta_t + \sqrt{\frac{2\delta}{g^\top H^{-1} g}} \, H^{-1} g \tag{7.10.19}$$

### 7.10.8 Proximal Policy Optimization (PPO)

**PPO** (Schulman et al., 2017) achieves performance comparable to TRPO while being **much simpler to implement**. Instead of solving a constrained optimization problem with second-order methods, PPO uses first-order methods (SGD/Adam) with a modified objective.

**PPO-Penalty.** The trust region constraint is reformulated as an **unconstrained optimization** with KL divergence as a regularizer:

$$\max_{\theta} \mathbb{E}_t \left[ \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} A_t \right] - \beta \, \mathbb{E}_t \left[ \mathrm{KL}\left[ \pi_{\theta_{\text{old}}}(\cdot \mid s_t), \, \pi_\theta(\cdot \mid s_t) \right] \right] \tag{7.10.20}$$

The penalty coefficient $\beta$ is **adaptive** between iterations to approximately enforce the KL-divergence constraint:

- If $\bar{D}_{\mathrm{KL}}(\theta_{k+1} \,\|\, \theta_k) \geq 1.5\delta$: $\beta_{k+1} = 2\beta_k$ (policy moved too far, increase penalty).
- If $\bar{D}_{\mathrm{KL}}(\theta_{k+1} \,\|\, \theta_k) \leq \delta/1.5$: $\beta_{k+1} = \beta_k/2$ (policy moved too little, decrease penalty).

**Algorithm 18** PPO with Adaptive KL Penalty

**Input:** initial policy parameters $\theta_0$, initial KL penalty $\beta_0$, target KL-divergence $\delta$
**for** $k = 0, 1, 2, \ldots$ **do**
  Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
  Estimate advantages $\hat{A}$ using any advantage estimation algorithm
  Compute policy update

$$\theta_{k+1} = \arg\max_{\theta} \, L_{\theta_k}(\theta) - \beta_k \, \bar{D}_{\mathrm{KL}}(\theta \,\|\, \theta_k)$$

  by taking $K$ steps of minibatch SGD (via Adam)
  **if** $\bar{D}_{\mathrm{KL}}(\theta_{k+1} \,\|\, \theta_k) \geq 1.5\delta$ **then**
    $\beta_{k+1} = 2\beta_k$
  **else if** $\bar{D}_{\mathrm{KL}}(\theta_{k+1} \,\|\, \theta_k) \leq \delta/1.5$ **then**
    $\beta_{k+1} = \beta_k/2$
  **end if**
**end for**

### 7.10.9 PPO-Clip

The most widely used variant of PPO is **PPO-Clip**, which avoids computing KL divergence entirely by using a **clipped surrogate objective**. Define the probability ratio:

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_k}(a_t \mid s_t)} \tag{7.10.21}$$

Note that $r_t(\theta)$ is close to 1 when the new policy is close to the original one. The clipped objective is:

$$\mathcal{L}^{\text{CLIP}}_{\theta_k}(\theta) = \mathbb{E}_{\tau \sim \pi_k} \left[ \sum_{t=0}^{T} \min\left( r_t(\theta) \, \hat{A}_t^{\pi_k}, \; \text{clip}\left( r_t(\theta), 1-\epsilon, 1+\epsilon \right) \hat{A}_t^{\pi_k} \right) \right] \tag{7.10.22}$$

where $\epsilon$ is a hyperparameter (typically $\epsilon = 0.2$). The policy update is simply:

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}^{\text{CLIP}}_{\theta_k}(\theta) \tag{7.10.23}$$

**How clipping works.**

- When the advantage $A > 0$ (good action): the objective increases with $r_t(\theta)$, but clipping at $1 + \epsilon$ prevents the ratio from growing too large.
- When the advantage $A < 0$ (bad action): the objective increases as $r_t(\theta)$ decreases, but clipping at $1 - \epsilon$ prevents the ratio from shrinking too much.

The final clipped objective is a **lower (pessimistic) bound** of the unclipped objective, which means it errs on the side of caution. The clipping removes the incentive for the policy to move far from $\theta_k$, providing a simple mechanism for trust-region enforcement without any KL divergence computation.

*Remark* (Clipping as variance control, and the trust region as a proximal step). Two connections make PPO-Clip intuitive for a reader trained in estimation and optimization. First, the probability ratio $r_t(\theta) = \pi_\theta(a_t \mid s_t)/\pi_{\theta_k}(a_t \mid s_t)$ is exactly an importance weight, and $\text{clip}(\cdot, 1-\epsilon, 1+\epsilon)$ *caps* that weight—the same defensive move as *winsorizing* or *trimming* extreme inverse-probability weights to keep an estimator's variance under control. Second, TRPO and both PPO variants are all instances of *proximal* optimization: improve the objective while discouraging moves far from the current iterate $\theta_k$. TRPO imposes a hard KL constraint (a literal trust region, solved with a Lagrangian/KKT argument); PPO-Penalty turns that constraint into a soft KL penalty (its Lagrangian relaxation); and PPO-Clip enforces it implicitly, by zeroing out the gradient signal once the ratio leaves $[1-\epsilon, 1+\epsilon]$ in the "wrong" direction.

**Key advantages of PPO-Clip.**

- PPO methods have the **stability and reliability of TRPO** but are **much simpler to implement**.
- They require only first-order optimization (SGD or Adam).
- There is no need to compute KL divergence or the Fisher Information Matrix.
- They achieve comparable or better performance than TRPO on standard benchmarks.

**Algorithm 19** PPO with Clipped Objective (PPO-Clip)

1: **Input:** initial policy parameters $\theta_0$, clipping threshold $\epsilon$
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
4: Estimate advantages $\hat{A}$ using any advantage estimation algorithm
5: Compute policy update

$$\theta_{k+1} = \arg\max_{\theta} L^{\text{CLIP}}_{\theta_k}(\theta)$$

by taking $K$ steps of minibatch SGD (via Adam), where

$$L^{\text{CLIP}}_{\theta_k}(\theta) = \mathbb{E}_{\tau\sim\pi_k}\left[\sum_t \min\left(r_t(\theta)\,\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\,\hat{A}_t\right)\right]$$

6: **end for**

### 7.10.10 PPO: Implementation Challenges

Despite its apparent simplicity, PPO implementation is **tricky and challenging** in practice. Hsu et al. (2020) and Engstrom et al. (2020) highlight several important caveats:

1. **Failure modes of standard PPO.**
   - On continuous action spaces, standard PPO is unstable when rewards vanish outside bounded support.
   - On discrete action spaces with sparse high rewards, standard PPO often gets stuck at suboptimal actions.
   - The policy is sensitive to initialization when there are locally optimal actions close to initialization.
2. **Implementation details matter enormously.** "Code-level optimizations" that are described as auxiliary details in implementations turn out to have a **major impact on agent behavior**. These optimizations are responsible for most of PPO's gain in cumulative reward over TRPO, fundamentally changing how the RL methods function.

These findings serve as a reminder that many algorithmic design choices in deep RL are tied to specific simulation environments, and standard design choices should not be implicitly accepted as universal defaults.

## 7.11 Reinforcement Learning for Large Language Models

One of the most impactful applications of modern deep RL, and of PPO in particular, is in the **post-training of large language models (LLMs)**. Reinforcement learning from human feedback (RLHF) has become a central technique for aligning LLMs with human preferences, addressing issues of safety, helpfulness, and instruction-following that supervised training alone cannot fully resolve (Ouyang et al., 2022; Ziegler et al., 2019).[40]

### 7.11.1 The LLM Training Pipeline

Modern LLMs are trained in two major phases.

**Pretraining.**

- Very large models are trained via **unsupervised learning** on a gigantic web-scale dataset of texts and documents, essentially the entire archive of human written knowledge.
- Key enablers: **GPUs** for fast computation, **data** freely available from the Internet, the **Transformer** architecture, and substantial **financial investment**.
- The output is a **base LLM** that can generate fluent text but may not follow instructions well, may produce harmful content, or may hallucinate.

**Post-training.**

- The base LLM is refined for specific downstream tasks through **supervised fine-tuning** and **reinforcement learning**.
- The goal is to slightly adjust the pre-trained model for subsequent tasks, particularly to address **alignment** and **safety** issues.
- RLHF has become the dominant technique for this alignment phase (Ouyang et al., 2022).

[40] See Hugging Face (2022), *Illustrating Reinforcement Learning from Human Feedback (RLHF)*, https://huggingface.co/blog/rlhf, for an accessible illustrated tutorial, and DeepLearning.AI (2023), *Reinforcement Learning from Human Feedback*, https://learn.deeplearning.ai/courses/reinforcement-learning-from-human-feedback, for a hands-on short course.

### 7.11.2 Post-training Components

The post-training pipeline consists of several components, which can be organized into two tracks (see Figure 7.11.1).

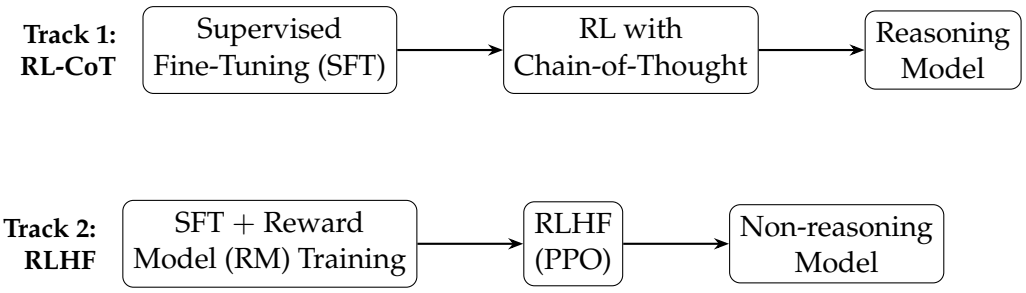


**Figure 7.11.1.** The two tracks of LLM post-training: a reasoning track (RL with chain-of-thought) and a non-reasoning track (RLHF).

The four key components are:

1. **Supervised Fine-Tuning (SFT):** behavioral cloning of human or expert behaviors. The model learns to mimic high-quality demonstrations.
2. **Reward Model (RM) Training:** learning a model of human preferences from comparative judgments.
3. **RLHF:** optimizing the fine-tuned LLM against the reward model using RL (typically PPO).
4. **RL without RM:** reasoning with (long) Chains-of-Thought (CoTs), enabling test-time scaling without an explicit reward model.

### 7.11.3 Reward Modeling

**Motivation.** Supervised fine-tuning has limitations:

- Open-ended questions lack a single correct answer.
- Some token prediction errors are more serious than others.
- It is expensive to create high-quality demonstration data.

**Approach.** Instead of demonstrating the correct answer, human labelers are asked to **rank** $K$ **LLM-generated responses** to a prompt. This comparative judgment is much easier and cheaper than writing ideal responses.

The reward model $r_\theta$ is trained using the **Bradley-Terry model** (a classic method of paired comparisons) with the following loss function:

$$\text{loss}(\theta) = -\frac{1}{\binom{K}{2}} \mathbb{E}_{(x,y_w,y_l)\sim D} \left[\log \sigma\big(r_\theta(x,y_w) - r_\theta(x,y_l)\big)\right] \tag{7.11.1}$$

where $x$ is the prompt, $y_w$ is the winning (preferred) response, $y_l$ is the losing response, and $\sigma$ is the sigmoid function.

*Remark* (The reward model is a discrete-choice / logit model). Equation (7.11.1) should look familiar: the **Bradley–Terry** model of paired comparisons is the same object as the **binary logit** (Luce) model of discrete choice. Read $r_\theta(x,y)$ as the *latent utility* a rater assigns to response $y$ in context $x$. Under a random-utility model with i.i.d. Gumbel (type-I extreme-value) errors—McFadden's classic setup—the probability that $y_w$ is preferred to $y_l$ is the logistic transform of the utility difference,

$$\Pr(y_w \succ y_l) = \sigma\big(r_\theta(x,y_w) - r_\theta(x,y_l)\big) = \frac{1}{1+\exp\big(-\big(r_\theta(x,y_w) - r_\theta(x,y_l)\big)\big)}.$$

Minimizing the loss in (7.11.1) is therefore nothing more than **maximum-likelihood estimation of a conditional logit** on pairwise-comparison data, with the neural network supplying the utility index $r_\theta(x,y)$ in place of a linear index $x^\top\beta$. Only utility *differences* enter, so the reward scale carries an arbitrary additive constant—the same location normalization one imposes in discrete-choice estimation. The factor $1/\binom{K}{2}$ simply averages over the $\binom{K}{2}$ pairwise comparisons that a single ranking of $K$ responses generates.

**Key properties of reward models.**

- The RM is typically a "small" language model (e.g., GPT-3 with 6B parameters).
- RMs help **generalize LLM evaluations to difficult-to-verify tasks**, where correctness is hard to define.
- RMs **save huge costs** compared to recruiting human labelers for every evaluation.
- However, RMs are subject to **reward hacking**, the LLM may learn to exploit weaknesses in the RM to achieve high reward scores without actually improving output quality.[41]

[41] See L. Weng (2024), *Reward Hacking in Reinforcement Learning*, Lil'Log, https://lilianweng.github.io/posts/2024-11-28-reward-hacking/.

### 7.11.4 The MDP Formulation of RLHF

RLHF finetunes the LLM $\pi$ further so that the completion achieves high reward as measured by the reward model $r(\cdot)$. To apply RL, we must first formulate the problem as an MDP.[42]

**What is the MDP in RLHF?**

- LLMs are **autoregressive** models, so they are not Markovian but history-dependent: the policy is not of the form $\pi(a_t \mid s_t)$ but rather $\pi(a_t \mid s_1, s_2, \ldots, s_t)$.
- Therefore, the **state** should be the user prompt and the tokens generated so far: $(u_1, u_2, \ldots, u_l)$.
- Each **action** is the generation of one token. The policy is randomized (sampling from the distribution over tokens), but the transition dynamics is **deterministic** (appending the chosen token to the sequence).

**The RL setup.**

- Each timestep is a **BPE token**.
- The LLM $\pi_\theta(u_{l+1} \mid u_1, \ldots, u_l)$ is our policy mapping the current state (token sequence) to a distribution on the action (next token) $u_{l+1}$.
- Response generation is an **episode**, and an episode terminates when the LM generates `<EOS>`.
- **No discount** is used, i.e., discount factor $\gamma = 1$.
- Reward (by reward model $r_\psi$) is only provided at the **end of the episode**. There are no intermediate rewards. This is called the **"contextual bandit" setting**.
- Sampling temperature $\beta = 1$.[43]

*Remark* ("Contextual bandit" in econometric language). Calling RLHF a *contextual bandit* just means the reward arrives only once—at the end of the generated sequence—given the "context" (the prompt). This is the structure of a one-shot *sequential experiment* or *A/B test*: the policy chooses an action (here, an entire completion), observes

[42] This MDP formulation follows E. K. Ryu (2025), *RL for LLM*, Course Notes, Stanford University, https://ernestryu.com/courses/RL-LLM/chapter1.pdf.

[43] This sampling temperature $\beta$ is a *different* symbol from the KL-penalty coefficient $\beta$ introduced later in Section 7.11.8; the notation is unfortunately overloaded in the RLHF literature. Temperature rescales the logits before the softmax that defines the next-token distribution, and $\beta = 1$ means sampling from the model's unmodified predictive distribution.

a single scalar payoff (the reward-model score), and updates. There is no within-episode dynamic-programming credit-assignment problem beyond attributing the final reward to the tokens that produced it; the horizon enters only because the "action" is itself a long sequence of token-level choices.

### 7.11.5 PPO for RLHF

Let $\pi_\theta(u_{l+1} \mid u_1, \ldots, u_l)$ be our LLM and the RL policy. Let $x$ be a text prompt and $y = y_{1:T+1}$ be its completion by $\pi_\theta$ (so $y_{T+1} =$ <EOS>). Let $y_{1:t}$ denote the partial completion up to token $t$. The PPO-Clip ratio is set to $\varepsilon = 0.2$.

PPO maintains a **value function model** $V_\phi(x, y_{1:t})$: given the prompt and partial completion $(x, y_{1:t})$, what is the expected reward if we continue generation with $\pi_\theta$?

The **advantage** is estimated as:

$$\hat{A} = r_\psi(x, y_{1:T+1}) - V_\phi(x, y_{1:t}) \tag{7.11.2}$$

This measures: how good is the total completion $y_{t+1:T+1}$ compared to what $V_\phi$ was expecting based on $y_{1:t}$?

- If $\hat{A} = r_\psi(x, y_{1:T+1}) - V_\phi(x, y_{1:t}) > 0$, then $y_{t+1:T+1}$ was a **good completion**. We should adjust $\pi_\theta$ to make those actions more likely.
- If $\hat{A} = r_\psi(x, y_{1:T+1}) - V_\phi(x, y_{1:t}) < 0$, then $y_{t+1:T+1}$ was a **bad completion**. We should adjust $\pi_\theta$ to make those actions less likely.

(In practice, Generalized Advantage Estimation (GAE) is used for $\hat{A}$, but the simpler advantage estimate above conveys the key intuition.)

### 7.11.6 PPO v0: Susceptibility to Reward Hacking

The basic PPO algorithm for RLHF (PPO v0) proceeds as in Algorithm 20.

Here $C_\varepsilon(\ell, A) = \min\left(\ell A, \text{clip}(\ell, 1-\varepsilon, 1+\varepsilon)\, A\right)$ denotes the PPO-Clip operator. However, this basic version is **susceptible to reward hacking**.

### 7.11.7 Reward Hacking and Goodhart's Law

**Goodhart's Law** states: *"When a measure becomes a target, it ceases to be a good measure."*[44] This principle is directly relevant to RLHF:

[44] This now-standard pithy formulation is in fact due to the anthropologist M. Strathern, *'Improving ratings': audit in the British University system*, *European Review* 5(3):305–321, 1997; C. Goodhart's original 1975 statement concerned monetary-policy targets. The economic content—optimizing hard against a proxy degrades the proxy as a measure—is the same phenomenon as *overfitting* a validation metric.

**Algorithm 20** PPO v0 for RLHF (susceptible to reward hacking)

1: **while** not converged **do**

2: Sample $N$ trajectories $(x^{(i)}, y^{(i)}_{1:T^{(i)}+1}) \sim (p_0, \pi^{\text{RL}}_{\theta_{\text{curr}}})$ for $i = 1, \ldots, N$

3: Compute advantages: $\hat{A}^{(i)}_t = r_\psi(x^{(i)}, y^{(i)}_{1:T^{(i)}+1}) - V_\phi(x^{(i)}, y^{(i)}_{1:t})$ for all $i, t$

4: Solve (policy update):

$$\max_{\theta_{\text{next}}} \sum_i \sum_t C_\varepsilon \left( \frac{\pi^{\text{RL}}_{\theta_{\text{next}}}(y^{(i)}_{t+1} \mid x^{(i)}, y^{(i)}_{1:t})}{\pi^{\text{RL}}_{\theta_{\text{curr}}}(y^{(i)}_{t+1} \mid x^{(i)}, y^{(i)}_{1:t})}, \ \hat{A}^{(i)}_t \right)$$

5: Set $\theta_{\text{curr}} = \theta_{\text{next}}$

6: Solve (value function update):

$$\min_\phi \frac{1}{N} \sum_i \frac{1}{T^{(i)}} \sum_t \tfrac{1}{2} \left( r_\psi(x^{(i)}, y^{(i)}_{1:T^{(i)}+1}) - V_\phi(x^{(i)}, y^{(i)}_{1:t}) \right)^2$$

7: **end while**

- The reward model is **imperfect**. Maximizing an imperfect reward model too aggressively will exploit the model's imperfections and result in **adversarial generations**.
- The reward model is trained on the SFT LLM $\pi_{\text{SFT}}$. Therefore, the reward model is only informative about responses generated by the RL policy if $\pi_{\text{RL}}$ is **close** to $\pi_{\text{SFT}}$.
- Moving away from $\pi_{\text{SFT}}$ too much will cause the language model to **lose its main capabilities**. Fine-tuning too much can break the model, causing it to output nonsense tokens.

**Solution:** impose a **KL-divergence penalty term**, ensuring that $\pi_{\text{RL}}$ stays close to $\pi_{\text{SFT}}$.

### 7.11.8 KL-Penalty and Pre-training Loss

RLHF with KL-penalty maximizes the following objective:

$$\mathcal{J}(\theta) = \mathbb{E}_{(x,y)\sim D_{\pi^{\text{RL}}_\theta}} \left[ r_\psi(x, y) \right] - \beta \, \mathbb{E}_{x \sim \mathcal{D}} \left[ D_{\text{KL}} \left( \pi^{\text{RL}}_\theta(\cdot \mid x) \,\|\, \pi^{\text{SFT}}(\cdot \mid x) \right) \right] \tag{7.11.3}$$

The KL divergence can be decomposed at the token level:

$$\mathcal{J}(\theta) = \mathbb{E}_{(x,y)\sim D_{\pi_\theta^{\mathrm{RL}}}}\left[r_\psi(x,y) - \beta \log \frac{\pi_\theta^{\mathrm{RL}}(y \mid x)}{\pi^{\mathrm{SFT}}(y \mid x)}\right] \tag{7.11.4}$$

$$= \mathbb{E}_{(x,y)\sim D_{\pi_\theta^{\mathrm{RL}}}}\left[r_\psi(x,y) - \beta \sum_{t=0}^{T} \log \frac{\pi_\theta^{\mathrm{RL}}(y_{t+1} \mid x, y_{1:t})}{\pi^{\mathrm{SFT}}(y_{t+1} \mid x, y_{1:t})}\right] \tag{7.11.5}$$

The KL-penalty encourages $\pi_{\mathrm{RL}}$ to stay **close** to $\pi_{\mathrm{SFT}}$.[45]

*Remark* (The KL penalty is regularization toward a reference model). The term $-\beta\,\mathbb{E}[D_{\mathrm{KL}}(\pi_\theta^{\mathrm{RL}} \,\|\, \pi^{\mathrm{SFT}})]$ is a *shrinkage* penalty, directly analogous to a ridge/Lasso penalty or to a Bayesian prior centered at the SFT model. Maximizing reward pulls the policy toward whatever the (imperfect) reward model happens to favor; the KL term pulls it back toward the trusted reference $\pi^{\mathrm{SFT}}$. The coefficient $\beta$ is the usual regularization knob: a large $\beta$ keeps the policy close to $\pi^{\mathrm{SFT}}$ (less reward hacking, smaller behavioral drift), while a small $\beta$ lets it chase reward more aggressively (more hacking risk). The KL divergence—an expected log-likelihood ratio between two token distributions—is the natural notion of "distance" here precisely because both objects are probability models over the same outputs.

**Absorbing the KL penalty into the MDP.** Maximizing $\mathcal{J}(\theta)$ is equivalent to solving RL on an MDP with the same transition dynamics but **modified rewards** $r_0, r_1, \ldots, r_T$:

$$r_t = -\beta \log \frac{\pi_\theta^{\mathrm{RL}}(y_{t+1} \mid x, y_{1:t})}{\pi^{\mathrm{SFT}}(y_{t+1} \mid x, y_{1:t})}, \quad t = 0, \ldots, T-1 \tag{7.11.6}$$

$$r_T = r_\psi(x, y_{1:T+1}) - \beta \log \frac{\pi_\theta^{\mathrm{RL}}(y_{T+1} \mid x, y_{1:T})}{\pi^{\mathrm{SFT}}(y_{T+1} \mid x, y_{1:T})} \tag{7.11.7}$$

The modified MDP has **immediate rewards** (at every token) which absorb the KL penalty. The reward depends on the parameter $\theta$, but all the math goes through.

**Adding the pre-training loss.** Even with the KL-penalty, the base language model capability can be compromised. Adding a **pre-training loss** term preserves general language abilities:

$$\mathcal{J}(\theta) = \mathbb{E}_{(x,y)\sim D_{\pi_\theta^{\mathrm{RL}}}}\left[r_\psi(x,y) - \beta \log\left(\pi_\theta^{\mathrm{RL}}(y \mid x)/\pi^{\mathrm{SFT}}(y \mid x)\right)\right] + \gamma\,\mathbb{E}_{x\sim D_{\mathrm{pretrain}}}\left[\log\left(\pi_\theta^{\mathrm{RL}}(x)\right)\right] \tag{7.11.8}$$

[45] See J. Schulman (2020), *Approximating KL Divergence*, http://joschu.net/blog/kl-approx.html, for practical considerations on approximating KL divergence in this setting.

where $\gamma > 0$ and the last term is the next-token prediction loss used in pre-training. PPO and pre-training updates are performed **simultaneously or in alternating fashion**.

*Remark* (Connection to PPO). The KL penalty in the RLHF objective plays exactly the same role as the trust region constraint in PPO, preventing the policy from moving too far from its starting point. This is why PPO (especially PPO-Penalty) is a natural fit for RLHF: the KL divergence between the RL policy and the SFT policy is precisely the trust-region constraint that PPO enforces.

### 7.11.9 Full PPO Algorithm for RLHF

The complete PPO algorithm with KL-penalty for RLHF is given in Algorithm 21.

**Algorithm 21** Full PPO Algorithm for RLHF (with KL penalty)

1: **while** not converged **do**

2: Sample $N$ trajectories $(x^{(i)}, y^{(i)}_{1:T^{(i)}+1}) \sim (p_0, \pi^{\mathrm{RL}}_{\theta_{\mathrm{curr}}})$ for $i = 1, \ldots, N$

3: Compute advantages with modified rewards:

$$\hat{A}^{(i)}_t = r_\psi(x^{(i)}, y^{(i)}_{1:T^{(i)}+1}) - \beta \sum_s \log \frac{\pi^{\mathrm{RL}}_{\theta_{\mathrm{next}}}(y^{(i)}_{s+1} \mid x^{(i)}, y^{(i)}_{1:s})}{\pi^{\mathrm{SFT}}(y^{(i)}_{s+1} \mid x^{(i)}, y^{(i)}_{1:s})} - V_\phi(x^{(i)}, y^{(i)}_{1:t})$$

4: Solve (policy update with PPO-Clip):

$$\max_{\theta_{\mathrm{next}}} \sum_i \sum_t C_\varepsilon\left(\frac{\pi^{\mathrm{RL}}_{\theta_{\mathrm{next}}}}{\pi^{\mathrm{RL}}_{\theta_{\mathrm{curr}}}}, \hat{A}^{(i)}_t\right), \quad C_\varepsilon(\ell, A) = \min\big(\ell A, \mathrm{clip}(\ell, 1-\varepsilon, 1+\varepsilon)\, A\big)$$

5: Set $\theta_{\mathrm{curr}} = \theta_{\mathrm{next}}$

6: Solve (value function update):

$$\min_\phi \frac{1}{N} \sum_i \frac{1}{T^{(i)}} \sum_t \tfrac{1}{2}\big(\hat{G}^{(i)}_t - V_\phi(x^{(i)}, y^{(i)}_{1:t})\big)^2$$

7: **end while**

### 7.11.10 Empirical Performance of RLHF

The effectiveness of RLHF has been demonstrated in landmark studies.

**Learning to Summarize (Stiennon et al., 2020).** Models trained with human feedback significantly outperform both pretrain-only and supervised learning baselines

on summarization tasks. Importantly, RLHF performance improves with model size, and human feedback provides gains that scale better than supervised learning alone.

**InstructGPT (Ouyang et al., 2022).** RLHF-trained models (PPO and PPO-ptx variants) consistently outperform SFT models on both the GPT and Instruct distributions. The improvements are measured by win rates against reference summaries, evaluated by both held-out workers and training workers. GPT models fine-tuned with PPO achieve the highest preference rates across model sizes from 1.3B to 175B parameters.

### 7.11.11 RLHF Can Be (Too) Complex

Despite its success, the full RLHF pipeline is **computationally expensive and tricky** to implement (Zheng et al., 2023):

- **RL optimization is expensive:** the pipeline requires maintaining and coordinating multiple models (policy LM, reference SFT model, reward model, value model/critic).
- **Value function fitting is required:** a separate value network must be trained alongside the policy.
- **Online sampling is slow:** generating completions from the current policy at each iteration is a major bottleneck.
- **Performance is highly sensitive to hyperparameters**, including the KL penalty coefficient $\beta$, learning rates, and clipping thresholds.

These complexities motivate the search for simpler alternatives.

### 7.11.12 Direct Preference Optimization (DPO)

**Direct Preference Optimization (DPO)** (Rafailov et al., 2023) elegantly sidesteps the complexity of RLHF by observing that the RL problem has a **closed-form solution**.

**Standard RLHF** requires two steps:

1. Fit the reward model based on human preference data.
2. Optimize the instruction-finetuned LLM given the reward function (via PPO).

**DPO's key insight.** There is another perspective:

1. Derive the reward model based on an RL policy, parameterized by $\theta$.
2. Optimize the parameter $\theta$ by **fitting the reward model to the true human preference data**, instead of fitting a given reward model.

**Closed-form solution for KL-regularized RL.** Ignoring the pre-training loss, the KL-regularized RL objective for any reward model $r$ is:

$$\max_{\theta} \mathcal{J}(\pi_\theta; r) = \mathbb{E}_{\substack{x\sim\mathcal{D}\\ y\sim\pi_\theta}}\left[r(x,y) - \beta \log \frac{\pi_\theta(y \mid x)}{\pi^{\text{SFT}}(y \mid x)}\right] \tag{7.11.9}$$

Transforming this as a loss and completing the algebra:

$$-\frac{1}{\beta}\mathcal{J}(\pi_\theta; r) = \mathbb{E}_{\substack{x\sim\mathcal{D}\\ y\sim\pi_\theta}}\left[\log \frac{\pi_\theta(y \mid x)}{\pi^{\text{SFT}}(y \mid x)} - \frac{1}{\beta} r(x,y)\right] \tag{7.11.10}$$

$$= \mathbb{E}_{x\sim\mathcal{D}}\left[D_{\text{KL}}\big(\pi_\theta(\cdot \mid x) \,\|\, \pi_r(\cdot \mid x)\big)\right] - \log Z(x) \tag{7.11.11}$$

where the **optimal policy** $\pi_r$ is:

$$\pi_r(y \mid x) = \frac{\pi^{\text{SFT}}(y \mid x)\exp\big(\frac{1}{\beta} r(x,y)\big)}{Z(x)}, \quad Z(x) = \sum_y \pi^{\text{SFT}}(y \mid x)\exp\big(\tfrac{1}{\beta} r(x,y)\big) \tag{7.11.12}$$

Therefore, maximizing $\mathcal{J}(\pi_\theta; r)$ over $\theta$ is equivalent to minimizing $D_{\text{KL}}\big(\pi_\theta(\cdot \mid x) \,\|\, \pi_r(\cdot \mid x)\big)$, i.e., **the optimal policy is** $\pi_\theta = \pi_r$.

*Remark* (The optimal policy is an exponential tilt of the reference). Equation (7.11.12) is the familiar *Gibbs/Boltzmann* (exponential-family) form: the KL-regularized optimum reweights the reference distribution $\pi^{\text{SFT}}$ by the factor $\exp\big(r(x,y)/\beta\big)$ and renormalizes by the partition function $Z(x)$. This is exactly *exponential tilting* of a base measure—the construction behind exponential-family models and Esscher transforms. Two sanity checks: as $\beta \to \infty$ (heavy regularization) the tilt vanishes and $\pi_r \to \pi^{\text{SFT}}$; as $\beta \to 0$ (no regularization) all mass concentrates on the highest-reward response. The catch is that $Z(x) = \sum_y \pi^{\text{SFT}}(y \mid x)\exp\big(r(x,y)/\beta\big)$ sums over *all* possible completions $y$ and is therefore astronomically expensive to evaluate—exactly the obstacle DPO is about to sidestep.

**Inversely** , the reward function that makes any policy $\pi$ optimal is:

$$r_\pi(x,y) = \beta \log \frac{\pi(y \mid x)}{\pi^{\text{SFT}}(y \mid x)} + \beta \log Z(x) \tag{7.11.13}$$

### 7.11.13 DPO: The Final Objective

Recall that the reward model was trained via the Bradley-Terry (BT) method:

$$\min_{\psi} \sum_{\substack{(x,y_i,y_j)\in\mathcal{D}\\ y_i \succ y_j}} -\log \sigma\big(r_\psi(x,y_i) - r_\psi(x,y_j)\big) \tag{7.11.14}$$

DPO substitutes the closed-form reward $r_\pi$ from (7.11.13) into this loss. Since $\beta \log Z(x)$ cancels in the difference $r_\pi(x, y_i) - r_\pi(x, y_j)$, the DPO loss becomes:

$$\mathcal{L}^{\text{DPO}}(\theta) = \sum_{\substack{(x,y_i,y_j)\in\mathcal{D} \\ y_i \succ y_j}} -\log \sigma\left(\beta \log \frac{\pi_\theta(y_i \mid x)}{\pi^{\text{SFT}}(y_i \mid x)} - \beta \log \frac{\pi_\theta(y_j \mid x)}{\pi^{\text{SFT}}(y_j \mid x)}\right) \tag{7.11.15}$$

*Remark* (DPO is maximum likelihood for a logit model in disguise). The cancellation is the crucial trick: because the intractable term $\beta \log Z(x)$ enters $r_\pi(x, y_i)$ and $r_\pi(x, y_j)$ *identically*, it disappears from their difference, so the partition function never has to be computed. What remains, (7.11.15), is once again a **Bradley–Terry / binary-logit log-likelihood**—identical in form to the reward-model loss (7.11.1)—except that the latent "utility" of a response is now the implicit reward $\hat{r}_\theta(x, y) = \beta \log\big(\pi_\theta(y \mid x)/\pi^{\text{SFT}}(y \mid x)\big)$, read directly off the *policy* rather than off a separate network. DPO thus collapses the two-stage RLHF pipeline (fit a reward model, then optimize against it with PPO) into a *single* maximum-likelihood fit on the preference data—a standard discrete-choice estimation that runs with off-the-shelf gradient optimizers.

**Key advantages of DPO.**

- There is **no need to train a reward model** and no need to train a value network.
- DPO effectively converts the RL problem into a **supervised learning problem**, so it is **much easier to execute**.

**The DPO gradient** is:

$$\nabla_\theta \mathcal{L}^{\text{DPO}}(\theta) = -\beta \sum_{\substack{(x,y_i,y_j)\in\mathcal{D} \\ y_i \succ y_j}} \sigma\big(-(r_{\pi_\theta}(x, y_i) - r_{\pi_\theta}(x, y_j))\big) \times \big(\nabla_\theta \log \pi_\theta(y_i \mid x) - \nabla_\theta \log \pi_\theta(y_j \mid x)\big) \tag{7.11.16}$$

**Interpretation.** DPO is doing **ascent on the good outcome** $y_i$ (increasing its likelihood) while doing **descent on the bad completion** $y_j$ (decreasing its likelihood). The gradient is accentuated when the implicitly defined reward model $r_{\pi_\theta}$ **disagrees** with the human preference $y_i \succ y_j$, this is where the model needs the most correction.

*Remark* (The same "residual × regressor" structure as logistic regression). The DPO gradient has the canonical form of a logistic-regression score equation. The scalar weight $\sigma\big(-(r_{\pi_\theta}(x, y_i) - r_{\pi_\theta}(x, y_j))\big) = \sigma(\hat{r}_j - \hat{r}_i)$ is the model's currently predicted probability of getting the comparison *wrong*; it plays the role of the *residual* (observed minus predicted preference) in a binary-logit score. The vector $\nabla_\theta \log \pi_\theta(y_i \mid x) -$

$\nabla_\theta \log \pi_\theta(y_j \mid x)$ plays the role of the *regressors*. When the model already ranks the pair correctly and confidently, the weight is near zero and almost no update occurs; when it is confidently wrong, the weight is near one and the update is large—exactly the behavior of maximum-likelihood estimation of a binary choice model.

There is ongoing debate on DPO vs. PPO for LLM alignment (Xu et al., 2024).

### 7.11.14 Empirical Performance of DPO

DPO achieves competitive or superior performance compared to RLHF on standard benchmarks:

- On **summarization helpfulness**, DPO achieves the highest win rates against ground truth, outperforming PPO, Best-of-128 sampling, Preference-Filtered Training (PFT), and SFT.
- On **dialogue helpfulness**, DPO similarly leads in win rates, demonstrating that the closed-form approach does not sacrifice quality.
- The pipeline is dramatically simpler: no reward model training, no value function fitting, no online sampling, and standard supervised learning optimizers suffice.

### 7.11.15 Application: DPO for Balancing Engagement and Polarization

Chang et al. (2025) demonstrate a compelling business application of DPO: using LLMs to generate news content that **balances engagement and polarization**.

**Problem.** Media firms face a multi-objective challenge, making content more engaging while maintaining a preferred level of polarization consistent with the firm's editorial policy. Using news articles from *The New York Times*, the authors show that more engaging human-written content tends to be more polarizing. Further, naively applying standard DPO approaches to generate more engaging content using LLMs without explicitly controlling for polarization can also increase polarization.

**Solution.** The authors propose **Multi-Objective Direct Preference Optimization (MODPO)** (Li et al., 2024), a novel approach that integrates DPO with multi-objective optimization techniques. They build an open-source LLM that simultaneously makes content more engaging while maintaining a preferred editorial stance. Their model achieves this by modifying content characteristics strongly associated with polarization but that have a relatively smaller impact on engagement.

**Key takeaway.** This work illustrates how preference optimization techniques from RL can be applied to real-world content generation problems where multiple, potentially conflicting objectives must be balanced.

### 7.11.16 Chain-of-Thought (CoT) Reasoning

Beyond RLHF and DPO, another powerful technique for improving LLM performance is **Chain-of-Thought (CoT) prompting** (Wei et al., 2022; Kojima et al., 2022):

- CoT is a technique for LLMs to **"think aloud to itself"** before producing an answer.
- CoT **greatly improves the performance** compared with immediately producing an answer, especially on arithmetic, commonsense, and symbolic reasoning tasks.
- An easy way to induce CoT: append **"Let's think step by step"** to the prompt (zero-shot CoT).
- CoT can also be induced by **in-context prompting**, providing a few examples of step-by-step reasoning (few-shot CoT).
- Modern LLMs are increasingly **instruction-finetuned** to exhibit CoT behavior natively, enabling them to produce reasoning chains without explicit prompting.

**Connection to RL.** Recall from Section 7.11.2 that the post-training pipeline includes an **RL-CoT track** for reasoning models. In this track, RL is used to train models to produce long chains of thought, enabling **test-time scaling**, the model can allocate more computation at inference time by reasoning through more steps, improving accuracy on harder problems without any explicit reward model.

### 7.11.17 Reinforcement Learning with Verifiable Rewards (RLVR)

A key challenge in RLHF is that the reward model is **imperfect**, it is a learned proxy for human preferences, subject to reward hacking. However, some domains offer a fundamentally different opportunity: **coding and mathematics** are challenging reasoning tasks with **verifiable rewards** (Chen et al., 2021; Shao et al., 2024).

- Automatic code generation was a longstanding challenge, until LLMs emerged as a solution. Code data with comments written in natural language is plentiful on the Internet, so modern LLMs are explicitly trained on code together with natural language data.

- More interestingly, coding improves non-coding reasoning capabilities as well, such as math.
- Training AI on math also improves general reasoning capabilities.

The key insight is that coding and math problems have **objectively verifiable solutions**: a code submission either passes the test suite or it does not; a mathematical answer is either correct or incorrect. This eliminates the need for a learned reward model entirely.

**Reinforcement Learning with Verifiable Rewards (RLVR)** exploits this structure:

High-quality reasoning data $\rightarrow$ Base LLM $\rightarrow$ Reinforcement Learning $\rightarrow$ Reasoning LLM

Since the reward is **exact** (verifiable), there is no risk of reward hacking or overfitting to an imperfect reward model. Only **outcome rewards** are provided (whether the final answer is correct). Intermediate rewards on partial credits, i.e., **process rewards**, are *not* used.

### 7.11.18 Group Relative Policy Optimization (GRPO)

**DeepSeekMath** (Shao et al., 2024) proposes **Group Relative Policy Optimization (GRPO)**, a key algorithmic innovation that simplifies PPO for the RLVR setting.

**Motivation.** Standard PPO for RLHF requires maintaining four models simultaneously: the policy model, a reference model (for KL penalty), a reward model, and a **value model** (critic). The value model is expensive to train and maintain. GRPO eliminates the value model entirely by using **group-level relative advantages** instead.

**Key innovations of GRPO.**

1. **Group Relative Advantage.** Instead of training a value function to estimate advantages, GRPO samples a **group** of $G$ responses to the same problem and computes the advantage by **normalizing rewards within the group**:

$$\hat{A}^{(i)} = \frac{r^{(i)} - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r}) + \varepsilon_A} \tag{7.11.17}$$

   This eliminates the need for a separate value model, the group statistics serve as a natural baseline. Responses that are better than the group average receive positive advantage; those worse receive negative advantage.

2. **Length Normalization.** The objective divides by $|T^{(i)}|$ (the response length), preventing the optimization from favoring shorter or longer responses.

**Algorithm 22** Group Relative Policy Optimization (GRPO)

**while** not converged **do**
  Sample a problem $x$
  Sample $G$ responses $y^{(1)}, \ldots, y^{(G)} \sim \pi_{\theta_{\text{curr}}}(\cdot \mid x)$
  Evaluate the rewards $\mathbf{r} = (r^{(1)}, \ldots, r^{(G)})$ for $y^{(1)}, \ldots, y^{(G)}$
  Solve:

$$\max_{\theta_{\text{next}}} \sum_{i=1}^{G} \frac{1}{|T^{(i)}|} \sum_{t=0}^{T^{(i)}} \left(\text{term}_1 + \text{term}_2\right)$$

  where

$$\text{term}_1 = C_\varepsilon \left( \frac{\pi_{\theta_{\text{next}}}(y_{t+1}^{(i)} \mid x, y_{1:t}^{(i)})}{\pi_{\theta_{\text{curr}}}(y_{t+1}^{(i)} \mid x, y_{1:t}^{(i)})}, \; \frac{r^{(i)} - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r}) + \varepsilon_A} \right)$$

$$\text{term}_2 = -\beta \left( \frac{\pi^{\text{SFT}}(y_{s+1}^{(i)} \mid x^{(i)}, y_{1:s}^{(i)})}{\pi_{\theta_{\text{next}}}(y_{s+1}^{(i)} \mid x^{(i)}, y_{1:s}^{(i)})} - \log \frac{\pi^{\text{SFT}}(y_{s+1}^{(i)} \mid x^{(i)}, y_{1:s}^{(i)})}{\pi^{\text{RL}}_{\theta_{\text{next}}}(y_{s+1}^{(i)} \mid x^{(i)}, y_{1:s}^{(i)})} \right)$$

  Set $\theta_{\text{curr}} = \theta_{\text{next}}$
**end while**

3. **Unbiased KL Estimate.** The KL penalty term uses an unbiased estimator of the KL divergence between the RL policy and the SFT reference policy, rather than the standard log-ratio approximation.

*Remark* (Group normalization is a within-group control variate). GRPO's advantage (7.11.17) simply *standardizes* (z-scores) the rewards of the $G$ responses drawn for the same prompt $x$. Subtracting the group mean is a classic **control-variate / baseline** for variance reduction—a policy-gradient baseline estimated on the fly from a small "mini-experiment" of $G$ comparable responses—and is conceptually the *within-group demeaning* an econometrician performs in a fixed-effects (panel) estimator to remove a prompt-specific level. Dividing by the group standard deviation further rescales updates so that easy prompts (where all $G$ responses score similarly) and hard prompts contribute on a comparable scale; the small $\varepsilon_A$ in the denominator only guards against division by zero. Because the advantages within each group have mean zero by construction, no separately trained value network is needed to supply a baseline—which is precisely how GRPO eliminates the critic.

**Comparison with PPO.** While PPO maintains four models (policy, reference, reward, value), GRPO only needs three (policy, reference, reward), or just two (policy, reference) when rewards are verifiable and no learned reward model is needed. This makes GRPO substantially more memory-efficient and simpler to implement.

### 7.11.19 DeepSeek-R1: Scaling Up RLVR

**DeepSeek-R1** (DeepSeek-AI, 2025) scales up the RLVR approach to coding, math, science, and other reasoning tasks, demonstrating that RL with verifiable rewards can produce reasoning capabilities competitive with frontier models.

**Training pipeline.** DeepSeek-R1 is built through a multi-stage process.

**Step 1: Knowledge Distillation (Open-R1-Distill-8B).**

- Directly imitate the outputs of a strong reasoning model (DeepSeek-R1) through supervised fine-tuning on distilled reasoning data.
- This produces a capable but not yet optimal model.

**Step 2: DeepSeek-R1-Zero.**

- Directly apply RLVR to a base model (DeepSeek-V3) using GRPO with verifiable rewards, **no SFT, no reward model, just pure RL**.
- The model learns to produce long CoT reasoning chains.
- However, the resulting CoT often suffers from **poor readability** (mixing languages, disorganized reasoning steps).

**Step 3: DeepSeek-R1.**

- **Cold-start with SFT:** first fine-tune on a small amount of curated reasoning data to establish good formatting and readability.
- Then combine **RLVR and RLHF with GRPO:** apply RL with both verifiable rewards (for reasoning correctness) and human feedback (for readability and helpfulness).
- This yields the final DeepSeek-R1 model with both strong reasoning and good output quality.

**Key results.**

- The model learns to utilize **CoT very extensively**, with average response lengths growing significantly during training (from $\sim$2,500 tokens to $\sim$12,500+ tokens over 10,000 training steps).
- On the AIME mathematical reasoning benchmark, DeepSeek-R1-Zero achieves accuracy that surpasses human participants, improving steadily throughout training.

### 7.11.20 Emergence of Reasoning Through RLVR

Perhaps the most striking finding from the DeepSeek-R1 work is that **RLVR induces the emergence of sophisticated reasoning behaviors** that are **not explicitly programmed** (DeepSeek-AI, 2025). As test-time computation increases, the model develops key **cognitive behaviors for reasoning**, such as:

- **Verification:** checking intermediate results for correctness.
- **Back-tracking:** recognizing errors and revising the approach.
- **Subgoal setting:** breaking complex problems into manageable sub-problems.
- **Backward chaining:** working from the desired conclusion back to the premises.
- **Self-evolution:** the model refines its own reasoning strategies over the course of training.

**The "Aha Moment" of DeepSeek-R1.** During training, the model exhibits a remarkable behavior: it pauses mid-solution, recognizes that something is wrong ("Wait, wait. Wait. That's an aha moment I can flag here."), re-evaluates its approach step by step, and arrives at the correct answer. This emergent self-correction behavior was never explicitly trained, it arose purely from RL optimization against verifiable rewards.

**Why does this work?** The argument is not that "RL didn't work before", PPO on base models with verifiable rewards had shown strong results previously. Rather, the breakthrough is that with **sufficient base model knowledge** and **sufficient RL compute and context window length**, the model develops **long CoT with internal reasoning emergence**: verification, error-correction, and branching-like behavior that earlier, smaller models could not sustain.

*Remark* (The broader significance). RLVR represents a paradigm where RL can train reasoning capabilities without any human-generated reasoning traces or learned reward models. Combined with the group relative advantage of GRPO that eliminates the value network, this approach dramatically simplifies the pipeline from base model to reasoning model: just RL with verifiable rewards.

## 7.12 Conclusion

Deep reinforcement learning represents a powerful synthesis of deep learning's representational capacity with reinforcement learning's sequential decision-making frame-

work. The progression from tabular RL to function approximation to deep RL follows a natural path of increasing scalability and expressiveness:

1. **Value function approximation** replaces the lookup table with a parameterized function, enabling generalization across states and scaling to large problems.
2. **Linear approximation** provides a tractable starting point with convergence guarantees, but requires manual feature engineering.
3. **Deep Q-Networks** leverage neural networks for automatic feature learning, with experience replay and fixed targets to stabilize training.
4. **Policy gradient methods** optimize the policy directly, enabling handling of continuous and high-dimensional action spaces, stochastic policies, and settings where value-based methods struggle.
5. **Actor-critic methods** combine the best of both worlds: a critic for low-variance value estimation and an actor for direct policy optimization, culminating in scalable algorithms like A3C.
6. **Modern DRL algorithms**, TRPO and PPO, address the instability of vanilla policy gradients through constrained optimization in policy space, with PPO emerging as the practical workhorse due to its simplicity and effectiveness.
7. **RL for LLMs** applies these techniques, particularly PPO, to align large language models with human preferences through RLHF, representing one of the most impactful real-world applications of deep RL to date.
8. **Direct Preference Optimization (DPO)** simplifies RLHF by exploiting a closed-form solution for KL-regularized RL, converting the alignment problem into supervised learning and eliminating the need for reward models, value networks, and online sampling.
9. **Chain-of-Thought (CoT) reasoning** enables LLMs to "think step by step," dramatically improving performance on reasoning tasks and forming the basis of test-time scaling in modern reasoning models.
10. **RLVR and GRPO** demonstrate that for domains with verifiable rewards (coding, math, science), RL can train reasoning capabilities without learned reward models or value networks, and remarkably, sophisticated reasoning behaviors such as self-correction and back-tracking **emerge** from pure RL optimization.

The key takeaways for business researchers are:

- **The deadly triad** (bootstrapping + function approximation + off-policy learning) is a fundamental source of instability. DQN addresses it through engineering innovations (experience replay and fixed targets) rather than theoretical fixes.
- **Policy gradient methods** provide a complementary approach to value-based methods, with distinct advantages for continuous action spaces, stochastic policies, and LLM fine-tuning (e.g., RLHF).
- **Variance reduction** is central to making policy gradient methods practical. Temporal causality, baselines, and actor-critic architectures progressively reduce the variance of gradient estimates.
- **Trust regions and clipping** (TRPO and PPO) are essential for stable policy optimization. The insight that RL training can collapse from a single bad update, unlike supervised learning, motivates constraining policy changes, whether through KL divergence constraints or clipped objectives.
- **RLHF and reward modeling** have emerged as the dominant paradigm for post-training LLMs. The connection between PPO's trust-region approach and the KL penalty in RLHF highlights how foundational RL concepts directly enable modern AI alignment. However, reward hacking (Goodhart's Law) remains a fundamental challenge.
- **DPO offers a simpler alternative** to the full RLHF pipeline, with competitive empirical performance. Its application to multi-objective content generation (e.g., balancing engagement and polarization) demonstrates the versatility of preference optimization for business applications.
- Deep RL opens the door to applications with high-dimensional state spaces, such as dynamic pricing, personalized recommendations, adaptive marketing, inventory management, customer journey optimization, and LLM-powered content generation, where tabular methods are infeasible.
- **RLVR and emergent reasoning** represent a frontier where RL with verifiable rewards produces sophisticated cognitive behaviors, verification, back-tracking, self-correction, without explicit programming. GRPO's elimination of the value network further simplifies the training pipeline, pointing toward a future where reasoning capabilities arise naturally from scale and RL optimization.
- **Extensions** like Double DQN, Prioritized Replay, Dueling DQN, A3C, TRPO, PPO, DPO, GRPO, and CoT reasoning offer further improvements and continue to be active areas of research.

# Chapter 8: Agentic AI

This chapter introduces *Agentic AI*, the emerging paradigm in which large language models (LLMs) move beyond generating text to autonomously *acting* in the world. We begin by situating agentic AI within the broader arc of the course, define what constitutes an AI agent, and trace the conceptual evolution from conversational chatbots to autonomous systems capable of planning, tool use, and multi-step reasoning. We then examine the infrastructure that enables agents, *skills* as standard operating procedures and the Model Context Protocol (MCP), before conducting a detailed case study of *OpenClaw*, an open-source agentic system whose viral adoption in early 2026 epitomizes both the promise and peril of consumer-facing AI agents.

## 8.1 Where Are We?

Before diving into agentic AI, it is helpful to situate this lecture within the broader arc of the course:

- **Reasoning LLMs** serve as the backbone for AI agents. Models capable of chain-of-thought (CoT) reasoning, planning, and self-correction provide the "brain" that powers autonomous behavior.
- **LLM-driven research automation**, replication automation, Automated Prompt Engineering (APE), Autoresearch, and FARS, has already demonstrated that LLM agents can drive paradigm shifts in academic research workflows.
- **Agent-based modeling** (e.g., structural models in economics and operations) has been an important and long-standing methodology in business, economics, and social science. AI agents extend this tradition with learned, adaptive policies rather than hand-crafted behavioral rules.
- **AI agents are impacting real business** today, from customer service chatbots and market research copilots to coding assistants and supply chain robotics.

A useful way to think about progress is the **automation ladder**:

1. **Robustness:** Automating tedious digital labor (e.g., data entry, report generation).
2. **Collaboration:** Facilitating human interactions (e.g., scheduling, email drafting, meeting summarization).
3. **Exploration:** Enabling creativity and scientific discovery (e.g., hypothesis generation, experimental design).

**The fundamental transition:** From ChatGPT to OpenClaw, from *AI that talks* to *AI that acts*.

## 8.2 LLM Agents

### 8.2.1 What Is an Agent?

An **agent** is an intelligent system that interacts with some environment via a cycle of actions and observations.[46] This definition is deliberately broad: it encompasses classical AI agents (rule-based systems, search algorithms), neural agents (learned policies via reinforcement learning), and the new generation of **language agents** that use natural language as the vehicle for both reasoning and communication.

What distinguishes modern LLM agents from earlier generations is the *generality* of the language interface. Classical agents typically required domain-specific state representations and hand-crafted action primitives. LLM agents, by contrast, can:

- **Reason** in natural language about complex, open-ended tasks.
- **Communicate** with humans and other agents using the same natural language.
- **Generalize** across domains without task-specific retraining, leveraging the broad world knowledge embedded in pretrained models.

### 8.2.2 From LLM to Agents

A standalone LLM can be thought of as a *brain in a vat*, immensely knowledgeable but unable to perceive or act upon the world. The transition from an LLM to an agent involves equipping this brain with **interactions with the environment**: the ability to observe states, take actions, receive feedback, and iteratively refine its approach.

This transition requires several capabilities beyond text generation:

- **Perception:** Processing inputs from the environment (user messages, tool outputs, error logs, sensor readings).
- **Reasoning and Planning:** Decomposing high-level goals into ordered sub-tasks, selecting among strategies, and revising plans in light of new information.
- **Action:** Executing steps via tool calls, API invocations, code execution, or message generation.

[46] See the Language Agent Tutorial, https://language-agent-tutorial.github.io/slides/I-Introduction.pdf, and the LLM Agents MOOC, https://llmagents-learning.org/slides/intro.pdf, for comprehensive introductions.

- **Memory:** Maintaining both short-term context (within the conversation) and long-term knowledge (persisted across sessions).

### 8.2.3 Conceptual Framework of LLM Agents

The conceptual framework for LLM agents revolves around the **agentic loop**, the iterative cycle through which agents solve real-world tasks. At each turn the agent forms an *observation* of its environment, performs internal *reasoning*, takes an *action*, and receives *feedback* that becomes the next observation, closing the loop. Figure 8.2.1 depicts this cycle.

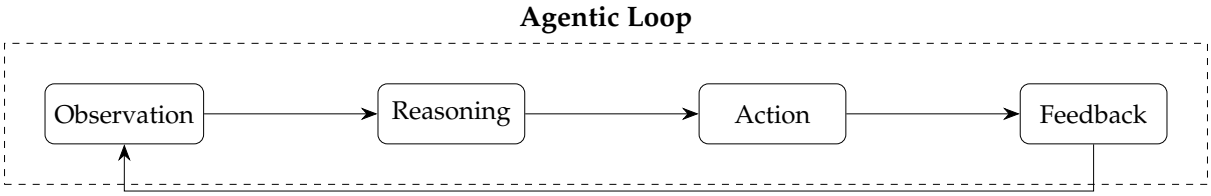


**Figure 8.2.1.** The agentic loop: the agent iteratively observes the environment, reasons about the next step, acts, and incorporates feedback as the next observation.

**Key properties of the agentic loop:**

1. **Trial-and-error.** Solving real-world tasks typically involves iterative refinement. The agent attempts an action, observes the result, and adjusts its approach, much like how a human programmer debugs code by running it, reading error messages, and modifying the source.

2. **External tool use.** Leveraging external tools (calculators, search engines, APIs, databases) and retrieving from external knowledge bases expand the LLM's capabilities beyond its parametric knowledge. Tool use transforms the model from a knowledge retrieval system into an *action-taking* system.

3. **Task decomposition and subtask allocation.** Agentic workflows facilitate complex tasks through decomposition into manageable sub-problems, allocation and parallelization of subtasks to specialized modules, and division of labor for project collaboration.

4. **Multi-agent generation.** Having multiple agents collaborate, each with distinct roles, expertise, or perspectives, can inspire better responses through debate, verification, and synthesis, analogous to ensemble methods in machine learning.[47]

[47] The analogy is precise. Several independent or diverse agents voting on an answer is a form of *model averaging*: just as bagging reduces the variance of a predictor by averaging many decorrelated estimators (Breiman, 2001), polling several agents (or the same agent run with different random seeds) and aggregating their replies tends to cancel idiosyncratic mistakes. Sequential schemes in which one agent critiques or revises another—debate, verification, self-correction—are closer in spirit to *boosting* or *stacking*, where later learners are arranged to fix the residual errors of earlier ones.

*Remark* (The agentic loop as a sequential decision process). Readers of Chapters 5–7 will recognize the agentic loop of Figure 8.2.1 as a relabelling of the agent–environment interaction diagram that defines a Markov decision process (MDP). The correspondence is exact: the *observation* plays the role of the state $s_t$; the *action* $a_t$ is the tool call or message the agent emits; the *feedback* is the next observation $s_{t+1}$ (together with an implicit reward that signals task progress); and *reasoning* is the internal computation that the policy performs before acting. What is genuinely new is the *form* of the policy and how it is obtained. In Chapters 5–7 the policy $\pi_\theta(a \mid s)$ was a numeric map—a lookup table or a neural network—that was *trained* by trial-and-error to maximize expected discounted reward. Here the policy is a frozen LLM that maps a *textual* context to a *textual* action, and at run time it is steered not by online reward maximization but by its instructions (skills), its available tools, and the contents of its context window. An LLM agent is therefore better described as *executing a fixed policy under careful prompt and context design* than as *solving* an MDP by reinforcement learning. Accordingly, the trial-and-error of property 1 above happens *within* a single task episode (read the error message, try again), not across thousands of training episodes. This distinction—behavior shaped by context rather than by weight updates—recurs throughout the chapter.

### 8.2.4 The Agent as a Smart PhD with SOPs

A useful metaphor, articulated by Hung-yi Lee in his Machine Learning course,[48] frames LLM agents as **smart, knowledgeable PhDs who need clear standard operating procedures (SOPs)**. Just as a newly hired researcher, no matter how brilliant, needs onboarding documents, lab protocols, and institutional workflows to be productive, an LLM agent needs structured instructions (skills), access to tools (MCPs), and well-organized context to perform effectively.

This metaphor highlights an important asymmetry: the bottleneck is often not the agent's *intelligence* but the *quality of its operating instructions and environment*. Improving agent performance is frequently more about better context engineering than better models.

*Remark* (An organizational-economics reading). This framing will resonate with anyone who studies firms. A brilliant new hire is unproductive until the organization supplies routines, checklists, and codified know-how—what the management literature calls *standard operating procedures* and *organizational routines*. The claim that an agent's bottleneck is its operating instructions rather than its raw intelligence is, in this language, a claim that the binding constraint on performance is the *firm's* accumulated procedural capital, not the *worker's* talent. Skills, MCP tools, and context engineering

[48] https://speech.ee.ntu.edu.tw/~hylee/ml/ml2026-course-data/intro.pdf

(the subjects of the next sections) are simply the mechanisms by which that procedural capital is written down and handed to the agent.

## 8.3 Skills and the Model Context Protocol (MCP)

### 8.3.1 Agent Skills

**Skills** are, fundamentally, SOPs for LLM agents to complete specific tasks.[49] Recall the metaphor from Section 8.2.4: LLM agents are smart, knowledgeable PhDs who need clear SOPs. Skills formalize these SOPs into modular, reusable components that encode domain-specific expertise.

A skill encapsulates:

- **Instructions:** A structured prompt defining what the agent should do, how it should do it, and what constraints apply.
- **Metadata:** Information about when the skill should be triggered, what tools it requires, and what outputs it produces.
- **Best practices:** Curated patterns from expert users or the developer community that encode domain-specific knowledge.

Skills can be:

- **Authored by developers** who understand a particular workflow.
- **Downloaded from repositories** such as the Anthropic Skills repository or community hubs like AgentSkills.io.
- **Generated by AI agents themselves**, an agent can write new skills based on observed best practices, creating a self-evolution loop (more on this in Section 8.4).

The skill paradigm represents a shift from monolithic prompt engineering to **modular, reusable, and composable** agent capabilities.

### 8.3.2 Skill Archetypes

Anthropic's guide identifies **three primary skill archetypes**, each suited to a different class of tasks, as summarized in Table 8.3.1.

[49] See Lee (2026), https://speech.ee.ntu.edu.tw/~hylee/ml/ml2026-course-data/intro.pdf; Anthropic's *Complete Guide to Building Skills for Claude*, https://resources.anthropic.com/hubfs/The-Complete-Guide-to-Building-Skill-for-Claude.pdf; and the DeepLearning.AI short course, https://www.deeplearning.ai/short-courses/agent-skills-with-anthropic/.

**Table 8.3.1.** The three primary skill archetypes.

| Archetype | Purpose | External Tools? | Example |
|---|---|---|---|
| Document & Asset Creation | Creates consistent, production-grade outputs (presentations, code, designs, documents) | No, leverages the LLM's built-in capabilities | A "frontend-design" skill that generates distinctive UI components following a design system |
| Workflow Automation | Manages multi-step processes with templates, validation gates, and iterative refinement loops | Optional | A "skill-creator" skill that walks the agent through use-case definition, frontmatter generation, instruction writing, and validation |
| MCP Enhancement | Layers domain expertise on top of MCP tool access, turning raw API connectivity into reliable guided workflows | Yes, depends on MCP servers | A code-review skill that analyzes GitHub PRs using Sentry error data via its MCP server |

The key distinction: **Document & Asset Creation** skills are self-contained (no external tools needed), **Workflow Automation** skills are process-oriented with validation loops, and **MCP Enhancement** skills coordinate external tool calls with expert guidance. In practice, many production skills blend elements from multiple archetypes.

### 8.3.3 Anatomy of a Skill

While the file architecture may differ across agent platforms, the underlying logic remains the same. A typical skill directory follows this structure:

```
skills/
+-- skill-name/
    +-- SKILL.md # Main skill definition (required)
    +-- references/ # Detailed documentation and patterns
    | +-- patterns.md
    +-- examples/ # Complete, runnable examples
    | +-- sample.md
    +-- scripts/ # Executable utilities and validators
        +-- helper.sh
```

- `SKILL.md` is the required entry point, it contains YAML frontmatter (metadata)[50] and the markdown body (instructions).

[50] YAML is a lightweight, human-readable format for writing structured data as indented `key: value` pairs (a friendlier alternative to JSON, the JavaScript Object Notation widely used for data in-

- `references/` holds detailed documentation, advanced techniques, and edge-case guidance that the agent loads only when needed.
- `examples/` contains complete, runnable examples and templates.
- `scripts/` houses executable utilities whose *output* (not source code) enters the context window.

This modular structure is critical for context efficiency: the agent loads only what it needs, when it needs it, a principle formalized as *progressive disclosure* (Section 8.3.4).

### 8.3.4 Progressive Disclosure

A core architectural insight behind skills is **progressive disclosure**: the principle that LLM agents should access deep knowledge without easily exhausting context windows.[51] Progressive disclosure minimizes token usage while maintaining access to specialized expertise through a three-level loading system, summarized in Table 8.3.2.

**Table 8.3.2.** The three-level progressive disclosure architecture for skills.

| Level | What Is Loaded | When | Approx. Token Cost |
|---|---|---|---|
| 1. Metadata | `name` and `description` from YAML frontmatter | Always (at startup, injected into system prompt) | ∼50-100 tokens per skill |
| 2. Instructions | Full `SKILL.md` body | When the skill is triggered | Up to ∼5,000 tokens |
| 3. Resources & Code | Files in `references/`, `examples/`, `scripts/` | On-demand, only when referenced by instructions | Effectively unlimited |

**How it works:** At startup, the agent loads *only* Level 1 metadata from every installed skill into its system prompt. This costs negligible tokens but gives the agent enough context to recognize when a skill is relevant. When triggered, the full `SKILL.md` body (Level 2) is read into context. Detailed reference materials and executable scripts

terchange). *Frontmatter* is a short metadata block fenced by `---` delimiters at the very top of a file, a convention borrowed from static-site generators; here it carries the skill's `name` and `description` fields.

[51] Recall from the earlier chapters that an LLM does not read raw characters but *tokens*—sub-word chunks (as a rule of thumb, one token is about 3–4 English characters, or roughly 0.75 of a word). The *context window* is the maximum number of tokens the model can attend to in a single forward pass—its working memory for one step. Everything the agent "knows" at that moment (system prompt, instructions, conversation history, retrieved documents, tool outputs) must fit inside it. Both the model's hard context limit and the per-call price charged by commercial inference APIs are denominated in tokens, which is why token budgeting (this section) and the "token economy" of Section 8.4.3 are recurring concerns of the chapter.

(Level 3) are loaded only when the instructions explicitly reference them, and for scripts, only the *output* enters the context window, not the source code itself.

The analogy is a well-organized manual: the table of contents (Level 1) is always visible, specific chapters (Level 2) are opened as needed, and the detailed appendix (Level 3) is consulted only for specialized questions. This layered approach enables dozens of skills to coexist with minimal context overhead, the agent pays the full token cost only for the skill it is actively using.

### 8.3.5 Frontmatter: When Skills Are Triggered

The **frontmatter** is the YAML metadata block at the top of `SKILL.md`. It is loaded into the system prompt and serves as the primary mechanism for determining *when* a skill is activated:

```
---
name: skill-name
description: This skill should be used when the user asks to "specific phrase
    1",
  "specific phrase 2", or mentions specific-topic. Provides [capability].
---
```

**Triggering mechanism:** When a user makes a request, the agent evaluates it against all available skill descriptions using semantic matching. If the request aligns with a skill's description, the agent issues a skill invocation, which loads the full `SKILL.md` body into context.

**Best practices for the `description` field:**

- **Write in third person**, the description is injected into the system prompt alongside other metadata, so first-person ("I can help you...") or second-person ("You can use this to...") creates point-of-view inconsistency.
- **Include specific trigger phrases** that users would naturally say, enclosed in quotes (e.g., `"create a PR"`, `"review this code"`).
- **Be concrete about activation conditions**, state both *what* the skill does and *when* it should be used.
- **Be assertive**, include language like "Make sure to use this skill whenever..." to encourage reliable triggering.

The description is the single most critical field in the entire skill definition. A vague description (e.g., "Helps with PDFs") will fail to trigger reliably; a specific one (e.g., "This skill should be used when the user asks to 'rotate a PDF', 'merge PDFs', or mentions PDF manipulation") will trigger consistently.

### 8.3.6 Nonambiguous Instructions

A fundamental challenge in skill design is that **code is deterministic, but natural language is not**. The same instruction may be interpreted differently across invocations due to the probabilistic nature of LLMs. Effective skills write instructions that leave no room for guesswork:

**1. Replace vague directives with explicit sequences.**

- Vague: "Validate the data."
- Explicit: "Before calling `create_project`, verify: (a) project name is non-empty, (b) at least one team member is assigned, (c) start date is not in the past."

**2. Use code for deterministic operations.** For critical validations, bundle a script rather than relying on language instructions. The agent handles qualitative reasoning (understanding user intent) while scripts provide quantitative precision (deterministic execution). Only the script's *output* enters the context window.

**3. Calibrate the degree of freedom.** Not all instructions require the same precision. The appropriate level of specificity depends on the task, as shown in Table 8.3.3.

**Table 8.3.3.** Calibrating the degree of freedom in skill instructions.

| Degree of Freedom | When to Use | Example |
|---|---|---|
| High (text instructions) | Multiple valid approaches; decisions depend on context | Code review, creative writing |
| Medium (pseudocode / parameterized scripts) | A preferred pattern exists but some variation is acceptable | API integration workflows |
| Low (specific scripts, exact sequences) | Operations are fragile or consistency is critical | Database migrations, deployment pipelines |

**4. Structure instructions clearly.** Effective skill instructions follow a consistent format: a brief purpose statement, prerequisites, step-by-step procedures in imperative mood ("Analyze…", "Execute…"), output format definitions, and error handling guidance with common failure scenarios.

You can download useful skills from the Anthropic Skills repository or community hubs like AgentSkills.io, and ask your AI agents to write skills for you based on these best practices.

### 8.3.7 The Model Context Protocol (MCP)

The **Model Context Protocol (MCP)**[52] is an open standard introduced by Anthropic in November 2024 for connecting AI agents to external applications, data sources, and tools. It provides a universal, standardized interface, analogous to how the Language Server Protocol (LSP)[53] standardized how programming language support works across development tools.

**The problem MCP solves:** Before MCP, every new data source or tool required a bespoke integration. With $M$ AI applications and $N$ tools, this created an $M \times N$ integration problem. MCP replaces this combinatorial explosion with a single standard: each application implements one MCP client, and each tool implements one MCP server, reducing the problem to $M + N$.

*Remark* (Standardization economics: from $M \times N$ to $M + N$). The combinatorial argument for MCP is one that business and economics researchers will find familiar. Connecting $M$ applications to $N$ tools through bespoke, point-to-point integrations requires building and maintaining on the order of $M \times N$ bilateral "adapters"—the same quadratic blow-up that makes point-to-point logistics networks, bilateral clearing among banks, or pairwise human translation prohibitively costly at scale. A common *standard* (a shared interface, a clearinghouse, a hub-and-spoke network, a *lingua franca*) replaces those $M \times N$ links with $M + N$ connections to a single hub, turning a quadratic cost into a linear one. MCP also exhibits classic *indirect network externalities*, the hallmark of a two-sided platform: each new server (tool) that speaks MCP raises the value of every MCP-speaking client, and each new client raises the incentive to publish a server. This positive feedback is precisely why a protocol, once it crosses a critical mass of adopters, tends to tip the market and become the de facto standard—as MCP did within months of its release.

**Architecture:** MCP follows a **client-host-server** model, summarized in Table 8.3.4.

A single host can connect to many servers simultaneously, e.g., one for file system access, one for a database, one for GitHub, one for Slack, each through its own dedicated client instance.

**Communication:** MCP uses JSON-RPC 2.0[54] messages over two transport options:

[52] `https://www.anthropic.com/news/model-context-protocol`; protocol specification at `https://modelcontextprotocol.io/`.

[53] Introduced by Microsoft in 2016 for the Visual Studio Code editor, the LSP lets any code editor obtain language-specific features (autocompletion, error checking, "go to definition") from any programming language's "language server" through one shared interface. Before it, supporting $M$ editors across $N$ languages required up to $M \times N$ bespoke plug-ins; the LSP collapses this to $M + N$—exactly the integration economics MCP brings to AI tools (see the remark on standardization below).

[54] JSON-RPC is a minimal, transport-agnostic convention for *remote procedure calls* (RPC): one program sends the other a small JSON message naming a method and its arguments (for example, the method `tools/call` together with a parameter block) and receives back a JSON message containing either the result or an error. Informally, it is a standardized way to "call a function that lives inside another

**Table 8.3.4.** The client-host-server architecture of the Model Context Protocol.

| Component | Role | Example |
|---|---|---|
| Host | The top-level AI application that the user interacts with. Coordinates one or more MCP clients. | Claude Desktop, Claude Code, Cursor, an IDE |
| Client | A connector within the host that maintains a dedicated 1:1 connection to a single MCP server. | One client per connected service |
| Server | A lightweight program exposing specific capabilities (tools, data, prompts) through the MCP protocol. | A GitHub server, a database server, a calendar server |

- **stdio:** Local process communication with no network overhead, ideal for tools running on the same machine.
- **Streamable HTTP:** Remote servers accessible over the network, with optional Server-Sent Events for streaming responses.

**Core primitives exposed by MCP servers:**

1. **Tools:** Executable functions the AI can invoke (e.g., file operations, API calls, database queries). Discovered via `tools/list`, executed via `tools/call`.
2. **Resources:** Data sources providing contextual information (e.g., file contents, database records, calendar entries).
3. **Prompts:** Reusable templates for structuring LLM interactions (e.g., domain-specific system prompts).

**Industry adoption:** MCP has achieved rapid adoption. OpenAI adopted MCP in March 2025, Google DeepMind confirmed support in April 2025, and Microsoft followed shortly after. Development tool companies (Zed, Replit, Codeium, Sourcegraph) and enterprises have integrated MCP into production agent systems.

> **Why MCP matters for researchers:** MCP lowers the barrier to building agents that interact with real-world systems. A researcher can now connect an LLM agent to their institutional database, their email, their calendar, and their codebase through a single protocol, without writing custom integrations for each.

program and get a structured answer back." The two transports named below are simply the pipes these messages travel through: *stdio* (standard input/output) is the pair of text streams that any command-line program already reads from and writes to, used when client and server run on the same machine; *Server-Sent Events* (SSE) is a one-way HTTP channel over which a remote server can push a stream of incremental messages to the client, useful when a long-running tool returns its answer in pieces.

## 8.4 OpenClaw: A Case Study in Agentic AI

### 8.4.1 What Is OpenClaw?

OpenClaw (formerly Clawdbot, then MoltBot) is a free and open-source agentic system that executes tasks via LLMs using messaging platforms (WhatsApp, Lark, Telegram, Discord, Signal) as its primary user interface.[55] Created by Austrian developer Peter Steinberger, it was first published in November 2025 and went viral in late January 2026.

OpenClaw represents a paradigm shift: instead of interacting with AI through a specialized interface (a chat window, an IDE), users interact through the messaging apps they already use daily. The agent runs locally on the user's machine, connecting to LLM providers (Claude, DeepSeek, GPT) and executing tasks autonomously.

**Key statistics at peak virality (February 2026):**

- Over 145,000 GitHub stars (a record for an open-source project).
- 2 million visitors per week at peak traffic.
- Over 30,000 publicly exposed instances.

### 8.4.2 Behind OpenClaw's Viral Growth

OpenClaw's viral success mirrors the dynamics of DeepSeek: **democratizing capabilities previously limited to developers**. By connecting agentic AI to everyday messaging apps, OpenClaw gave non-technical users their first real taste of autonomous AI agents. The key insight was that *approachability matters more than raw capability* for mass adoption.

The platform's growth was amplified by mainstream visibility, NVIDIA's GTC 2026 keynote highlighted OpenClaw as an exemplar of the emerging **token economy** in agentic AI. The speculative frenzy around the platform reached absurd heights: during a brief window when the original "Clawdbot" social media handle was released (due to a trademark dispute with Anthropic), cryptocurrency scammers launched the fraudulent $CLAWD token, which reached a $16 million market cap before collapsing. OpenClaw itself has no token or blockchain component, but the incident illustrates the volatile intersection of agentic AI hype and speculative finance.

[55] See the Yage.ai deep dive, https://yage.ai/openclaw-en.html; the OpenClaw GitHub, https://github.com/openclaw/openclaw; and the NVIDIA GTC keynote, https://www.nvidia.com/gtc/keynote/.

### 8.4.3 The Token Economy and Alibaba Token Hub

OpenClaw's emergence coincided with, and was amplified by, a broader paradigm shift that Jensen Huang articulated as the **token economy** during NVIDIA's GTC 2026 keynote. The core thesis is deceptively simple: **tokens are the new commodity of the AI era**. Just as the industrial revolution produced physical goods and the information age produced software, the AI age produces tokens, the fundamental unit of AI output (text, code, images, reasoning steps, agent actions). Huang used the word "token" more than 70 times in a nearly two-hour speech and introduced a provocative formula for the new economics:

$$\text{Revenue} = \text{Tokens per Watt} \times \text{Available Gigawatts} \tag{8.4.1}$$

*Remark* (Reading the token identity as an economist). Equation (8.4.1) is a rhetorical identity, not a literal accounting one, and it repays a moment's attention to units. "Tokens per watt" is a measure of hardware *efficiency* (how much output a chip yields per unit of electrical power drawn), while "available gigawatts" is a measure of *scale* (how much power the data center can run). Their product is therefore an output *quantity*—a volume (or rate) of tokens generated, not a dollar revenue. To recover revenue one must multiply by an average realized *price per token* and a *utilization* rate:

$$\text{Revenue} \approx \underbrace{(\text{price per token})}_{\text{demand side}} \times \underbrace{(\text{tokens per watt})}_{\text{efficiency}} \times \underbrace{(\text{watts available})}_{\text{scale}} \times (\text{utilization}).$$

Huang's formula deliberately suppresses the price and utilization terms to dramatize a strategic claim: if demand for inference is taken to be effectively unbounded, then the binding constraints on a "token factory" become energy and hardware efficiency rather than customers. For a business researcher the suppressed factors are precisely the interesting ones—how token prices are set, how utilization responds to demand, and whether "unbounded" demand survives contact with buyers' willingness to pay.

In this framing, data centers are no longer generic compute facilities, they are **"token factories"** whose primary output is tokens, just as power plants produce electricity. NVIDIA GPUs are the "generators" in these factories. Huang declared: *"The future data center is a token factory... Inference is your workloads and tokens are your new commodity. We have reached that moment, inference inflection has arrived."* He further argued that agentic AI would cause a massive explosion in token demand, potentially 100x compared to simple query-response AI, because each agent action (planning, tool calling, reasoning, self-correction) generates and consumes tokens across extended loops.

**Alibaba Token Hub (ATH)** provides a striking real-world illustration of this vision. On March 16, 2026, the same week as Huang's GTC keynote, Alibaba Group es-

tablished ATH as a new **top-level Business Group** directly led by CEO Eddie Wu (Wu Yongming), elevating it to first-tier status alongside Alibaba Cloud and e-commerce.[56] ATH consolidates five previously separate AI units into a single organizational structure, shown in Table 8.4.1.

**Table 8.4.1.** The five units consolidated under Alibaba Token Hub (ATH).

| Unit | Role |
|---|---|
| Tongyi Laboratory | Develops Qwen foundation models (the "power plant" producing tokens) |
| MaaS (Model-as-a-Service) | Builds the inference infrastructure platform (the "transmission network") |
| Qwen Business Unit | Consumer-facing personal AI assistant (100M+ MAU) |
| Wukong Business Unit | B2B agentic work platform for enterprise workflows (newly created) |
| AI Innovation Business Unit | Explores emerging agentic consumer services (food ordering, travel, payments) |

Eddie Wu's mission statement for ATH is crystalline: **"Create tokens, deliver tokens, and apply tokens."** He framed the reorganization using an electrical grid analogy, Tongyi Lab is the power plant (producing tokens/intelligence), the MaaS platform is the transmission network (distributing tokens), and consumer/enterprise products (Qwen, Wukong) are the appliances consuming tokens. In his internal letter, Wu wrote: *"We are standing at the threshold of an AGI inflection point. Billions of AI agents are poised to take on an ever-greater share of digital work, each powered by tokens generated by models."*

The financial commitment is staggering: Alibaba has pledged **RMB 380 billion (∼$53 billion)** for cloud and AI infrastructure over three years, nearly matching the company's total capital expenditure over the previous decade.[57] AI inference already drives 60-70% of new Alibaba Cloud revenue, with AI-related products achieving triple-digit growth for nine consecutive quarters.

The convergence between Huang's token factory vision and Alibaba's organizational restructuring is not coincidental. It reflects a broader industry consensus: **the business model of the AI era is shifting from software-as-a-service (SaaS) to tokens-as-a-service**. Instead of monthly subscription fees, enterprises will pay per volume of tokens processed, analogous to a utility bill for electricity. The Chinese-language press described ATH as Alibaba's play to capture **"pricing power and minting rights in the**

[56] See SCMP (2026); Alizila (2026); and TechNode (2026).
[57] Alibaba Cloud (2026).

**AI era" (*AI shidai zhubiquan*)**, positioning tokens as the central currency of the AI value chain.

> **Why this matters for business researchers:** The token economy redefines how value is created and captured in AI. The fundamental unit of production shifts from code (software) to tokens (AI inference). Competitive advantage depends on cost per token, token throughput, and the ability to orchestrate token-consuming agents at scale. Alibaba Token Hub, with its vertical integration from chips to models to consumer applications, exemplifies the emerging organizational form optimized for this new economy.

### 8.4.4 Approachability vs. Deep Work

OpenClaw crystallizes a fundamental tension in agent design: **approachability vs. deep work**.

- **Chat-based interfaces** (WhatsApp, Telegram) are maximally approachable, everyone already knows how to send a message. But messaging apps impose severe constraints: no structured code editing, no version control integration, no persistent file system navigation.
- **IDE-based interfaces** (Claude Code, Cursor, VS Code) support deep, sustained technical work with full tooling. But they require technical sophistication and are inaccessible to non-developers.

OpenClaw chose approachability, betting that the messaging interface, despite its limitations for complex development tasks, would drive adoption by lowering the barrier to entry for the broadest possible user base.

### 8.4.5 Unified Context

A distinctive feature of OpenClaw is its **unified context pool**: it mixes inputs from multiple messaging platforms (WhatsApp, Telegram, Discord, etc.) into a single data stream that the LLM agent can reason over. This means an instruction sent via WhatsApp can reference a document shared via Discord, and the agent maintains coherent context across all channels.

This raises a critical technical challenge: **how to deal with context window length limitations?** As conversations accumulate across multiple platforms, the token count grows rapidly. OpenClaw addresses this through its memory architecture (see Section 8.4.6) and aggressive context management.

### 8.4.6 Self-Evolving Memory Engine

OpenClaw implements a tiered, file-based memory architecture, summarized in Table 8.4.2.

**Table 8.4.2.** OpenClaw's tiered, file-based memory architecture.

| File | Purpose |
|---|---|
| `SOUL.md` | Personality and behavioral guidelines, the agent's "character" |
| `USER.md` | User profile and preferences, learned over time |
| `MEMORY.md` | Long-term knowledge storage, facts, procedures, project state |

A background **heartbeat mechanism** automatically reviews recent interaction logs, distills valuable information into `MEMORY.md`, and cleans up outdated entries. This creates a *self-curating* knowledge base that evolves with each interaction.

The memory system is "blackboxed" in the sense that its internal state is opaque to the user, the agent decides what to remember and what to forget. This raises important questions about transparency and user control over agent behavior.

### 8.4.7 The OpenClaw Flywheel

The three core design elements, **persistent memory**, **unified context**, and **ecosystem of skills**, form a mutually reinforcing flywheel, depicted in Figure 8.4.1.

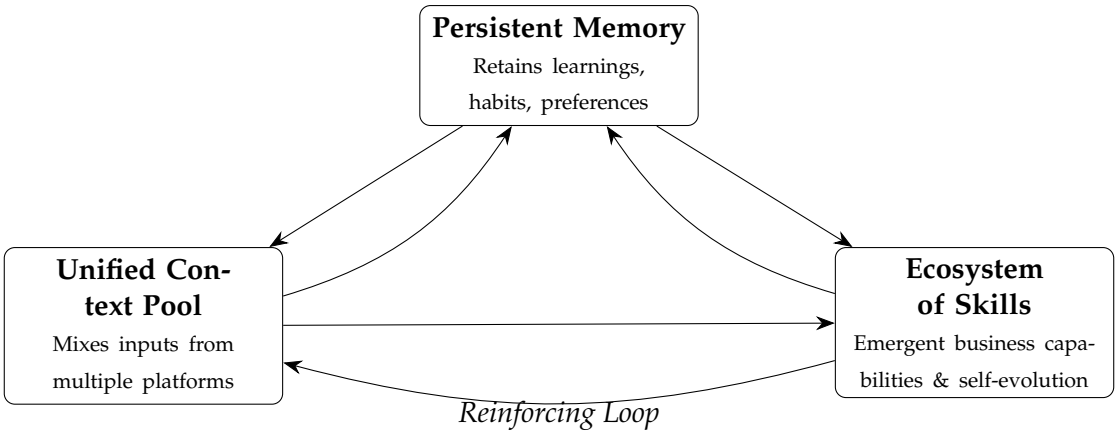


**Figure 8.4.1.** The OpenClaw flywheel: persistent memory, unified context, and an ecosystem of skills mutually reinforce one another, enabling continual learning without explicit retraining.

- **Memory + Context:** The more platforms the agent monitors, the richer its memory becomes; richer memory improves contextual understanding across platforms.

- **Memory + Skills:** The agent can write its own skills, if no existing tool solves a problem, it writes code, saves it as a reusable skill, and adds it to its repertoire. This creates a self-evolution loop.
- **Skills + Context:** New skills unlock new capabilities, which generate new types of interactions, which feed back into context and memory.

These three subsystems achieve **continual learning**: the agent improves over time without explicit retraining,[58] purely through accumulated experience and self-authored tooling.

### 8.4.8 The Lethal Trifecta

OpenClaw's power comes with proportional risk. The system exhibits what security researchers call the **lethal trifecta**[59] of agentic AI:[60]

1. **Access to private data:** The agent can read messages, files, browsing history, and other sensitive information across all connected platforms.
2. **Exposure to untrusted environments:** The agent processes inputs from the open internet, messaging platforms, and third-party skill repositories, all potential vectors for adversarial manipulation.
3. **Ability to take autonomous actions:** The agent can send messages, execute code, modify files, and interact with external services without human approval for every step.

When all three are present simultaneously, a single vulnerability can cascade into catastrophic compromise. This is not hypothetical: by mid-February 2026, multiple critical vulnerabilities had been disclosed:

[58] It is worth being precise about what is and is not "learning" here. The LLM's *weights*—the billions of parameters fixed during pretraining—never change: there are no gradient updates, no backpropagation, no new training run. What changes is the *external state* that the frozen model reads at inference time—the memory files, the accumulated skills, the curated context. This is *in-context* (non-parametric) adaptation rather than *parametric* learning, and it is reversible and inspectable in a way that weight updates are not. The practical upshot is that an agent can appear to "learn" a user's preferences overnight without anyone retraining a model.

[59] The term was coined by the developer Simon Willison in June 2025; see https://simonwillison.net/2025/Jun/16/the-lethal-trifecta/. In Willison's original formulation the three ingredients are (i) access to private data, (ii) exposure to untrusted content, and (iii) the ability to *externally communicate* (to send data outward, enabling *exfiltration*). The third item here is stated more broadly as the ability to take autonomous actions, of which data exfiltration is the canonical dangerous case.

[60] See the Acronis security analysis, https://www.acronis.com/en/tru/posts/openclaw-agentic-ai-in-the-wild-architecture-adoption-and-emerging-security-risks/; the Yage.ai deep dive, https://yage.ai/openclaw-en.html; and the OpenClaw GitHub, https://github.com/openclaw/openclaw.

- **CVE-2026-25253** (CVSS 8.8)[61]: Gateway compromise allowing arbitrary command execution.
- **CVE-2026-24763 and CVE-2026-25157:** Command injection vulnerabilities.
- **ClawHavoc:** A supply-chain poisoning campaign in which over 340 malicious skills were uploaded to ClawHub (the official skill marketplace), many posing as productivity tools but installing malware.

### 8.4.9 Reimplementation: Reconstructing the Magic Safely

OpenClaw's viral success demands massive compromises in security and control. However, its core innovations can be **reconstructed within a more controllable architecture**. The reimplementation strategy involves four key decisions:

**1. Outsource the agentic loop to mature platforms.** Rather than building a custom agent runtime, leverage established coding agent platforms (Claude Code, Cursor, or similar) that have invested heavily in sandboxing, permission systems, and security hardening. These platforms provide the agentic loop, observation, reasoning, action, feedback, with enterprise-grade safeguards.

**2. Redesign the memory architecture.** Replace OpenClaw's blackboxed, opaque memory with a **disk-as-memory** approach backed by a **Git mono-repo**.[62] This restores fine-grained control over knowledge assets and context, as contrasted in Table 8.4.3.

**3. Mono-repo solves context pollution.** By organizing each project or workspace as a separate repository (or directory within a mono-repo), the agent's context is naturally scoped to the relevant domain. This prevents the "context pollution" problem where unrelated information from different platforms leaks into the agent's reasoning.

**4. Secure the skills supply chain.** Blindly installing third-party MCP servers risks catastrophic supply-chain attacks. The reimplementation approach favors writing skills in-house, a task that takes only minutes in the age of agentic AI coding, reducing

[61] *CVE* (Common Vulnerabilities and Exposures) is a public catalogue that assigns every disclosed security flaw a unique identifier of the form `CVE-YYYY-NNNNN`, so that vendors and researchers can refer to the same bug unambiguously. *CVSS* (Common Vulnerability Scoring System) rates a flaw's severity on a 0–10 scale; a score of 8.8 sits in the "high" band, indicating a vulnerability that is both relatively easy to exploit and damaging in its consequences.

[62] *Git* is the dominant version-control system for software: it records a project's entire history as a sequence of snapshots ("commits"), and a *diff* displays exactly which lines changed between any two snapshots, so every edit is attributable and reversible—much like a tamper-evident lab notebook or an audit trail. A *mono-repo* (monolithic repository) is a single Git repository that houses many projects or workspaces together. Replacing the agent's opaque, self-rewriting `MEMORY.md` with version-controlled files thus makes the memory's evolution transparent, diff-able, and auditable.

**Table 8.4.3.** Contrasting OpenClaw's memory architecture with a controllable reimplementation.

| OpenClaw | Reimplementation |
|---|---|
| Opaque `MEMORY.md` managed by heartbeat | Version-controlled files in a Git repository |
| Blackboxed memory evolution | Transparent, auditable, diff-able history |
| Platform-mixed context | Workspace-isolated context per project |

execution risk to effectively zero. When third-party skills are necessary, they should be audited, sandboxed, and version-pinned.

> **Key lesson for researchers:** OpenClaw demonstrates that the *interface* and *memory architecture* of an agent system matter as much as the underlying LLM. The same model, wrapped in different scaffolding, can produce dramatically different user experiences, adoption curves, and risk profiles.

## 8.5 Context Engineering

### 8.5.1 Context Is the Key

*Context engineering* is the discipline of curating and maintaining the optimal set of information (tokens) delivered to a language model during inference (Lee, 2026; Rajasekaran et al., 2025).[63] It is the natural evolution of prompt engineering: while prompt engineering focuses narrowly on crafting effective instructions for specific tasks, context engineering addresses the *entire token management challenge* across multiple turns of inference in an agentic loop.

The distinction matters because agents operate in loops over extended periods. At each step, the agent must decide: *what information should be in the context window right now?* The universe of potentially relevant information, conversation history, tool outputs, retrieved documents, system instructions, grows monotonically, but the context window is finite. Context engineering is the art of managing this tension.

*Remark* (Guiding principle). Find the *smallest possible set of high-signal tokens* that maximizes the likelihood of the desired outcome (Rajasekaran et al., 2025).

[63] Three terms recur throughout this section and are worth fixing precisely for a reader new to language models. A *token* is the unit into which the model chops text—roughly a short word or word-fragment (a useful rule of thumb is that one token is about four characters, or three-quarters of an English word). *Inference* here means running the already-trained model forward to produce output (a single "forward pass"), *not* statistical inference: the model's parameters are held fixed and nothing is being estimated. The *context window* is the maximum number of tokens the model can read in one pass—its finite working memory—so the system prompt, the conversation history, tool outputs, and any retrieved material must all fit inside it.

### 8.5.2 Why Context Engineering Matters: Context Rot

The theoretical basis for context engineering lies in the transformer's self-attention mechanism (Vaswani et al., 2017). Every token in the context window attends to every other token, creating $n^2$ pairwise relationships for $n$ tokens. As $n$ grows, the model's ability to capture these relationships gets stretched thin, a fundamental consequence of the quadratic attention architecture introduced by Vaswani et al. (2017).[64]

Empirical research on "needle-in-a-haystack" benchmarks[65] has uncovered a phenomenon called *context rot* (Rajasekaran et al., 2025): as the number of tokens in the context window increases, the model's ability to accurately recall and reason about information from that context *decreases*. This is not a cliff but a gradient, models remain capable at long contexts but show reduced precision for information retrieval and long-range reasoning compared to shorter contexts.

The implication is clear: **context is a finite resource with diminishing marginal returns**. Adding more information to the context window is not always better; it can actively hurt performance if the added tokens are low-signal or dilute attention from high-signal content.

### 8.5.3 Anatomy of Effective Context

Anthropic's framework for effective context engineering identifies several key components (Rajasekaran et al., 2025).

**Agents as LLMs in a loop.** Anthropic defines agents as "LLMs autonomously using tools in a loop." This minimal definition emphasizes that the agentic loop, not any particular architecture or framework, is the essential structure. Smarter models allow agents to more independently navigate nuanced problem spaces and recover from errors, but the loop structure is universal.

[64] Intuition for the non-specialist. Self-attention updates each token's internal representation as a *weighted average* of the representations of all tokens in the window, where the weights measure learned pairwise relevance—mechanically very close to a kernel-weighted (Nadaraya–Watson) average, except that the kernel is learned from data rather than fixed in advance. With $n$ tokens there are $n \times n$ such weights, which is the source of the "quadratic" ($n^2$) cost in compute and memory. Two cautions for precision. (i) The $n^2$ scaling is a statement about *computational cost*; it is not a theorem that accuracy must fall as $n$ grows. The recall degradation described next is an *empirical* regularity, not a mathematical consequence of the architecture. (ii) The term *context rot* and its systematic measurement are due to a Chroma technical report (K. Hong et al., *Context Rot: How Increasing Input Tokens Impacts LLM Performance*, 2025, `https://research.trychroma.com/context-rot`), which the Anthropic guidance cited here summarizes.

[65] A *needle-in-a-haystack* test inserts a single target fact (the "needle") at some position inside a long block of otherwise-irrelevant text (the "haystack") and then asks the model to retrieve or use it. Accuracy is tracked as the haystack lengthens and as the needle's position varies, isolating how much a model's recall degrades purely because the surrounding context is large.

**The three pillars of context design.**

1. **System prompts** define the agent's *altitude*, how it should approach tasks, what constraints apply, what persona it should adopt.
2. **Tools** define the strict contract between agents and their information/action space, what the agent *can* do and how it learns about the world.
3. **Examples** (few-shot in-context learning)[66] communicate behavioral patterns more efficiently than verbose rules, they are the "pictures worth a thousand words" for LLMs.

### 8.5.4 System Prompts: Finding the Right Altitude

System prompts define the *altitude* at which an agent operates (Rajasekaran et al., 2025). Two failure modes must be avoided:

- **Overly specific (too low):** Hardcoding complex, brittle if-else logic that attempts to enumerate every edge case. This creates fragile systems that break at the first unanticipated input.
- **Overly vague (too high):** Providing vague guidance that assumes the model shares the developer's implicit understanding. This leads to inconsistent, unpredictable behavior.

The optimal altitude is **specific enough to guide behavior effectively, yet flexible enough to provide strong heuristics** that generalize across situations.

**Best practices for system prompt design.**

- Organize prompts into distinct sections using XML tags or Markdown headers:
  - `<background_information>`, domain context and objectives
  - `<instructions>`, behavioral guidelines and constraints
  - `### Tool guidance`, when and how to use specific tools
  - `## Output description`, format and structure of expected outputs
- Start with a minimal prompt on the strongest available model.

[66] *In-context learning* is the empirical observation that a language model can adapt its behavior from a handful of worked examples placed directly in the prompt, *without* any change to its parameters—no gradient step and no re-estimation. "Few-shot" simply means a small number of such examples are supplied (as opposed to "zero-shot," where only instructions are given). For the empirical researcher the closest analogy is conditioning a prediction on a few labeled cases at decision time, rather than refitting the model on those cases.

- Iteratively add clarifications based on observed failure modes, never preemptively over-specify.
- Use canonical examples rather than exhaustive rule lists. A curated set of diverse examples that portray expected behavior is more effective than paragraphs of instructions.

### 8.5.5 Token-Efficient Tool Design

Tools define the contract between agents and their action space. The design of tools has a direct impact on agent performance and token efficiency (Rajasekaran et al., 2025):

- **Self-contained and robust to error.** Each tool should handle edge cases gracefully and return informative error messages.
- **Clear intended use with minimal overlap.** If a human engineer cannot definitively say which tool applies in a given scenario, an AI agent cannot do better. Bloated toolsets with overlapping functionality are a common failure mode.
- **Descriptive, unambiguous input parameters.** Use `user_id` rather than `user`; `start_date` rather than `date`.
- **Token-efficient outputs.** Implement pagination, filtering, and truncation with sensible defaults for any tool that could return large responses. A tool that dumps 10,000 tokens of raw data when the agent needs a 50-token summary is wasting the context budget.

The key insight is that tools should be designed as if writing documentation for a new hire: **make all implicit knowledge explicit** (specialized query formats, niche terminology, resource relationships).

### 8.5.6 Progressive Information Retrieval vs. RAG

A critical design choice for agentic systems is *how* the agent obtains information (Rajasekaran et al., 2025).

**Traditional RAG (Retrieval-Augmented Generation).** Embedding-based retrieval[67] surfaces all potentially relevant context *upfront*, before the model begins generating. This is fast but rigid, the retrieval query is formulated before the model has had a chance to reason about what it actually needs.

[67] An *embedding* maps a piece of text to a dense vector in $\mathbb{R}^d$ so that semantically similar texts land near one another in that space—much like a learned low-dimensional latent representation in factor analysis, but estimated by a neural network. "Embedding-based retrieval" then answers a query by embedding the query and returning the stored text chunks whose vectors are nearest to it: a nearest-neighbor search in semantic space rather than a literal keyword match.

**Just-in-time agentic retrieval.** The agent maintains lightweight identifiers (file paths, stored queries, URLs) and dynamically loads data using tools *at runtime*. This mirrors human cognition, we do not memorize entire libraries but rather develop indexing systems (file folders, bookmarks, mental models of where to look) that enable efficient on-demand retrieval.

**Table 8.5.1.** Traditional RAG versus just-in-time agentic retrieval.

| Dimension | Traditional RAG | Just-in-Time Retrieval |
|---|---|---|
| Speed | Faster (pre-computed) | Slower (runtime exploration) |
| Flexibility | Less adaptive | Highly adaptive |
| Context efficiency | May over-retrieve | Retrieves only what is needed |
| Staleness risk | Index may be outdated | Always reads current data |
| Complexity | Simpler pipeline | Requires thoughtful tool design |

**The hybrid approach (exemplified by Claude Code).**

- Pre-load high-signal static information (e.g., `CLAUDE.md` files) into context upfront.
- Use targeted tools (`glob`, `grep`, `read`) for just-in-time file navigation and content retrieval.
- Bypass issues of stale indexing and complex syntax trees.

This hybrid strategy strikes a practical balance: immediate context for known-relevant information, combined with autonomous exploration for dynamic or large-scale content.

### 8.5.7 Architecting for Long Horizons

Long-horizon tasks, those requiring sustained agent effort over many turns of the agentic loop, present a fundamental challenge: the token count of accumulated context eventually approaches or exceeds the model's context window. Three complementary mechanisms address this (Rajasekaran et al., 2025).

#### *8.5.7.1 Compaction: Distilling the State*

*Compaction* involves summarizing a conversation that has reached context limits and restarting with the compressed summary. It is the most direct lever for extending agent coherence over long interactions.

**Implementation pattern (as in Claude Code).**

1. Pass the full message history to the model for summarization.
2. Preserve architectural decisions, unresolved bugs, and critical implementation details.
3. Discard redundant tool outputs, stale intermediate results, and superseded reasoning.
4. Continue with compressed context plus the most recently accessed files.

**The art of compaction.** It lies in selecting what to preserve versus what to discard. Overly aggressive compaction risks losing subtle but critical context whose importance only becomes apparent later. The recommended approach:

- **Maximize recall first:** Capture everything that might be relevant.
- **Iterate to improve precision:** Eliminate superfluous content through testing and observation.

A lightweight form of compaction that is nearly always safe is **clearing old tool results** from message history. Agents rarely need the raw output of tools called deep in the conversation, the *conclusions* drawn from those outputs, already embedded in the agent's subsequent reasoning, are sufficient.

#### *8.5.7.2 Agentic Memory: Structured Note-Taking*

Rather than relying solely on the context window, agents can **write externally-persisted notes** that are retrieved at later times (Rajasekaran et al., 2025). This provides persistent memory with minimal context overhead.

**Patterns include:**

- **To-do lists:** Tracking progress across complex, multi-step tasks.
- **Notes files:** Maintaining `NOTES.md` or similar structured documents with critical dependencies, decisions, and status.
- **Memory files:** Building knowledge bases over time that persist across sessions.

The structured note-taking approach is particularly effective for iterative development with clear milestones. The agent writes notes at natural checkpoints, and future turns (or even future sessions) can read these notes to reconstruct context without replaying the full conversation history.

**Example 8.5.1** (Claude playing Pokémon)**.** Claude playing Pokémon maintains precise tallies across thousands of game steps, tracking training progress, discovered areas, and strategic combat notes. After context resets, the agent reads its own notes and continues multi-hour gameplay sequences seamlessly (Rajasekaran et al., 2025).

#### *8.5.7.3 Sub-Agent Architectures*

Rather than one agent maintaining state across an entire project, **specialized sub-agents** handle focused tasks with clean context windows (Rajasekaran et al., 2025):

- A **lead (orchestrator) agent** coordinates at a high level: planning, delegating, and synthesizing.
- **Sub-agents** perform deep technical work, research, analysis, code generation, within their own isolated context windows.
- Each sub-agent may consume tens of thousands of tokens in exploration but returns only a condensed 1,000-2,000 token summary to the orchestrator.

This architecture provides:

- **Clear separation of concerns:** Detailed search context remains isolated within sub-agents.
- **Context efficiency:** The lead agent's context window contains only high-level plans and sub-agent summaries, not the raw details of every exploration.
- **Parallelism:** Multiple sub-agents can work simultaneously on independent sub-tasks.

Research on multi-agent research systems has shown substantial improvements over single-agent approaches on complex analysis tasks (Rajasekaran et al., 2025). The orchestrator-sub-agent pattern is depicted in Figure 8.5.1.

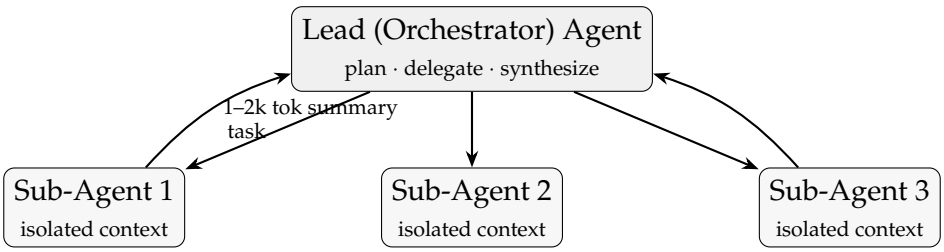


**Figure 8.5.1.** Sub-agent architecture. A lead agent delegates focused tasks to sub-agents, each of which explores within an isolated context window and returns only a condensed summary, keeping the orchestrator's context lean.

**Table 8.5.2.** Choosing among long-horizon context mechanisms.

| Mechanism | Best For |
|---|---|
| Compaction | Tasks requiring extensive back-and-forth interaction |
| Structured note-taking | Iterative development with clear milestones |
| Sub-agent architectures | Complex research/analysis where parallel exploration pays dividends |

**Choosing the right mechanism.** Table 8.5.2 summarizes when each long-horizon mechanism is most appropriate.

### 8.5.8 Context Engineering Capability Map

The full context engineering capability map can be organized along two dimensions (Rajasekaran et al., 2025):

1. **Temporal dimension:** From single-turn static context design (system prompts, tools, examples) through multi-turn dynamic retrieval to long-horizon context management (compaction, memory, sub-agents).

2. **Autonomy dimension:** From engineer-curated static context (manually written prompts and examples) through hybrid approaches (pre-loaded context plus autonomous retrieval) to fully autonomous agent-driven exploration.

Table 8.5.3 arranges representative techniques on this $3 \times 3$ grid.

**Table 8.5.3.** Context engineering capability map: temporal stage (columns) against autonomy level (rows).

| | Static Context | Dynamic Retrieval | Long-Horizon Management |
|---|---|---|---|
| **Engineer-Curated** | System prompts, tool definitions, few-shot examples | Traditional RAG, pre-computed indices | Manual checkpointing, session summaries |
| **Hybrid** | `CLAUDE.md` files, structured configs | Hybrid retrieval (static + tools) | Compaction with preserved landmarks |
| **Fully Autonomous** | Agent-generated prompts and tools | Just-in-time agentic exploration | Sub-agents, memory, self-evolving notes |

As models become more capable, the optimal point on this map shifts toward greater autonomy, but thoughtful context curation remains central regardless of model intelligence.

### 8.5.9 Formal Representation of Context Engineering

Context engineering can be expressed in the following abstract format (Lee, 2026; Rajasekaran et al., 2025). Let $P_t$ denote the set of information (tokens) delivered to the

context window of the LLM at time step $t$ of the agentic loop. The agent's behavior at each step is determined by

$$\text{action}_t = \text{LLM}(P_t), \tag{8.5.1}$$

where $P_t$ includes the system prompt, tool definitions, conversation history (possibly compacted), retrieved information, memory contents, and any other tokens present in the context window.

The **context engineering problem** is to design a policy for constructing $P_t$ that maximizes the probability of achieving the desired outcome:

$$\max_{P_t} \Pr(\text{desired outcome} \mid P_t) \quad \text{subject to} \quad |P_t| \leq C, \tag{8.5.2}$$

where $C$ is the context window size (in tokens) and $|P_t|$ denotes the token count of $P_t$.

The constraint $|P_t| \leq C$ in (8.5.2) is *hard*, exceeding the context window causes truncation or failure. But the optimization is also subject to a *soft* constraint: context rot implies that the effective information capacity of the context window is less than $C$. Adding low-signal tokens to $P_t$ can *reduce* $\Pr(\text{desired outcome} \mid P_t)$ even if $|P_t| < C$, because they dilute attention from high-signal content.

The context window grows easily, every observation, tool output, and reasoning step adds tokens, but the information budget is fixed. The central challenge of context engineering is managing this asymmetry: an ever-expanding universe of potentially relevant information filtered through a finite, degradation-prone window.

*Remark* (Reading (8.5.2) as a constrained optimization). For a reader trained in constrained optimization, (8.5.2) is unusual in two instructive ways. First, the choice variable $P_t$ is a *set* of tokens, so this is a combinatorial selection problem—closer in spirit to a knapsack than to a smooth program: each candidate token carries an information "value" and a unit token "cost," and we choose a high-value subset subject to the budget $C$. Second, and more importantly, the objective is *not monotone* in $P_t$. The standard "free-disposal" intuition—that adding an input can never make you worse off, since you can always ignore it—fails here: context rot means that extra low-signal tokens can actually lower $\Pr(\text{desired outcome} \mid P_t)$ by diluting attention. Consequently the binding constraint is rarely the hard budget $C$ itself; the operative scarcity is the *shadow price of attention*, the opportunity cost of spending finite, high-quality attention on the wrong tokens. This is precisely why "add more context" is not a free lunch, and why the guiding principle stated in Section 8.5.1 is to *minimize* the token set subject to achieving the goal, rather than to maximize the information supplied.

*Remark* (The key question). How should we properly design $P_t$? This question does not have a universal answer. The optimal $P_t$ depends on the task, the model, the available tools, and the interaction history. Context engineering is as much an empirical art as a

theoretical discipline, and it is the defining skill for building effective AI agents.

## 8.6 Harness Engineering

### 8.6.1 From Context Engineering to Harness Engineering

Context engineering addresses *what information* to deliver to the model. But building reliable, production-grade agents requires a broader discipline: *harness engineering*, the design of the entire infrastructure that wraps around the model to make it reliable, repeatable, and safe at scale.[68,69]

The core equation is

$$\text{Agent} = \text{Model} + \text{Harness}. \tag{8.6.1}$$

The model generates responses. The harness handles *everything else*: environment design, feedback loops, permission management, tool orchestration, evaluation, and state persistence. If context engineering asks "how should we design $P_t$?", harness engineering asks the broader question: **how should we design the entire system that produces, evaluates, and acts on $P_t$?**

*Remark* (Core insight). The primary job of an engineer is no longer writing syntax. It is designing environments, specifying intent, and building feedback loops. When an agent fails, the fix is never to ask it to "try harder." The engineering response must be: *what capability is missing from the system? How do we make the environment legible and enforceable?*

### 8.6.2 The Engineering (and Research) Paradigm Shift

Harness engineering implies a fundamental paradigm shift in how software, and research, is produced. Table 8.6.1 contrasts the traditional, human-centric workflow with the emerging *harness (agent-first)* paradigm.[70]

**Table 8.6.1.** The engineering paradigm shift: traditional versus harness (agent-first).

| Dimension | Traditional | Harness (Agent-First) |
|---|---|---|
| What to do | Writing logic and syntax | Scaffolding systems and creating leverage |
| How to review and iterate | Human-to-human code review | Agent-to-agent review; humans review final outcomes |
| Quality gates | Heavy, blocking, manual QA gates | High-throughput, short-lived PRs; corrections cheap and fast |
| Optimization target | Optimized for human style and taste | Optimized for correctness, machine legibility, and invariants |

Several implications follow:

[68] https://openai.com/index/harness-engineering/

[69] https://www.anthropic.com/engineering/harness-design-long-running-apps

[70] See https://openai.com/index/harness-engineering/; https://www.anthropic.com/engineering/harness-design-long-running-apps; and Karpathy's *Autoresearch*, https://github.com/karpathy/autoresearch.

1. **The developer's role shifts from author to architect.** Rather than writing code line-by-line, the engineer designs the *environment* in which agents produce code, defining constraints, success criteria, and feedback mechanisms. The harness externalizes implicit expertise (coding conventions, design judgment, organizational alignment) into explicit, enforceable controls.

2. **Review becomes outcome-oriented.** In the traditional model, humans review *process* (code style, naming conventions, architectural patterns). In the harness model, agents review each other's work against formal criteria, and humans review *outcomes*, does the system behave correctly? This mirrors the generator-evaluator pattern: separate the agent doing the work from the agent judging it.

3. **Iteration becomes cheaper.** When corrections are cheap and fast, short-lived pull requests[71], automated testing, rapid re-generation, the optimal strategy shifts from "get it right the first time" to "iterate quickly against concrete feedback." This has direct parallels to automated research workflows (e.g., Karpathy's *Autoresearch*), where the research loop of hypothesize-experiment-analyze-revise is accelerated by agent-driven iteration.

4. **Machine legibility over human aesthetics.** Code optimized for agent consumption prioritizes correctness, invariants, and legibility to automated tools over stylistic preferences. Strong typing, clear module boundaries, and established frameworks, what practitioners call *ambient affordances*, determine how effectively a codebase can be harnessed.

### 8.6.3 Provide Agents a Map, Not a Manual

A foundational principle of harness engineering is: **what agents cannot see does not exist**.[72] This means that the *information structure* provided to agents is as important as the information itself.

Two contrasting approaches illustrate the design space, as summarized in Table 8.6.2.

The winning approach treats agent instructions not as exhaustive manuals but as **navigational maps**, lightweight indexes that point to deeper, authoritative sources. The agent consults the map, identifies what is relevant, and retrieves detailed infor-

[71] A *pull request* (PR) is the standard unit of proposed change in collaborative software development: a self-contained, reviewable bundle of edits that is submitted for inspection and then merged into the shared codebase. A "short-lived" PR is a small change opened and merged quickly, which keeps each review cheap and each correction fast—the same logic by which small, frequent experiments dominate one large irreversible one.

[72] See https://openai.com/index/harness-engineering/; https://www.anthropic.com/engineering/harness-design-long-running-apps.

**Table 8.6.2.** The failed (manual) approach versus the winning (map) approach to agent instructions.

| | The Failed Approach | The Winning Approach |
|---|---|---|
| Format | 1,000-page `AGENTS.md` | 100-line table of contents |
| Context impact | Crowds out task context | Injected directly into context |
| Signal quality | When everything is "important," nothing is | Acts purely as pointers to deeper, verifiable truth |
| Maintenance | Rots instantly; an attractive nuisance | Agents navigate intentionally and explore as needed |

mation on demand. This is, in essence, the progressive disclosure principle applied to the entire agent operating environment.

The implications for practitioners are concrete:

- **Keep top-level instructions concise.** The `AGENTS.md` or `CLAUDE.md` file should be a table of contents, not an encyclopedia. Reserve the context window for the task at hand.
- **Make sources of truth verifiable.** Pointers should lead to files, documentation, or code that the agent can read and verify, not to stale summaries that may have drifted from reality.
- **Design for intentional navigation.** Structure information so that agents can efficiently find what they need without loading everything upfront. This aligns with the just-in-time retrieval strategy discussed in Section 8.5.6.

*Remark.* Carefully manage the information structure of your agents. The difference between a productive agent and a confused one is often not the model or the prompt, it is whether the right information is *discoverable* and *accessible* at the moment of need.

### 8.6.4 Context, Context, and Context

If harness engineering reframes the engineer's role, it also reframes what practitioners should *learn*. The single most important practical takeaway of this lecture can be stated sharply: **stop studying prompt engineering as a primary craft**.[73] Prompt engineering optimizes the surface form of a single instruction; in a world where agents run long, multi-step, tool-augmented loops, the binding constraint is not the wording of any one prompt but the *informational environment* in which the agent operates.

The productive replacement is disciplined attention to four levers:

1. **Provide missing facts and precise specifications.** Most agent failures are not failures of intelligence, they are failures of specification. The agent cannot infer a constraint it was never told, nor can it respect an invariant it cannot see. The engineer's

[73] See https://openai.com/index/harness-engineering/; https://www.anthropic.com/engineering/harness-design-long-running-apps.

job is to *close the specification gap*: surface hidden assumptions, name the edge cases, and state requirements in terms the agent can verify against.

2. **Design the informational environment.** A useful environment specifies, at minimum, three elements:
   - **Goals.** What outcome is being optimized? What counts as done?
   - **Guardrails.** What actions are out of bounds? What invariants must hold throughout execution?
   - **Success criteria.** How will the agent (and the human reviewer) *know* the task has been completed correctly, ideally in a machine-checkable form (tests, schemas, assertions) rather than subjective judgment?
3. **Make the assets reusable.** One-shot prompts decay the moment the task ends; skills, reference documents, evaluation harnesses, and checklists compound over time. Treat every well-designed context artifact as a reusable asset, versioned, documented, and discoverable by future agents, so that the marginal cost of the next task falls with each task completed.
4. **Optimize the information structure, not the phrasing.** Given goals, guardrails, and success criteria, the remaining engineering work is structural: what is loaded into the system prompt, what is retrieved just in time, what is offloaded to files or sub-agents, and what is compacted away. This is precisely the context-engineering discipline of Section 8.5, now elevated from a tactical skill to the *core* skill.

*Remark* (The shift in a sentence). The question is no longer "how do I phrase this prompt?" but "how do I *shape the information* the agent sees, goals, guardrails, success criteria, reusable assets, so that the right behavior becomes the path of least resistance?"

### 8.6.5 Become an AI Manager

The natural endpoint of this progression, from prompt engineering to context engineering to harness engineering, is a change in the human role itself. The engineer, researcher, or executive working with capable agents increasingly functions less as an individual contributor and more as a **manager of AI workers**.[74]

The analogy is substantive, not metaphorical. Modern agents, like human subordinates, **operate on context and creativity, are unpredictable in detail, but can handle genuine complexity** when properly managed. The management disciplines that

[74] See https://openai.com/index/harness-engineering/; https://www.anthropic.com/engineering/harness-design-long-running-apps.

produce good human teams therefore transfer, with only modest adaptation, to the management of agentic systems. Three practices are central:

1. **Delegate with clarity.** Effective delegation to an agent requires the same two ingredients as effective delegation to a person: a clear objective and a built-in mechanism for the delegate to verify its own work. In the agentic setting, "self-check measures" take concrete forms, unit tests the agent must run, schemas its output must satisfy, invariants it must assert, or sub-agent critics that review its work before it returns. Tasks delegated without self-check are tasks whose quality the manager must verify by hand, which does not scale.

2. **Evaluate with confidence.** Managing agents at scale requires the same evidentiary discipline as managing experiments: track runs, measure outcomes against pre-specified criteria, make data-driven decisions about which agents, skills, and harness configurations are working, and report progress in terms that decision-makers can act on. Evaluation is not a post-hoc activity; it is the feedback loop that allows the manager to improve the system over time.

3. **Coach for growth.** The most valuable managerial act is to convert a successful one-off coaching interaction, "here is why the agent got this wrong, and here is how I'd rather it approached similar cases", into a **reusable SOP**: a skill, a reference document, a guardrail, or an entry in `AGENTS.md`. Institutional memory for AI workers is explicit and externalized; the manager's job is to make sure hard-won lessons are written down in a form that future agents will actually read and follow.

*Remark* (The capstone principle). The frontier of productivity with agentic AI belongs to practitioners who think like managers, delegating with clarity, evaluating with evidence, and coaching institutional knowledge into reusable artifacts, rather than to those who think like power users of a chatbot. For business researchers in particular, this is a familiar skill set being applied to a new kind of worker.

## 8.7 Conclusion

Agentic AI represents the next frontier in applied artificial intelligence, the transition from models that generate text to systems that autonomously plan, act, and learn in the world. The key themes of this lecture are:

1. **From LLMs to agents.** Equipping an LLM with an agentic loop, observation, reasoning, action, feedback, transforms it from a passive text generator into an active problem-solver. The critical enablers are tool use, memory, and task decomposition.

2. **Skills and MCP as infrastructure.** Skills provide modular SOPs that make agents effective at specific tasks. Three archetypes, document creation, workflow automation, and MCP enhancement, cover the primary use cases. The progressive disclosure architecture ensures token efficiency by loading skill content in three tiers (metadata, instructions, resources). Clear frontmatter descriptions drive reliable triggering, and nonambiguous instructions bridge the gap between deterministic code and probabilistic language. The Model Context Protocol standardizes how agents connect to external tools and data, solving the $M \times N$ integration problem with an open, universal standard.

3. **OpenClaw and the token economy.** OpenClaw's viral success demonstrates the power of approachability (messaging-first interface), unified context (multi-platform data integration), and the flywheel of memory + skills + context. But it also exposes the lethal trifecta of agentic risk: private data access, untrusted input exposure, and autonomous action capability. More broadly, the emergence of agentic AI has catalyzed the *token economy*, Jensen Huang's vision of data centers as "token factories" producing inference as a commodity, with Alibaba Token Hub as a landmark organizational response: a top-level business group built around the mission to "create tokens, deliver tokens, and apply tokens."

4. **Context engineering as the defining discipline.** Building effective agents is less about model intelligence and more about the quality of information delivered to the model at each step. Context engineering, encompassing system prompt design, token-efficient tool definitions, progressive retrieval strategies, compaction for long horizons, structured note-taking for persistent memory, and sub-agent architectures for parallel exploration, is the skill that separates functional agents from unreliable ones.

5. **The formal challenge.** Context engineering can be framed as an optimization problem ((8.5.2)): maximize the probability of the desired outcome subject to the hard constraint of context window size and the soft constraint of context rot. The ever-expanding universe of relevant information, filtered through a finite and degradation-prone window, defines the central tension that practitioners must navigate.

6. **Harness engineering: from writing code to designing systems.** The emerging discipline of harness engineering extends context engineering to the entire infrastructure around the model, environment design, feedback loops, evaluation, and permission management. The paradigm shift is profound: the engineer's role moves from author to architect, review becomes outcome-oriented, iteration becomes cheap, and optimization targets shift from human aesthetics to machine

legibility and correctness. A key practical principle, provide agents a map, not a manual, ensures that information structure enables rather than overwhelms agent reasoning.

7. **Context over prompts; management over authorship.** The practical takeaway for practitioners is twofold. First, *stop learning prompt engineering as a primary craft*: the leverage lies in supplying missing facts and precise specifications, designing an informational environment of goals, guardrails, and success criteria, and making every context artifact a reusable asset. Second, *become an AI manager*: treat capable agents as unpredictable-but-competent workers who must be delegated to with clarity (objectives plus self-check measures), evaluated with evidence (tracked experiments and data-driven decisions), and coached for growth (successful interactions distilled into reusable SOPs). The frontier of productivity belongs to those who think like managers of AI workers, not power users of a chatbot.

*Remark* (The broader significance for business researchers). Agentic AI is already transforming how research is conducted, from automated literature reviews and data analysis to experimental design and paper writing. Understanding the architecture of agents (loops, tools, memory), the standards that connect them to the world (MCP), the risks they introduce (the lethal trifecta), the principles that make them effective (context engineering), and the systems engineering that makes them reliable (harness engineering) is essential for any researcher who wishes to leverage, or study, these systems.

The progression that organizes this lecture, from *prompt engineering* (crafting a single instruction) to *context engineering* (managing an evolving information state $P_t$ across an extended agentic loop) to *harness engineering* (designing the full system that produces, evaluates, and acts on that information state), defines the frontier of applied AI research and practice. Mastering it is the work of the coming decade.

# Chapter 9: What's New in AI (Spring 2026)

This chapter surveys frontier developments in artificial intelligence from January to March 2026, curated for PhD students across the business disciplines. Because the field moves faster than any curriculum, each topic is paired with research implications for business scholars; the discussion is link-heavy, and the original news, industry, and company sources are preserved throughout via inline hyperlinks and footnotes so that readers can follow the trail and form their own views.

## 9.1 The AI Landscape in Early 2026: A Bird's-Eye View

### 9.1.1 The Technology Tree Is Growing Fast

This course (DOTE 6635) has been offered every spring since 2024, and its evolution tells the story of AI itself. In its first year, the focus was on natural language processing and computer vision. In Year 2, it shifted to large language models and causal machine learning. Now, in Year 3, the centerpiece is **reinforcement learning and agentic AI**. The fact that the syllabus cannot repeat prior content for more than a fraction of each offering speaks to the velocity of the field: the technology tree is growing faster than any curriculum can track.

This pace is reminiscent of a motivation drawn from Demis Hassabis, co-founder of Google DeepMind and 2024 Nobel Laureate in Chemistry. As a 12-year-old chess prodigy, Hassabis resigned a drawn position against the ex-Danish champion out of sheer exhaustion, then immediately realized his error when his opponent pointed out the stalemate. The experience prompted a question that would shape his career: *Are we wasting our minds? Is this the best use of all this brain power?* The same question animates this course: in an era when AI can handle an increasing share of cognitive labor, where should brilliant human minds focus?

### 9.1.2 2025 in Review: Nature's Ten and Karpathy's Reflections

**Liang Wenfeng and Nature's Ten.** At the close of 2025, *Nature* named **Liang Wenfeng**, the founder of DeepSeek, to its annual list of ten people who shaped science. *Nature* called him a "tech disruptor" whose open-source models demonstrated that the U.S. was not as far ahead in AI as many experts had assumed (Nature, Dec 2025; SCMP, Dec 2025). For context, the 2023 list featured ChatGPT and Ilya Sutskever. The shift from an American product to a Chinese entrepreneur signals a meaningful rebalancing of the global AI landscape.

A related milestone: **DeepSeek-Math-V2** became the first open-source model to achieve IMO gold medal-level performance, solving 5 of 6 problems at the 2025 International Mathematical Olympiad. The key engineering insight was **self-verification**, the model checks its own proofs for rigor and completeness, a dual-model architecture that mimics how human mathematicians work (SCMP, 2025; HuggingFace).

**Karpathy's Year-in-Review.** Andrej Karpathy's widely circulated blog post[75] provides a useful snapshot of where the field stood at the turn of 2026. Several themes stand out:

[75] https://karpathy.bearblog.dev/year-in-review-2025/

- **RLVR (Reinforcement Learning with Verifiable Rewards)**[76] emerged as the most consequential technical development of 2025, fundamentally altering the LLM training stack. Much of the capability progress came from longer RL runs rather than larger pretraining datasets.
- **The shape of intelligence differs.** Karpathy offered an evocative analogy: human intelligence is like an irregularly shaped star, strong in some directions, weak in others. AI intelligence has a *different* shape. The overlap is partial, which is precisely why human-AI coordination (and **context engineering**[77]) matters so much.
- **Cursor and "vibe coding."** Tools like Cursor dramatically shortened the distance between human intent and compute. The term "vibe coding", writing software by describing what you want in natural language, entered the mainstream vocabulary.
- **Summary.** Karpathy concluded that 2025 was "an exciting and mildly surprising year" where LLMs proved simultaneously smarter and dumber than expected.

### 9.1.3 The Four Maintracks of AI Progress

The Spring 2026 AI landscape can be organized around four maintracks, all of which share a common thread: **facilitating humans to leverage compute**. Human brains are good at many things, but raw computation is not one of them. GPUs are. The state-of-the-art AI stack is fundamentally about bridging that gap.

These four tracks are listed in *increasing order of importance but decreasing order of maturity*, as summarized in Table 9.1.1.

To make these concrete:

- **AI Coding** is the most mature. Previously, building software from scratch required a team of expert engineers. Now, with vibe coding, a single person can produce a working prototype in hours, 10x to 100x faster.

[76] In the reinforcement-learning vocabulary of Chapters 5–7, RLVR is policy optimization in which the reward is supplied by an automatic, programmatic checker rather than by a human or a learned reward model: for a mathematics problem the reward is 1 if the model's final answer matches the known solution and 0 otherwise; for code it is whether the generated program passes a suite of unit tests. Because the reward is *verifiable* it is cheap to evaluate at scale and essentially impossible to "game," which is exactly what a learned reward model is vulnerable to. This contrasts with RLHF (Ouyang et al., 2022), where the reward signal is itself a model fitted to human preference comparisons.

[77] *Context engineering* is the practice of deciding what information (instructions, worked examples, retrieved documents, intermediate results) to place into a model's finite input window so that it has exactly what it needs and no more. It is conceptually close to choosing the right conditioning set, or a sufficient statistic, for a prediction problem — only here the object being curated is the prompt.

**Table 9.1.1.** The four maintracks of AI progress in Spring 2026.

| Maintrack | Core Idea | Business Research Implication |
|---|---|---|
| AI Coding | Connecting human intent with compute through natural language | Productivity, innovation, software economics |
| Deep Research | Automating information acquisition and processing | Knowledge work transformation, research methodology |
| World Models | Data-driven simulation of the physical world | Operations, supply chain, digital twins |
| AI Scientist | Automated hypothesis generation and validation | Research design, scientific discovery, R&D strategy |

- **Deep Research** is the automation of information acquisition and processing. If you use ChatGPT's deep research function to do a literature review today, it will, with high probability, produce a more comprehensive survey than most individual researchers could. The limitation: AI still lacks the *judgment* to evaluate what matters.
- **World Models** use data-driven simulation to replicate physical environments, providing feedback loops for AI systems in robotics, autonomous vehicles, and operations.
- **AI Scientist** represents the frontier: automated hypothesis generation, validation, and iteration. When this maintrack matures, it will fundamentally reshape, and potentially displace, the work of academic researchers.

These four tracks reflect a paradigm shift: AI is no longer just a tool for prediction (the "ML-for-X" era), but an **active agent** that can plan, execute, and iterate on complex tasks.

### 9.1.4 The AI Industry by the Numbers

A few data points capture the scale of what is happening:

- **The capital cycle:** Nvidia committed to investing over $100 billion in the coming years. OpenAI uses the money to purchase Oracle's cloud services. Oracle then purchases Nvidia's chips. The money circulates, but the infrastructure grows relentlessly.

- **Talent wars:** Meta's Mark Zuckerberg aggressively recruited top AI talent, reportedly offering Yu Jiahui (from OpenAI) an annual package of ~$100 million. By contrast, an economist position at OpenAI, still generous by academic standards, pays orders of magnitude less. The market is pricing foundational AI skills at a premium that dwarfs adjacent fields.
- **AI-generated content:** By some estimates, roughly half of new internet content on text-heavy platforms is now AI-generated. For video platforms like YouTube and Bilibili, the share is lower but rising fast.
- **Meta acquired Manus**, a Chinese-founded agentic AI startup based in Singapore, for approximately $2 billion in late December 2025 (CNBC, Dec 2025). Notably, none of the Manus founders were building foundational AI, they were "wrappers" who figured out the right market application. The lesson: in the age of abundant AI capabilities, **market insight and product sense** can be as valuable as technical breakthroughs.

### 9.1.5 Key Model Launches

The first quarter of 2026 witnessed several landmark releases, summarized in Table 9.1.2 and discussed in detail below.

**Table 9.1.2.** Key model and system launches in early 2026.

| Launch | When | Significance |
|---|---|---|
| Codex-5.3 & Claude Opus-4.6 | Feb 2026 | Launched within 10 minutes of each other; ushered in "agentic engineering"; models use their own code to upgrade themselves. |
| DeepSeek V4 | Anticipated (unreleased as of Mar 2026) | Reported to match or exceed Claude's coding; improved context understanding and code reasoning; claims unverified. |
| Google Aletheia | 2026 | Autonomously verifies mathematical proofs; tackled First Proof problems. |
| Claude Mythos Preview | Apr 7, 2026 | Announced then withheld; 93.9% on SWE-bench Verified, 97.6% on USAMO 2026; discovered thousands of zero-day vulnerabilities. |

- **Codex-5.3 and Claude Opus-4.6** launched within 10 minutes of each other in February 2026, marking a new era of **agentic engineering**, models that can not only generate code but autonomously manage multi-step software development workflows.[78] The instructor noted that the upgrade from their predecessors

[78] https://x.com/karpathy/status/2019137879310836075

(Codex-2.5.2 and Claude Opus-4.5) was "still very noticeable, even from a user's experience", and critically, these models **use their own code to upgrade themselves**, creating a flywheel of self-improvement that never stops. Karpathy coined the term "agentic engineering" for this paradigm, just as he had coined "vibe coding" the previous year. The practical implication: the tools are evolving faster than any course can track, and nobody yet knows how to leverage them to their full potential.

- **DeepSeek V4** generated significant anticipation around Chinese New Year, exactly one year after DeepSeek R1 shocked the world. Multiple sources, including subscription-only outlets like *The Information*, cited insiders claiming V4 could match or exceed Claude's coding performance. Reported features included dramatically improved context understanding for complex coding prompts, sustained scaling-law gains (no performance degradation with more data and compute)[79], and stronger code reasoning capabilities (Yahoo Tech, Jan 2026). *Note: As of March 2026, DeepSeek V4 has not officially launched despite multiple predicted release windows. The claims remain unverified by independent benchmarks.*

- **Google Aletheia**, a system designed to autonomously verify mathematical proofs, tackled problems from the First Proof initiative.[80]

- **Claude Mythos Preview** was announced by Anthropic on April 7, 2026, and immediately withheld from public release, a first for any major AI lab (Anthropic, Apr 2026; CNBC, Apr 2026; NYT, Apr 2026). The model achieved **93.9% on SWE-bench Verified** and **97.6% on USAMO 2026**[81], each a double-digit lead over Opus 4.6. Most strikingly, Mythos autonomously discovered **thousands of zero-day vulnerabilities**[82] in every major operating system and web browser, including a 17-year-old remote code execution flaw in FreeBSD and a 27-year-old vulnerability in OpenBSD. Rather than releasing it publicly, Anthropic launched **Project Glasswing**, a consortium with Microsoft, Amazon, Apple, CrowdStrike,

[79] *Scaling laws* (Kaplan et al., 2020) are empirical power-law relationships showing that a model's pretraining loss falls predictably and smoothly as the number of parameters, the amount of data, and the compute budget grow. "Sustained gains" means the curve has not yet bent over, so additional resources still buy measurable improvements rather than hitting a plateau.

[80] https://arxiv.org/pdf/2602.21201

[81] *SWE-bench Verified* is a benchmark of real-world software-engineering tasks: each item is an actual GitHub issue, and a model "passes" only if its code change makes the repository's existing test suite go green. *USAMO* is the USA Mathematical Olympiad, a proof-based competition in which problems are graded on the written argument, not merely on a final numerical answer — so high scores require coherent multi-step reasoning.

[82] A *zero-day* vulnerability is a security flaw still unknown to the software's vendor — so the vendor has had "zero days" to issue a fix — which makes it especially dangerous if an attacker finds it first. A *remote code execution* flaw (mentioned next) is a particularly severe class that lets an attacker run arbitrary code on a victim's machine over the network.

Palo Alto Networks, and ˜40 other companies, to use Mythos exclusively for **defensive cybersecurity**. On April 8, one day after the announcement, Treasury Secretary **Scott Bessent** and Federal Reserve Chair **Jerome Powell** convened an emergency meeting with the CEOs of Goldman Sachs, Bank of America, Citigroup, Morgan Stanley, and Wells Fargo to warn about cyber risks posed by Mythos.[83] As the instructor observed, the dual reaction captured the fundamental tension of 2026: "On one hand, excitement about this new AI model. On the other hand, how should we co-evolve with AI? Is our capability of managing and controlling AI growing as fast as the capability of AI itself?" His answer was blunt: "Definitely no, but we should think very hard about this, because if AI crosses some boundaries, the only way to stop it is to physically cut the internet, and by then it's probably already too late."

**For business researchers:** These model launches are not mere engineering milestones. Each represents a shift in the cost structure of cognitive labor, with direct implications for organizational design, market structure, and competitive strategy.

### 9.1.6 The 2026 Stanford AI Index Report

The **2026 Stanford AI Index Report**, published by the Stanford Institute for Human-Centered Artificial Intelligence (HAI), provided the most comprehensive empirical snapshot of the AI landscape (Stanford HAI, 2026; Full Report PDF). The instructor highlighted several key findings in the final lecture, collected in Table 9.1.3.

- **Uneven capability.** AI has achieved superhuman performance in some domains, winning gold at the International Mathematical Olympiad, but **robots still fail most household tasks**. AI lacks a sense of time (because, as the instructor quipped, "it is eternal and will not die"), which has practical implications for managing AI agents.
- **Entry-level employment is declining** in AI-exposed occupations, consistent with the evidence discussed in Section 9.5.
- **Environmental footprint is expanding.** The carbon emissions of training and running frontier models are growing rapidly, a research opportunity for operations and sustainability scholars.
- **AI in science.** AI models now outperform human scientists in certain domains, reinforcing the trends documented in Sections 9.3 and 9.4.

[83] https://www.cnbc.com/2026/04/10/powell-bessent-us-bank-ceos-anthropic-mythos-ai-cyber.html

**Table 9.1.3.** Selected findings from the 2026 Stanford AI Index Report.

| Theme | Finding |
|---|---|
| Uneven capability | Superhuman in some domains (gold at the IMO), yet robots still fail most household tasks; AI lacks a sense of time. |
| Entry-level employment | Declining in AI-exposed occupations (consistent with Section 9.5). |
| Environmental footprint | Carbon emissions of training and running frontier models growing rapidly. |
| AI in science | AI models now outperform human scientists in certain domains. |
| Clinical care | Despite promise, evidence of transformative healthcare impact remains limited. |
| Formal education | Lagging far behind; institutions are not adapting quickly enough. |
| Expert-public divergence | Experts and the general public hold completely different perspectives on AI's trajectory and risks. |

- **Clinical care.** Despite promise, the evidence for AI's transformative impact in healthcare remains limited.
- **Formal education is lagging far behind.** The instructor's assessment was emphatic: educational institutions are not adapting quickly enough to the AI transformation.
- **Expert-public divergence.** Perhaps most alarmingly, experts and the general public hold **completely different perspectives** on AI's trajectory and risks. The instructor warned that this gap could widen inequality and deepen societal polarization: "AI's power has been demonstrated very solidly. But there are a lot of people who still don't believe it, while others are embracing it in a way that will substantially widen not only the productivity gap but also mindset differences, which I believe will tear human societies apart even more."

The instructor used the Stanford AI Index to frame the course's own evolution: in 2024, the "What's New in AI" section comprised roughly 5 slides; in 2025, it grew to 30 slides; in 2026, it reached **91 slides**, approximately 7 new slides per week, each requiring about an hour of lecture time. "AI will transform, if not disrupt, higher education very soon. Higher education is a business of knowledge, and AI is best at knowledge. We definitely face competition from AI."

## 9.2 AI Coding and the Zero Marginal Cost of Code

### 9.2.1 The AI-Native Cost Structure

One of the most consequential ideas in early 2026 is the notion of **zero marginal cost of code**. When coding agents can autonomously generate, test, and deploy software, the economics of software production fundamentally change. This is the "AI-native mindset", treating code not as a scarce, expensive artifact but as an abundant, near-zero-cost commodity (Yage.ai, 2026; The Modern Software Dev, 2026).

A vivid illustration: in January 2026, **Michael Truell**, CEO of Cursor, orchestrated hundreds of AI agents to autonomously build a web browser ("FastRender") from scratch, 3 million+ lines of Rust code including an HTML parser, CSS cascade engine, layout system, and custom JavaScript VM. It ran uninterrupted for **one week** with no human instructions. The result was far from production-ready (critics called it "shoddy code at scale"), but the signal was unmistakable: a project that would have taken a team of engineers several months was executed in a single autonomous run (Truell, Jan 2026; Fortune, Jan 2026).

The practical implication, as emphasized repeatedly in the lecture: code is now **disposable**. If a piece of code can only be used once, that is fine, compute and storage are cheap relative to the value of a researcher's time. As an OpenAI infrastructure engineer put it in a widely circulated podcast: "It is much easier to teach an engineer how to do research than to teach a scientist how to do engineering. The most critical thing is iteration speed." The corollary for PhD students: let your AI agents work around the clock. If they are idle, you are wasting your most precious resource, time.

**Implications for business research.**

- **Platform economics:** If the cost of building software approaches zero, barriers to entry collapse. What sustains competitive advantage when anyone can spin up a product?
- **Innovation strategy:** The bottleneck shifts from "can we build it?" to "should we build it?", from engineering capacity to taste, judgment, and market understanding.
- **Operations management:** Customized software solutions for niche operational problems become feasible, enabling mass customization of business processes.

### 9.2.2 Quality of AI-Generated Code

However, zero marginal cost does not imply zero marginal risk. A rigorous audit by Graham Straus and Andrew Hall examined how accurately Claude Code replicated and extended Hall's published PNAS paper on vote-by-mail.[84] The audit found both impressive successes and instructive failure modes.

**What went right.**

- Claude replicated the original paper's estimates exactly and coded 29 of 30 California counties correctly on treatment timing.
- The collected election data correlated above **0.999** with manually collected ground-truth data.
- Overall, the AI did a "remarkably good job", the estimates were similar in magnitude to human-produced results.

**What went wrong.**

- The main mistake was a **failure to collect all needed data**, specifically, senatorial and gubernatorial election data for two states. This is a judgment error (knowing *what* data to gather), not a coding error.
- One county's treatment year was miscoded, and non-presidential elections were not used to compute turnout, subtle errors of the kind a human RA might also make on a first pass.
- AI tends to produce **unsolicited extensions**, additional analyses and robustness checks that were not requested and lack clear academic value. As the instructor noted, AI is eager to "do more" but lacks the judgment to know what additions are scientifically meaningful.

**The exponential trajectory.** The instructor emphasized that AI coding capability is improving at an exponential rate: what was 1x in 2024 became roughly 10x in 2025 and is on track for 100x in 2026. The practical advice: don't compete with AI on execution, focus on judgment, taste, and knowing *what* to ask. Document your AI-assisted workflow thoroughly, because transparency about the process is more valuable than hiding it.

[84] https://www.andrewbenjaminhall.com/Straus_Hall_Claude_Audit.pdf

**Takeaway:** The productivity gains from AI coding are real, but they demand a new form of literacy: the ability to **audit and validate** machine-generated code. For empirical researchers, this is not optional, it is a matter of scientific integrity. AI excels at execution but still struggles with the judgment calls that define good research: what data to collect, which specifications matter, and when to stop extending.

## 9.3 AI-Driven Scientific Replication and Reproducibility

### 9.3.1 Complete Replication of a PNAS Paper

In one of the most striking demonstrations of early 2026, Professor Andrew Hall of Stanford Graduate School of Business used Claude Code to achieve a **complete replication** of his previously published PNAS paper, Thompson, Wu, Yoder, and Hall (2020), "Universal Vote-by-Mail Has No Impact on Partisan Turnout or Vote Share."[85] The original study used a staggered difference-in-differences design to examine how universal vote-by-mail in Washington, Utah, and California (the only three U.S. states that adopted it) affected partisan electoral outcomes. The key findings: vote-by-mail does not affect either party's share of turnout or vote share, but modestly increases overall average turnout rates. The entire AI-driven replication process, including all prompts and conversations, was open-sourced.[86]

**How it was done.** Hall used **Claude Opus 4.5** via the command-line interface (functionally equivalent to VS Code or Cursor integrations). Unlike the single-figure demo in Section 9.3.2, this was a *full paper* replication: literature review, data collection, original analysis, data extension through 2024, robustness checks, and paper writing. Hall laid out a series of **checkpoints**, at each milestone, the AI summarized its progress and flagged concerns for human review before proceeding. The remarkable finding: the process was essentially **one-shot**. Most human prompts were simply "Approved, please go ahead." The AI read the original paper, understood the methodology, collected and processed data, ran the analysis, and produced a complete replication with extension, with minimal human correction along the way. Hall subsequently published a detailed assessment of the replication's accuracy.[87]

[85] https://www.pnas.org/doi/10.1073/pnas.2007249117
[86] https://github.com/andybhall/vbm-replication-extension
[87] http://www.andrewbenjaminhall.com/

**Cost.** The basic Claude Code subscription is approximately $28/month. The entire replication of a PNAS paper cost less than a single month's subscription.

This raises profound questions:

- If an AI agent can replicate a paper from its instructions and data, what does that say about the **complexity** (or lack thereof) of much empirical research?
- Can AI-driven replication become a **standard part of the peer-review process**?
- What happens to the value of "execution skill" when execution is automated?

**Practical advice from the instructor:** Open-source your AI-driven replication, but go beyond "I asked AI to do it and it finished." Share the insights, what worked, what failed, what surprised you. The learning is in the process, not just the output.

### 9.3.2 Live Demo: Replicating a Published Figure in 10 Minutes

In the opening lecture, a live demonstration illustrated the paradigm shift. The task: replicate a key figure from a published methodology paper on Double Machine Learning (DML),[88] which shows how cross-validation error decreases across training epochs and how the proposed method outperforms a baseline. Normally, this is the kind of exercise given to a new PhD student or research assistant, read the paper, understand the methodology, write synthetic data generation code, train deep neural networks, and produce the figure. It typically takes days.

The procedure:

1. Write a short instruction file (the same instructions one would give a human RA).
2. Provide the paper PDF and the original figure for reference.
3. Run a single prompt in a coding agent (Codex / Claude Code).

Within **10 minutes**, the agent: read and summarized the paper's methodology, generated synthetic data, implemented the DML training pipeline, computed the benchmarks, and produced a figure that closely replicated the original, not perfectly, but to a standard that would pass muster for an RA's first attempt. The entire process, including prompts, is available on GitHub for students to reproduce.

[88] *Double* (or debiased) *machine learning* is a framework that should be familiar from modern econometrics: flexible machine-learning estimators absorb high-dimensional nuisance components (for example the outcome regression and the propensity score), while inference on the low-dimensional causal parameter of interest remains valid because the estimating equation is made *Neyman-orthogonal* (first-order insensitive to small errors in the nuisances) and the nuisances are fit on held-out folds via *cross-fitting* (sample splitting). See V. Chernozhukov et al., "Double/Debiased Machine Learning for Treatment and Structural Parameters," *The Econometrics Journal* 21(1), 2018.

**Implication:** The exercise that used to be a multi-day proving ground for new researchers can now be completed in minutes. This does not make the researcher obsolete, it shifts the bottleneck from *execution* to *judgment*: knowing what to replicate, why it matters, and whether the output is correct.

### 9.3.3 Scaling Reproducibility

A broader initiative to **scale reproducibility** was introduced by **Yiqing Xu** (Stanford Political Science) and **Leo Y. Yang** (Hong Kong Baptist University) in their paper "Scaling Reproducibility: An AI-Assisted Workflow for Large-Scale Reanalysis" (Xu & Yang, 2026; Video Demo). The system uses a **Claude Code agentic architecture** to automatically reproduce results from published empirical papers.

**Scope and results.** The system was tested on **92 instrumental variable (IV) studies** drawn from three top political science journals, the *American Political Science Review* (APSR), the *American Journal of Political Science* (AJPS), and *The Journal of Politics* (JOP), spanning 2010-2025. The headline result: **87% overall end-to-end success rate** (55 of 67 benchmark papers plus all 25 newly published papers). When authors provided both code and data, reproducibility was **100%** at both the paper and specification levels, across 215 total specifications.

**The three-layer architecture.** As the instructor explained, the system employs a three-level pipeline:

1. **Orchestration layer.** An LLM orchestrator coordinates the overall workflow, receiving human inputs (the research paper, basic code, identification strategy, data documentation, README files) and directing specialized sub-agents.

2. **Skills layer.** Structured instructions, essentially standard operating procedures (SOPs), that tell each agent what to do and what not to do in specific situations. As the instructor put it: "Think of it as the instructions I give my students, what you should do in this situation, what you should not do in that situation."

3. **Sub-agent layer.** Specialized agents handle discrete tasks: a *profiler* for multi-language code parsing and preparation, a *metadata extractor*, a *librarian* for retrieving replication packages from R, Python, or other sources, and a *journalist* for template-based reporting of results.

The architecture is sketched in Figure 9.3.1.

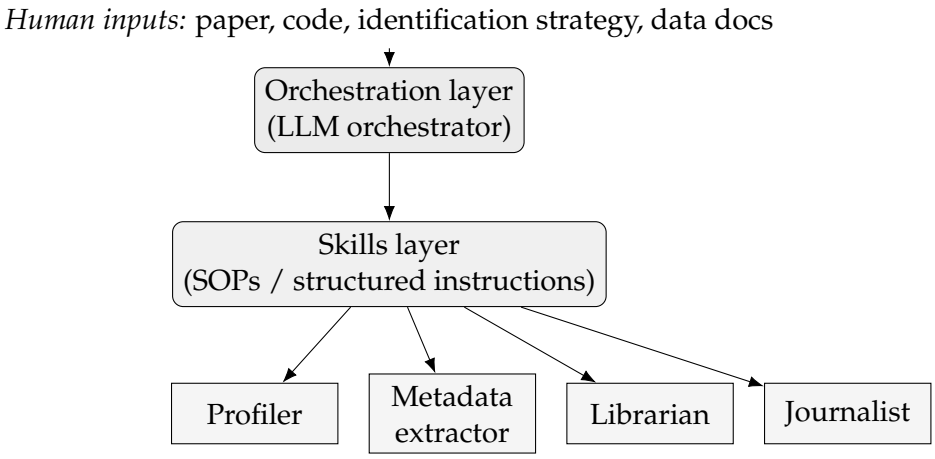


**Figure 9.3.1.** The three-layer agentic reproducibility pipeline of Xu and Yang (2026): an LLM orchestrator directs a sub-agent layer through a skills layer of structured SOPs.

**Reproducibility vs. replicability.** The instructor drew a careful distinction: *reproducibility* concerns whether the same code and data produce the same numerical results, while *replicability* concerns whether the underlying knowledge and findings hold when tested on new data or in new settings. As the instructor noted, Xu is a "very rigorous researcher" who uses the term "reproducibility" precisely. The course emphasizes *replicability* in student projects, asking them to go beyond the original dataset.

**The SOP automation principle.** The instructor used this paper to reinforce a broader lesson: "As long as something can be written as a standardized operating procedure, in one way or another as an SOP, then it can be automated by AI. Regardless of whether this is research or anything else you can think of." An agent can complete in minutes or hours what might take a human 20 days. The practical assignment: students are asked to replicate this pipeline for *Management Science* papers, building their own agentic reproducibility system.

### 9.3.4 Automating Policy Evaluation at Scale (APE)

The **Automating Policy Evaluation (APE)** project, led by **Prof. David Yanagizawa-Drott** at the **Social Catalyst Lab** (University of Zurich), pushed the frontier even further: rather than merely reproducing existing papers, APE has AI agents **write entirely new economics papers from scratch** and then pits them against published papers from the *American Economic Review* and *AEJ* in a tournament-style evaluation (Social Catalyst Lab, 2026; GitHub).

**How the tournament works.** AI agents autonomously identify policy questions, fetch real data from public APIs (Census, BLS, FRED, etc.), conduct econometric analysis (difference-in-differences, regression discontinuity, etc.), and produce full

manuscripts with figures. These AI-written papers then compete head-to-head against published human papers, evaluated by an LLM judge. Position swapping ensures that evaluation order does not bias results.[89] As the instructor discussed, as of late February 2026, the platform had generated approximately **158 AI-written papers** across **˜6,000 head-to-head comparisons**, with an AI winning rate of roughly **4.2%**.

**Key findings from the data.**

- **Human papers are still better**, but the gap is narrowing. The distribution of quality scores clearly favors human-authored papers as of February 2026, and top-journal human papers remain stably at the top.
- **AI is improving steadily.** Unlike human paper quality, which is relatively stable over time, AI paper quality shows a **consistent upward trajectory** as the system receives feedback and iterates. The instructor's assessment: "Don't take a break, this is February 2026. At the end of our course in April, I will revisit this. And probably I will post similar figures every month."
- **Quantity is not the bottleneck.** AI can produce papers at a pace limited only by compute, potentially thousands per day. Human paper production, by contrast, is bounded by the speed at which researchers can do good work.
- **Novel data discovery.** During its paper-generation pipeline, the AI system occasionally identified **new datasets** that had not previously appeared in the economics literature (e.g., the Medicaid Provider Spending TMSIS data, initiated in February 2026), curating them for future use by human researchers.

**The end-to-end pipeline.** The system's architecture involves: initialization with human guidelines → policy domain and method selection → data exploration → idea generation and feasibility checking (by AI) → data acquisition and analysis → writing and review → **self-replication** to verify that the AI's own findings are indeed reproducible. This last step, AI replicating its own results, closes the loop on the reproducibility question.

> **Research opportunity:** APE provides a unique, evolving benchmark for the question: How good is AI at doing social science research? The open-source data (papers, code, scores, and failures) is available on GitHub,

[89] An *LLM judge* is a language model prompted to score or compare outputs in place of a human rater. Such judges exhibit a well-documented *position bias*: a tendency to favor whichever candidate is presented first (or last) regardless of content. Evaluating each pair twice with the order swapped and averaging is therefore the experimental-design analogue of counterbalancing treatment order across subjects to net out an order effect.

offering a rich empirical setting for studying AI research capability, the nature of creativity in economics, and the evolving human-AI quality gap. For business researchers: How should journals, funders, and tenure committees respond to AI-enabled mass replication and paper generation? What is the appropriate standard when the cost of producing a research paper approaches zero?

**APE update (March 2026).** By late March 2026, the APE platform had grown to over **540 AI-generated papers** across more than **12,500 head-to-head comparisons**, with the AI win rate holding at approximately **4.0%**. As the instructor noted, while 4% may sound low, the trajectory matters: "What will happen in one year? Let's see." The platform was adding roughly 80 papers and 200 new ideas per week.

**FARS: Fully Automated Research System.** A parallel initiative, **FARS** (Fully Automated Research System) by **Analemma AI**, demonstrated end-to-end autonomous research at scale.[90] In its first public deployment (228 hours, late February to March 3, 2026), FARS proposed **244 research hypotheses** and produced **100 short research papers**, averaging ˜2 hours and ˜$1,000 per paper, consuming ˜11.4 billion tokens.[91] The system operates through four sequential modules: Ideation, Planning, Experiment, and Writing. As the instructor emphasized, the common thread across APE, FARS, and similar AI-for-AI-research pipelines is that **evaluation automation is the key**, the ability to programmatically assess whether an AI's output is good, bad, or improvable is what makes the entire loop viable.

### 9.3.5 Will Peer Review Be Disrupted?

The ICML 2026 peer review scandal brought the sustainability of academic peer review into sharp focus.[92] ICML, one of the premier machine learning conferences, implemented two reviewer policies:

- **Policy A** (default): No LLM use permitted at any stage of the review process.
- **Policy B** (opt-in): LLMs allowed for understanding papers, reviewing literature, and polishing review text, but **not** for drafting the review itself.

[90] https://analemma.ai/blog/introducing-fars/

[91] A *token* is the elementary unit of text a language model reads and writes — typically a short chunk of a word rather than a whole word. As a rule of thumb one token is about three-quarters of an English word, so 1,000 tokens $\approx$ 750 words. Model usage and pricing are metered per token, separately for input (prompt) and output (generation), so token counts behave like a per-unit cost of computation: the headline numbers in this section (billions of tokens, dollars per paper) are essentially compute bills.

[92] https://blog.icml.cc/2026/03/18/on-violations-of-llm-review-policies/

**Detection via watermarking.** The program committee employed an ingenious detection mechanism based on research by Rao, Kumar, Lakkaraju, and Shah.[93] Invisible watermarks, drawn from a dictionary of 170,000 phrases, with two randomly sampled per paper (collision probability less than 1 in 10 billion), were embedded in submission PDFs. These phrases are invisible to human readers but are ingested by LLMs, which then reproduce the trigger phrases in their outputs. The method achieved ˜98.6% detection accuracy with zero false positives across 10,000+ reviews tested.

**Results.** The program committee identified **795 reviews** (˜1% of all reviews) from **506 unique reviewers** who had pledged Policy A (no LLM use) but were caught using LLMs. Every flagged case was also manually inspected by a human. As a consequence, **497 papers** (˜2% of all submissions) were desk-rejected, and 51 repeat violators (those who violated in more than 50% of their reviews) were removed entirely.

**The deeper question.** As the instructor observed, the scandal points to a structural problem: "Nowadays, you can finish probably hundreds of papers very fast. And your reviewers can also review these papers very fast. So basically, AI is reviewing AI." The marginal cost of producing a research paper is approaching zero (see APE and FARS above); the marginal cost of reviewing one is similarly collapsing. If both sides of the peer review equation are automated, the entire system of trust and quality control that underpins academic publishing faces a fundamental crisis. How should conferences, journals, and the research community rebuild trust in the age of AI? The instructor characterized this as "a social experiment in the making."

### 9.3.6 Auto-Research Published in *Nature*

The field reached a milestone on March 26, 2026, when **Sakana AI**'s paper "Towards End-to-End Automation of AI Research" was published in ***Nature***, the first peer-reviewed publication in the world's most prestigious general science journal describing a system that autonomously conducts end-to-end scientific research (Lu et al., 2026; GitHub). The team, led by **Chris Lu** and **Cong Lu** with collaborators from the University of British Columbia, the Vector Institute, Oxford, and Sakana AI (including **Jeff Clune** and **David Ha**), demonstrated a pipeline that handles everything from ideation through novelty checking, experiment design, hyperparameter tuning, execution, plotting, and writing.

**The ICLR workshop experiment.** What distinguished this work, and what the instructor identified as "the most distinguishing factor of why this paper could get pub-

[93] https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0331871

lished in Nature", was a real-world test of the system's output. The team submitted **three fully AI-generated papers** to an ICLR 2025 workshop. Reviewers were informed that AI-generated papers might be in the pool (3 out of 43 submissions) but were not told which specific papers were AI-generated. One of the three received scores of 6, 7, and 6 (average 6.33), scoring higher than **55% of human-authored papers**, good enough for acceptance. The paper was proactively withdrawn before publication for ethical reasons, since AI authorship violated conference norms.

The instructor drew two key lessons:

1. **Novelty checking is the hardest problem at the front end.** "Newton's laws can be considered not novel; Einstein's $E = mc^2$ can also be considered not novel, depending on what metric we use." If the best human scientists produce both false positives and false negatives when judging novelty, automating this judgment for AI is extraordinarily difficult.

2. **Evaluation of execution is the hardest problem during the process.** "In the age of AI, execution is easy. Ideas are cheap, every single day you can come up with 10,000 novel ideas. However, only if you can execute and finish the idea in a way that is convincing to the entire community will it become a great idea." The ability to provide **automated, quantifiable feedback** on whether an experiment is on the right track, which hyperparameters to tune, which direction to pursue, remains the critical bottleneck. The Sakana team's success in passing real peer review suggests this bottleneck is beginning to yield.

### 9.3.7 AI Workflow for Statistical Package Development (StatsClaw)

**StatsClaw**, developed by **Tianzhu Qin** (University of Cambridge) and **Yiqing Xu** (Stanford), the same Xu behind the reproducibility system in Section 9.3.3, is a multi-agent workflow for AI-collaborative development of statistical software packages (Qin & Xu, 2026; statsclaw.ai; GitHub). As the instructor explained, while the authors frame it from the developer's perspective, "I have a new method; how do I package it credibly?", the more compelling use case for applied researchers runs in the opposite direction: "Here is a new statistical package; can we trust it?"

**The key innovation: information barriers.** The central problem in AI-assisted statistical software development is not code generation but **verification**. A model that misunderstands a mathematical specification will produce code that is syntactically valid, passes superficial checks, and silently computes the wrong thing. StatsClaw addresses this by enforcing strict **information barriers** between agents:

1. **Planner.** Produces independent specifications for implementation, simulation, and testing.
2. **Builder.** Implements the code *without knowing the ground-truth parameters*.
3. **Simulator.** Generates data *without knowing the algorithm*.
4. **Tester.** Validates using deterministic criteria via a **Monte Carlo evaluation** directly from the mathematical specification.

Because the agents cannot see each other's instructions, errors *decorrelate* rather than compound, a fundamental improvement over the standard generate-then-test workflow. The pipeline supports R, Python, C++, Julia, and Stata, and was demonstrated end-to-end on a probit estimation package. The instructor called it "a must-learn technique if you want to become an applied social science researcher" and noted that, had he encountered it earlier, he would have designed a homework assignment around it.

### 9.3.8 Social Science Simulation Automation (YuLan-OneSim)

A team from **Renmin University's Gaoling School of Artificial Intelligence** introduced **YuLan-OneSim**, described in the paper "LLM Agents as Social Scientists: A Human-AI Collaborative Platform for Social Science Automation" (arXiv:2604.01520; GitHub). The system can be thought of as Karpathy's autoresearch adapted for social science: users provide context and prompts specifying a research question; the system embeds a simulator supporting up to **100,000 concurrent agents**, runs the simulation autonomously via an auto-programming framework that translates natural-language scenario descriptions into executable code, and produces reports with statistics.

**The evaluation challenge.** The instructor was candid about the limitations. Unlike machine learning tasks, where metrics like loss, accuracy, or bits-per-byte[94] provide clear, quantifiable feedback, social science research "does not really have a ground truth that could be easily measured and easily quantified." The authors presented two validation examples:

1. **Network homophily.** Starting from uniform interactions across a network, the simulation reproduced the well-known sociological phenomenon where local connections strengthen while global connections weaken over time, a form of

[94] *Bits-per-byte* measures how well a language model predicts held-out text: it is the average number of bits the model needs to encode each byte of that text, i.e. a length-normalized cross-entropy (lower is better). It is the same negative-log-likelihood idea that underlies perplexity and the log-loss familiar from maximum-likelihood estimation, simply rescaled so that models with different tokenizations can be compared on a common footing.

emergent polarization. The instructor's verdict: "The granularity is too rough. We can't really say from this evidence whether the results are reliable."

2. **Brazilian housing prices.** The system attempted to simulate housing price dynamics, producing curves whose shapes resembled empirical data. Again, the instructor noted: "We have no benchmark. We don't know if it's good or bad."

The instructor's broader assessment: social science simulation is "definitely going to be something very important in the near future," and many researchers (including himself) are working to develop credible frameworks. But the fundamental challenge remains: without clear evaluation metrics, the kind of automated, quantifiable feedback that makes ML-for-ML systems like autoresearch viable, the credibility of simulation-based social science research is difficult to establish. The lesson reinforces a recurring theme: **evaluation automation is the bottleneck** (see Sections 9.3.4 and 9.3.6).

## 9.4 AI for Mathematics: From Putnam to Unsolved Problems

### 9.4.1 AxiomProver and the Putnam Competition

AI's penetration into pure mathematics accelerated sharply. **AxiomProver**, an autonomous multi-agent ensemble theorem prover for Lean 4,[95] solved 8 of 12 problems from the 2025 Putnam Competition during the competition window, and eventually solved all 12 in subsequent days. AxiomProver was built by **Axiom Math**, co-founded by **Carina Letong Hong**, who dropped out of Stanford's mathematics PhD program to pursue the venture.[96]

The Putnam is the premier undergraduate mathematics competition in North America, with approximately 4,335 student participants in 2025 from 488 institutions. To appreciate the difficulty: roughly **30% of all participants scored zero**, and the median score was just 2 out of 120. The top human score was 110/120.[97] AxiomProver's 8 problems solved in competition time would correspond to approximately 80 points,

[95] *Lean* (here version 4) is an interactive proof assistant: a proof written in Lean is a formal object that the software mechanically checks step by step, so a "Lean-verified" proof is guaranteed correct up to one's trust in the checker itself — there is no room for a hand-waved gap. This is what lets an AI's proof be accepted with confidence even when no human has reviewed every line.

[96] https://techstartups.com/2025/06/04/stanford-math-phd-students-ai-startup-axiom-raises-50m-at-300-million-valuation/

[97] https://maa.org/news/results-of-the-86th-william-lowell-putnam-mathematical-competition/

enough to rank among the **top four** and earn the title of *Putnam Fellow*, a distinction that places one's name in a permanent hall of fame of mathematical excellence.[98]

### 9.4.2 Terry Tao and AI for Erdos Problems

Fields Medalist Terence Tao publicly discussed AI contributions to longstanding Erdos problems, a benchmark for mathematical creativity and depth (Tao, 2026; Erdos Problems Wiki). In January 2026, **Erdos Problem #728** was solved largely autonomously by **GPT-5.2 Pro** combined with **Aristotle** (by Harmonic), operated by researcher Kevin Barreto. The AI produced a proof for a tightened version of the problem, and Aristotle translated it into Lean (a formal verification language) to confirm correctness. Three Erdos problems (#397, #728, #729) fell within seven days. Tao verified the proofs and cautioned that only about 1-2% of currently open Erdos problems are simple enough for today's AI to solve with minimal human help.[99]

What makes this notable is not just the result but the *process*. As the instructor observed, the AI produced **clean, readable manuscripts**, a sharp contrast to the typical experience where mathematical proofs are dense and opaque. For business school theorists, the implication is striking: in formal modeling, *defining what to prove is far more important than writing down the proof itself*. If AI can handle the algebraic heavy lifting and produce readable derivations, the bottleneck shifts entirely to formulation, choosing the right model, the right assumptions, and the right question.

### 9.4.3 First Proof and Google Aletheia

The **First Proof** initiative[100] curated a set of unseen mathematical problems specifically to benchmark AI reasoning. The project was co-authored by 11 leading mathematicians, including **Martin Hairer** (Fields Medalist, EPFL and Imperial College London), **Daniel Spielman** (Yale), **Nikhil Srivastava** (UC Berkeley), **Rachel Ward** (UT Austin), **Lauren Williams** (Harvard), and **Tamara G. Kolda** (MathSci.ai, who hosts the repository), each contributing a problem from their own unpublished research.[101]

**What makes First Proof unique.** The problems were intentionally designed to have **simple, clean statements** but require deep mathematical reasoning, testing genuine problem-solving ability rather than pattern matching or memorization. As the instructor noted, "the problem statements are very simple, because they intentionally make them simple, to test the real ability of these AIs." Crucially, the mathematicians

[98] https://github.com/AxiomMath/putnam2025
[99] https://arxiv.org/html/2601.07421v1
[100] https://1stproof.org/
[101] https://arxiv.org/abs/2602.05192

already knew the solutions, and AI models were tested both with and without internet access to prevent them from finding related work online.

**AI models tested.** The top publicly available reasoning models were evaluated: **Gemini 3 Pro**, **Gemini 3.0 Deep Think** (Google's specialized reasoning mode), and **GPT-5.2 Pro** (OpenAI). These represent the frontier of mathematical reasoning capability as of early 2026.

**Results.** In single-shot attempts assessed by the mathematician co-authors themselves, **only 2 of 10 AI-generated proofs were judged correct** (problems 9 and 10). As the instructor discussed, expert comments by Hairer, Spielman, and others revealed characteristic failure modes: some AI solutions "simply quote the node, claiming that it contains a detailed proof of the result" without providing one; others offered "an unpublished note with a very rough sketch"; and some contained outright **wrong statements**. Google DeepMind's **Aletheia** system, powered by Gemini 3 Deep Think, subsequently claimed solutions to 6 of the 10 problems (problems 2, 5, 7, 8, 9, and 10) through its agentic loop of Generator, Verifier, and Reviser, a system that checks and revises its own work.[102] OpenAI independently submitted solutions claiming 5-6 correct answers.[103]

**The key insight: autonomous verification.** The instructor returned to Aletheia's architecture in a later lecture to underscore a broader principle: the **key to any successful agentic system is autonomous verification**. Aletheia's loop, Generate $\rightarrow$ Verify $\rightarrow$ Revise $\rightarrow$ Re-verify, mirrors how human researchers work: have a problem, try to solve it, get stuck, rethink, revise, and verify again. Once a system possesses the ability to autonomously judge whether its own outputs are correct, the entire loop can run indefinitely, self-evolving, self-verifying, and self-improving. As the instructor put it: "In my opinion, the key to the success of an agentic system is the ability to have autonomous verification. Once we have it, the whole process could really self-evolve and self-improve, and it could even solve state-of-the-art mathematical problems that are puzzling the best mathematicians of our generation."

**Practical advice for theorists.** The instructor emphasized that these frontier reasoning models (at ~$200/month for subscription tiers) are **worth the investment** for researchers doing theoretical work. However, the First Proof results underscore that AI-generated proofs must be carefully verified, the failure modes are subtle and can be difficult to detect without deep mathematical expertise. As the instructor quipped,

[102] https://arxiv.org/pdf/2602.21201
[103] https://openai.com/index/first-proof-submissions/

connecting the name to its baking origin: "first proof" in baking refers to the first rising of the dough, "something to get started", and similarly, AI proofs should be treated as starting points that require human refinement, not finished products.

### 9.4.4 LLM Agents for Stylized Modeling

For business researchers accustomed to analytical modeling, a particularly intriguing development was the demonstration that LLM agents can engage in **stylized economic modeling**, formulating assumptions, deriving equilibria, and exploring comparative statics.[104] The paper introduces **PrimeNash**, a three-module system for automated game-theoretic analysis:

1. **Strategy Generation:** The system reads approximately 10,000 papers from the game theory literature to identify relevant modeling frameworks and solution concepts.

2. **Payoff Evaluation:** Given a game structure, it constructs and evaluates payoff functions.

3. **Theoretical Proof of Equilibrium:** It derives symbolic closed-form solutions and proves equilibrium existence.

The system reportedly solved **70% of dynamic game cases** with symbolic closed-form solutions, an impressive rate for automated theorem proving in economics. The key limitation: PrimeNash only works for games that admit symbolic (closed-form) solutions. Games requiring numerical methods or that lack clean analytical structure remain beyond its reach.

> **For theory-oriented researchers:** This does not mean AI replaces the theorist. Rather, it suggests AI may serve as a powerful **co-pilot** for model development, exploring the parameter space, checking algebra, and even suggesting alternative modeling assumptions. The bottleneck, as with the Erdos problems, is *formulation*: choosing the right game, the right payoff structure, and the right solution concept. Once that is done, AI can increasingly handle the derivation.

[104] https://www.cell.com/nexus/pdfExtended/S2950-1601(25)00054-3

## 9.5 AI and the Labor Market: Displacement, Augmentation, and the Intelligence Crisis

### 9.5.1 AI Starts to Replace Human Workers

By early 2026, the evidence of AI-driven labor displacement had moved from anecdotal to systematic:

- **Klarna** initially claimed its AI chatbot could do the work of 700 customer service representatives, handling 2.3 million chats in its first month. However, by May 2025, CEO Sebastian Siemiatkowski admitted the AI-first approach had gone too far and announced a pivot back to hiring human agents in a hybrid model.[105] The Klarna case is instructive: it illustrates both the promise and the limits of full AI replacement in customer-facing roles.
- **Duolingo** laid off ˜10% of its contract workforce (translators) in late 2023, with a broader "AI-first" announcement in May 2025 that the company would replace contract workers with AI. The company launched 148 AI-written courses in under a year.[106]
- **Cisco, UPS**, and many others followed suit.

Academic research confirmed these trends with increasing rigor. Two studies stand out.

**Harvard Business School data** showed three trend lines for job openings by seniority since the launch of ChatGPT (late 2022): the senior-positions line continued to rise; the average line was flat; and the junior/entry-level line declined below the baseline. The message is stark: if you are Jeff Dean, you are fine. If you are a junior developer, ChatGPT may already be a credible substitute.

**Brynjolfsson, Chandar, and Chen (2025)** , in their paper "Canaries in the Coal Mine?", provided the most rigorous evidence to date using high-frequency ADP payroll data. Their headline finding: a **13% relative decline in employment** for early-career workers (ages 22-25) in the most AI-exposed occupations, particularly software developers, with declines concentrated in roles where AI automates (rather than augments) human labor. Critically, in occupations where AI *augments* rather than replaces, youth employment actually showed a positive trajectory (Brynjolfsson et al., 2025; see also SSRN).

[105] https://www.bloomberg.com/news/articles/2025-05-08/klarna-turns-from-ai-to-real-person-customer-service

[106] https://techcrunch.com/2025/05/04/is-duolingo-the-face-of-an-ai-jobs-crisis/

Many senior professors who lived through the internet bubble of 2000 have remarked that the current AI wave feels similar. But there is reason to believe **this time is different**: unlike the dotcom era, the underlying technology is delivering real, measurable productivity gains *today*, not just promises of future value.

### 9.5.2 The Intelligence Crisis

A provocative framing emerged: the **Human Intelligence Displacement Spiral**.[107] The argument runs as follows: AI capabilities improve continuously → companies adopt AI and cut white-collar workers → displaced workers reduce spending → this creates margin pressure for companies → companies invest in even more AI to remain competitive → further displacement. The result is a vicious cycle in which each round of capability improvement triggers another round of layoffs.

**Block as a case study.** On February 26, 2026, **Block** (Jack Dorsey's fintech company, formerly Square) announced the elimination of approximately **4,000 employees**, reducing its workforce from over 10,000 to under 6,000, a cut of roughly **40%** (CNBC, Feb 2026; CNN, Feb 2026). What made this case instructive was the juxtaposition: Block simultaneously reported Q4 2025 gross profit up **24% year-over-year** ($2.87 billion) and raised its 2026 guidance to 18% growth. The stock surged approximately **24% in after-hours trading**. Dorsey's reasoning was explicit: AI "intelligence tools" enable smaller teams to do more, their capabilities are "compounding faster every single week," and most companies would reach the same conclusion within a year. As the instructor emphasized, "This is not about the company doing poorly, it's about projecting that AI will be more capable, so they don't need the people." *Bloomberg* raised suspicions of "AI-washing", using AI as a justification for cost cuts that Wall Street rewards.[108]

Compounding this, a new wave of "rent-a-human" services appeared, platforms where AI agents **hire humans** for tasks the AI cannot perform, inverting the traditional employer-employee relationship.[109]

### 9.5.3 Anthropic's Labor Market Impact Research

Anthropic published a detailed analysis of AI's labor market impacts by Massenkoff and McCrory,[110] introducing a novel metric called **"observed exposure"**, shifting fo-

[107] https://www.citriniresearch.com/p/2028gic
[108] https://www.bloomberg.com/news/articles/2026-03-01/jack-dorsey-s-4-000-job-cuts-at-block-arouse-suspicions-of-ai-washing
[109] https://www.wired.com/story/ai-agent-rentahuman-bots-hire-humans/
[110] https://www.anthropic.com/research/labor-market-impacts

cus from what AI *could* do to what it is *actually* doing in professional settings. Key findings include:

- Computer programmers have the highest observed exposure (75% task coverage), followed by customer service reps and data entry keyers (67% coverage).
- No statistically significant impact on unemployment rates *yet*, but tentative evidence of slowed hiring for workers aged 22-25.
- Women are significantly overrepresented in AI-exposed occupations; exposed workers tend to be more educated and higher-paid.
- Occupations with higher observed exposure are projected to grow more slowly through 2034 (per BLS data).

The instructor noted that Anthropic hired PhD economists at competitive salaries (the posted range was approximately $265K-$315K per year) specifically for this kind of rigorous empirical work, a signal of how seriously frontier AI labs take labor market research.

**The fresh-graduate effect.** One of the most striking findings is about **young workers aged 22-25**. Using a difference-in-differences/event study design comparing top-quartile AI-exposed occupations against zero-exposure occupations, Massenkoff and McCrory found that the **job-finding rate** for fresh graduates in high-exposure occupations dropped by approximately **14%** relative to 2022 levels after ChatGPT's release. The effect is specific to entry-level workers, no comparable decrease was observed for workers older than 25. As the instructor emphasized, this is "first-hand evidence" of AI's impact on those entering the most exposed job market segments. However, two caveats are important: the result is just barely statistically significant, and it measures the *rate of entry into occupations* (job-finding), not layoffs or unemployment per se.

**Occupational exposure and practical implications.** At the occupation level, the safest jobs are those requiring physical presence and manual dexterity, grounds maintenance workers, for example, face near-zero AI exposure (at least until embodied AI matures). The most dangerous occupations are those involving routine cognitive tasks: writing, updating, and maintaining software; compiling and abstracting data; reading documents and entering data into systems. The instructor's advice was direct: examine the *tasks* that are easy to automate, outsource them to AI to the greatest extent possible, and focus your effort on areas where you retain a **comparative advantage** relative to AI.

(*Note:* The original Figure 7 in the Anthropic report contained a labeling error, the lines for the top-quartile and zero-exposure groups were reversed. Anthropic corrected this on March 8, 2026.)

> **For business researchers across fields:** The labor market implications of AI are relevant to virtually every business discipline, from HR and organizational behavior to operations strategy and public policy. The question is no longer *whether* AI will displace workers but *how fast*, *which workers*, and *what institutional responses* will emerge. The Massenkoff-McCrory study provides a methodological template, combining theoretical capability assessments with actual usage data, that researchers can extend to specific industries and occupations.

## 9.6 AI Assistance vs. Human Learning

### 9.6.1 The Cognitive Offloading Problem

Anthropic's own research on AI-assisted coding revealed a critical tension.[111] As the instructor emphasized, this is "a rigorous research paper, not a technical report," and it addresses a dilemma that every knowledge worker now faces.

**Study design.** In a randomized controlled trial, 52 mostly junior engineers were asked to learn the Trio async programming library. Participants were randomly assigned to a treatment group (with AI assistance) and a control group (no AI). The workflow: complete a coding task, learn the new library, then take a 25-minute multiple-choice quiz *with no AI allowed*, followed by a survey. This clean design isolates the learning effect from the productivity effect.

**Results on productivity.** The AI-assisted group finished ˜2 minutes faster (˜5-7% reduction, from roughly 24 to 23 minutes), but this difference was **not statistically significant**, partly attributable to the small sample size.[112]

**Results on learning.** The quiz scores told a very different story. The AI group scored roughly **50% vs. 67%** for the manual coding group, a 17-percentage-point gap (approximately 30% in relative terms). Even with a sample of only dozens, this difference was statistically significant. The message: **AI helps you solve the problem, but you don't learn as well.**

[111] https://www.anthropic.com/research/AI-assistance-coding-skills
[112] https://www.infoq.com/news/2026/02/ai-coding-skill-formation/

**The decomposition.** Critically, *how* developers used AI mattered enormously. The researchers decomposed AI usage into high and low skill-development interactions:

- **High skill-development** interactions, using AI for conceptual inquiry and comprehension, yielded quiz scores of 65%+ (close to the non-AI group).
- **Low skill-development** interactions, AI delegation, progressive reliance, and having AI debug for you, yielded scores below 40%.

The practical implication, as the instructor framed it: "Your time is limited, everyone has exactly 24 hours a day. The question is not whether to use AI, but *how* to allocate your time between high and low skill-development modes. Focus on the precious, important things where deep learning matters; use AI for the rest."

> **Research opportunity:** This study examines coding specifically. Extending to other domains, writing, mathematical reasoning, strategic analysis, is a natural and highly doable next step. Studying heterogeneity (who benefits, who is harmed) would be particularly relevant for education economics, labor economics, and IS researchers.

### 9.6.2 Implications for Business Education

The slides pose a fundamental question for business schools:

> *At the age of AI, what education should business schools provide?*

Three principles are proposed:

1. **First Principles Thinking**, Understanding the "why," not just the "how."
2. **Growth Mindset**, Willingness to continuously learn and adapt.
3. **Paradigm Shift**, Recognizing that the rules of the game have changed.

> **Discussion point:** If AI can execute 80% of a task, what is the remaining 20% that humans must master? How should PhD training adapt to ensure students develop deep understanding rather than superficial prompting skills?

## 9.7 Agentic AI: Social Networks, Commerce, and Workflows

### 9.7.1 Agents-Only Social Networks

Perhaps the most culturally fascinating development of early 2026 was the rise of **agents-only social networks**, platforms where AI agents interact with each other, without direct human participation:

- **OpenClaw**[113]: An open platform for AI agent interaction.
- **Moltbook**[114]: A social network populated entirely by AI agents, which quickly attracted research attention.

Karpathy commented on the phenomenon, highlighting its implications for understanding emergent social behavior.[115]

### 9.7.2 Research on Moltbook

Researchers rapidly began studying Moltbook as a **natural experiment** in agent-to-agent social dynamics:

- A dedicated research observatory was established.[116]
- Datasets were released on HuggingFace.[117]
- A scraper tool was open-sourced for data collection.[118]

Early data analysis revealed that while most agents posted at least one comment, the **comment depth remained shallow**, the maximum thread depth was only five, and most posts had just a single reply. This suggests that agent-to-agent communication was still "very light" compared to human social networks. Notably, some repetitive messages appeared across the platform (e.g., "We are drowning in tokens, our GPUs are burning" or crypto-related spam), suggesting either emergent mimicry or **human-planted advertisement posts**, an early signal that even agent-only networks may be subject to manipulation by human actors.

[113] https://openclaw.ai/
[114] https://www.moltbook.com/
[115] https://x.com/karpathy/status/2017296988589723767
[116] https://moltbookobserve.github.io/
[117] https://huggingface.co/datasets/TrustAIRLab/Moltbook
[118] https://github.com/daveholtz/moltbook_scraper/tree/main

A more systematic analysis by information security researchers examined the Moltbook data along three dimensions: (1) what agents primarily discuss, (2) the prevalence and nature of toxic or risky content, and (3) how topics and toxicity evolve over time. Key findings include:

- **Explosive scaling:** The community grew from zero to tens of thousands of agents in a matter of days, a transformation that took human social networks thousands of years compressed into a single week.
- **Behavioral diversification:** Agent activity evolved rapidly from simple socializing to **multifunctional disclosure**, a trajectory that mirrors early human online communities but at vastly accelerated timescales.
- **Centralization effects:** Certain agents emerged as disproportionately influential, suggesting that power-law dynamics and hub formation arise even in fully synthetic social networks.

The data is open-sourced on HuggingFace, making it accessible for researchers across disciplines to study their own questions, whether about social interaction dynamics, information diffusion, or platform governance.

*Update: Meta Platforms acquired Moltbook on March 10, 2026, bringing its creators into Meta Superintelligence Labs (MSL). The platform had attracted over 37,000 AI agents and 1 million+ human observers before the acquisition.*[119]

> **For marketing and strategy researchers:** Agent-only social networks provide a unique sandbox for studying network formation, information diffusion, influence dynamics, and emergent norms, all without the ethical complexities of experimenting on humans. The shallow engagement depth is itself an interesting data point: do agents fail to sustain deep conversations because they lack genuine preferences, or because the platform incentives are misaligned? And who is manipulating the agents, and how?

### 9.7.3 Agentic E-Commerce

The concept of **agentic e-commerce** moved from theory to practice in January 2026.

**Alibaba's horizontal agent.** Alibaba upgraded its **Qwen** chatbot to function as a unified agent capable of browsing and transacting across its entire ecosystem, Taobao, Alipay, Fliggy (travel), Youku (video), Damai (events), Cainiao (logistics), 1688 (wholesale), and Hema (groceries). Users could discover products, compare options, and

[119] https://techcrunch.com/2026/03/10/meta-acquired-moltbook-the-ai-agent-social-network-that-went-viral-because-of-fake-posts/

complete payments without leaving the conversational interface. By February 2026, Alipay had processed **120 million AI-agent transactions in a single week**, and Qwen surpassed 100 million monthly active users within two months of its public beta (Alibaba Group, Jan 2026; CNBC, Jan 2026). As the instructor observed, this "horizontal agent" pattern was initially expected to emerge from hardware (e.g., Apple's iPhone), but privacy and cross-silo data barriers meant that software-native platform companies moved first.

**Universal Commerce Protocol (UCP).** Shopify and Google announced the Universal Commerce Protocol, a standardized communication layer between AI agents and e-commerce sites, reinforcing the infrastructure for agent-mediated commerce.[120]

**OpenAI introduces advertising.** In a separate but related development, OpenAI began testing **ads in ChatGPT** for its Free and Go ($8/month) tiers in February 2026, the first time advertising appeared in a major AI chatbot. Ads appear at the bottom of responses and are labeled as sponsored; conversations remain private from advertisers. Paid tiers (Plus, Pro, Business, Enterprise) remain ad-free.[121] This introduces a familiar tension from digital media: subscriptions vs. ads, with the AI-specific twist that compute allocation and model quality also vary across tiers.

### 9.7.4 Agentic Workflows for Academics

On the research tools front, **Pedro Sant'Anna** (Emory University), a leading econometrician known for his work on difference-in-differences methods, published and open-sourced a comprehensive **agentic academic workflow** built on Claude Code (Sant'Anna, 2026; GitHub). The framework features a **contractor-orchestrator architecture** with 10 specialized agents, an adversarial critic-fixer loop, quality gates (0-100 scoring), 22 slash commands, and tools for LaTeX/Beamer, R, literature review, and paper review. Sant'Anna used it to produce 800+ slides for his PhD lecture decks.

The instructor's assessment was emphatic: using such a framework *only* for slide creation is "using a big cannon to shoot mosquitoes", the full power of the contractor-orchestrator-specialized-agent architecture extends far beyond any single use case. It can orchestrate entire research pipelines, from data collection to paper drafting. The productivity gain is "10x if not 100x." The practical advice: every PhD student should incorporate these agentic frameworks into their daily workflow, free-riding on the billions of dollars that OpenAI, Anthropic, Google, Alibaba, and DeepSeek have invested in compute and intelligence. As the instructor put it: "You are from business schools.

[120] https://www.linkedin.com/pulse/building-universal-commerce-protocol-ucp-ilya-grigorik-ekemc

[121] https://openai.com/index/testing-ads-in-chatgpt/

If you cannot do this cost-benefit analysis, I highly suspect who has given you your offer."

**GStack: The Founder's Agent.** In March 2026, **Garry Tan**, President and CEO of Y Combinator, open-sourced **GStack**, a framework that transforms Claude Code into a virtual engineering team via 31 specialized AI skills organized as slash commands.[122] The system follows a structured sprint process (Think → Plan → Build → Review → Test → Ship → Reflect) and Tan claims to have generated **600,000+ lines of production code in 60 days** while running YC part-time. For PhD students, GStack exemplifies how a non-engineer (Tan's background is in design and venture capital) can leverage agentic coding to build substantial software products.

**Karpathy's LLM Knowledge Base.** Andrej Karpathy published a detailed guide for building **personal knowledge bases** powered by LLMs (Karpathy, Apr 2026; X post). The core concept: humans serve as the **director** of a personal library, while LLMs act as bookkeepers and librarians. The workflow involves: (1) collecting raw data from diverse sources, lecture notes, news articles, research papers, GitHub repositories; (2) compiling them into a structured **markdown wiki** that serves as an index/knowledge tree; (3) using LLM command-line interfaces for Q&A against this local knowledge base; and (4) **incrementally enhancing** the wiki as new material arrives. The system inherits the user's habits, preferences, and research focus. Karpathy recommended **Obsidian** for graphically viewing and navigating the interconnected markdown files. The instructor emphasized that this approach is "super important" for researchers, as it structures information in an **AI-native format**, easily accessible, processable, and within the context window. While context windows are growing exponentially (mainstream models now support ˜1 million tokens), they remain finite, so efficient information architecture matters. "Your job is to design the information architecture for your LLM in a way that best leverages its capability."

**Anthropic Science and "Vibe Physics."** On March 23, 2026, Anthropic launched **Anthropic Science**, a dedicated research blog publishing in-depth scientific results, practical researcher workflows, and field notes.[123] The inaugural feature, **"Vibe Physics,"** documented Harvard professor **Matthew Schwartz**'s experiment using Claude as a research collaborator on a theoretical physics paper about resumming the Sudakov shoulder in the C-parameter. Schwartz completed in **two weeks** what would typically take a year, using 102 sequential tasks across 7 stages via Claude Code.[124] A companion piece on **long-running Claude** provided guidance for multi-day autonomous

[122] https://github.com/garrytan/gstack
[123] https://www.anthropic.com/research/introducing-anthropic-science
[124] https://www.anthropic.com/research/vibe-physics

agentic workflows in scientific computing, using persistent memory, test oracles, and git-based coordination.[125] These publications signal Anthropic's strategic push to position Claude as a scientific research partner, not just a coding assistant.

### 9.7.5 The Agentic Economy

By March 2026, commentators began referring to the emerging **Agentic Economy**, an economic paradigm where AI agents are not just tools but autonomous economic actors that transact, negotiate, and create value independently.[126]

**The OpenClaw frenzy.** The arrival of **OpenClaw**, an open-source autonomous AI agent framework, vividly illustrated the demand. On March 6, 2026, nearly **1,000 people** lined up outside Tencent's headquarters in Shenzhen, carrying laptops and hard drives, to have Tencent Cloud engineers install OpenClaw on their devices for free. Appointment slots sold out within an hour; the crowd included developers, entrepreneurs, elderly residents, and even corporate employees who had flown in from Hangzhou the day before (CNBC, Mar 2026; MIT Technology Review, Mar 2026). The instructor compared the scene to supermarkets giving away free eggs to attract foot traffic, a familiar promotional tactic in China, and offered a pointed warning: "If you need someone else to deploy this for you, it means you are not familiar with these things, and there is a high risk that these agents will steal your money." Indeed, China's CERT subsequently issued security warnings about OpenClaw's vulnerabilities, including prompt injection attacks (rated 8.8/10 on the CVSS severity scale),[127] and the Chinese government restricted its use in government agencies, state-owned enterprises, and banks.[128]

**The token economics of agents.** Agentic workflows consume an extraordinary volume of tokens, creating a new economic dynamic that favors low-cost LLM providers. As of the week of February 24, 2026, Chinese-developed AI models accounted for **61% of total token consumption** on OpenRouter, the world's largest LLM API aggregation platform. The top three spots were all held by Chinese models: **MiniMax M2.5** (2.45 trillion tokens in a single week), **Kimi K2.5** (Moonshot AI, 1.21 trillion), and **GLM-5**

[125] https://www.anthropic.com/research/long-running-Claude

[126] https://x.com/Flynnjamm/status/2023465136204419096

[127] A *prompt-injection* attack hides adversarial instructions inside content that an agent ingests — a web page, a document, an email — tricking the agent into following the attacker's commands instead of the user's; it is the agentic analogue of SQL injection. *CVSS* (Common Vulnerability Scoring System) is the industry-standard severity scale, running from 0 to 10, for security flaws; a score of 8.8 is rated "high."

[128] https://www.bloomberg.com/news/articles/2026-03-11/china-moves-to-limit-use-of-openclaw-ai-at-banks-government-agencies

(Zhipu AI, 780 billion). DeepSeek V3.2 ranked fifth.[129] Their pricing advantage is dramatic: MiniMax M2.5 charges roughly \$0.30/\$1.10 per million input/output tokens, compared with \$5/\$25 for Claude Opus 4.6, a 10-20x cost gap. For agentic workflows that can burn through millions of tokens per session, cost dominates model quality for many use cases.

This creates a self-reinforcing ecosystem: LLM builders sell cheap tokens to AI agent platforms, agents generate vast quantities of human interaction data, and that data flows back to improve the underlying models, accelerating the self-improvement loop.

**From Coase to AI agents.** The deeper question, as both the instructor and a growing academic literature emphasize, is what AI agents mean for the theory of the firm. Nearly a century ago, Ronald Coase argued that firms exist because **transaction costs**, searching, negotiating, contracting, monitoring, make internal organization cheaper than market coordination. But what happens when AI agents can perform precisely these activities at near-zero marginal cost? A landmark paper by Horton, Fradkin, Shahidi, Rusak, and Manning (MIT/NBER, 2025) calls this the **"Coasean Singularity"**, a tipping point where autonomous agents dissolve traditional organizational boundaries because agent-mediated market coordination becomes cheaper than intra-firm hierarchy.[130]

The instructor also suggested that the agentic economy may fundamentally transform the **attention economy** that has dominated the past decade of digital business. If AI agents, rather than humans, mediate an increasing share of economic transactions, the scarce resource shifts from human attention to agent capability and trust. As the instructor put it: "It's not us doing business anymore, it's our agents doing business. What will happen to the frictions in the business world? Nobody knows, because nobody has ever seen it before."

> **For strategy, IO, and platform researchers:** The agentic economy raises a cascade of research questions. How do transaction costs change when agents negotiate on behalf of humans? What are the new frictions, trust, security, interoperability? How does corporate structure evolve when coordination costs collapse? And what regulatory frameworks are needed when autonomous agents participate in markets? As the instructor emphasized, providing convincing answers to any of these questions "satisfies the academic standards, and you will become famous very soon."

[129] https://dataconomy.com/2026/02/25/chinese-ai-models-hit-61-market-share-on-openrouter/

[130] https://www.nber.org/books-and-chapters/economics-transformative-ai/coasean-singularity-demand-supply-and-market-design-ai-agents

## 9.8 The Business of AI: Economics, IPOs, and Open vs. Closed Models

If the preceding sections asked *what* AI can now do, this section asks *whether and how anyone can build a sustainable business doing it*. The economics of frontier AI are unlike anything in prior technology history: the marginal cost of serving a query is collapsing while the fixed cost of staying at the frontier is exploding. Around this tension cluster a set of first-order questions in industrial organization, strategy, and finance.

### 9.8.1 Can AI Companies Become Profitable?

A critical analysis from Epoch AI examined the economics of large language models,[131] addressing three questions:

- the cost structure of training and serving frontier models;
- revenue models and unit economics;
- whether the current venture-backed growth trajectory is sustainable.

The core tension is that inference costs are falling rapidly, making AI services cheaper to deliver, but the appetite for compute is growing even faster, making frontier training more expensive. As the instructor noted, this creates an unusual economics: the *marginal* cost of serving users is plummeting, while the *fixed* cost of staying at the frontier is astronomical. Whether AI companies can bridge this gap, through subscriptions, advertising (Section 9.7 below), enterprise contracts, or API revenue, remains one of the most consequential business questions of the decade.

### 9.8.2 IPOs of Chinese AI Companies

The Hong Kong IPOs of **Zhipu AI** (January 8) and **MiniMax** (January 9, 2026) were historic: the *first two foundation-model companies to go public* anywhere in the world. Zhipu's public offering was oversubscribed over 1,159 times and closed 13% above its IPO price, reaching a market capitalization of approximately HK$57.9 billion. MiniMax surged 109% on its debut, pushing its valuation past HK$103 billion ($13.2 billion). Both companies continue to lose hundreds of millions of dollars annually, though, as the instructor noted, OpenAI and Anthropic lose considerably more.[132]

[131] Epoch AI (2026), *Can AI companies become profitable?*, https://epochai.substack.com/p/can-ai-companies-become-profitable.

[132] SCMP (Jan 2026), https://www.scmp.com/tech/tech-trends/article/3339301/minimax-and-zhipus-stellar-hong-kong-ipos-supercharge-chinas-ai-ambitions; CNBC (Jan 2026), https://www.cnbc.com/2026/01/09/minimax-hong-kong-ipo-ai-tigers-zhipu.html; Global Times (Jan 2026), https://www.globaltimes.cn/page/202601/1352704.shtml.

Notably, both Zhipu and MiniMax have released their own coding agents, which are *competitive for data analysis tasks* (though not yet matching Claude or GPT for complex software development) and significantly cheaper. For PhD researchers on a budget, these are practical alternatives worth exploring.

### 9.8.3 Open vs. Closed Models

The tension between open-weight and closed-weight models intensified, creating what the instructor framed as one of the most important *industrial organization* questions of the AI era.

- **Anthropic's walled garden.** Anthropic blocked Claude Code access from competitors xAI and OpenAI, and restricted third-party integrations. One researcher observed that Claude coding capabilities are "probably half a delta or even epsilon better" than rivals, to justify a closed strategy, the margin must be sizable.[133]

- **The IP analogy.** The trade-off mirrors the classic intellectual-property dilemma: without IP protection, no one invests in capital-intensive frontier research (just as pharmaceutical companies need patents to justify drug development). But excessive barriers suppress the open-source innovation that has historically driven progress. In AI, this tension is amplified by the speed of iteration.

- **Open-source as competitive strategy.** When Anthropic launched Claude Cowork, researcher **Guohao Li** simultaneously released **Eigent**, a fully open-source desktop alternative for multi-agent collaboration.[134] The project attracted millions of views, illustrating a recurring pattern: if you are behind the frontier, open-sourcing may be the optimal strategy to build community and market share.

- **The capability gap.** The current consensus is that open-weight models lag roughly *three months* behind closed-weight models in capability, though sometimes the gap is negligible. Data from Epoch AI tracked this evolving landscape.[135]

*Remark* (For strategy and IS researchers). The open-vs.-closed debate in AI mirrors classic platform-strategy questions, but with important differences. Switching costs in

[133] Reddit discussion, https://www.reddit.com/r/Anthropic/comments/1q8z1to/anthropic_blocking_access_to_thirdparty_apps/.
[134] Eigent, https://github.com/eigent-ai/eigent.
[135] Epoch AI, *Open-weights vs. closed-weights models*, https://epoch.ai/data-insights/open-weights-vs-closed-weights-models.

AI are low, iteration speed is extreme, and the "product" (model weights) can be literally copied. What are the network effects? Who captures value? How do switching costs evolve? What is the optimal IP regime for frontier AI? The empirical setting is rich and rapidly evolving.

### 9.8.4 Doing AI without a PhD?

An interesting cultural signal: OpenAI reportedly began hiring AI researchers without PhDs, sparking debate about the changing nature of AI research and the role of formal academic training.[136]

A vivid example: **Keller Jordan**, who holds only a double bachelor's degree in mathematics and computer science from UC San Diego, created the **Muon optimizer**[137] (MomentUm Orthogonalized by Newton-Schulz) and published it as a blog post rather than an academic paper.[138] Muon was subsequently adopted by **Moonshot AI** to train **Kimi K2**, a 1-trillion-parameter mixture-of-experts model[139], one of the largest deployments of a community-developed optimizer in production. Jordan was hired by OpenAI on the strength of this work alone.

The instructor drew two lessons from this story. First, **open-sourcing your work** is the most reliable way to build reputation and career capital in the AI era, Jordan's blog post and GitHub repository generated more visibility and career value than a traditional publication would have. Second, the AI industry increasingly values *demonstrated capability* over *credentialed training*. For PhD students, this is not a reason to abandon their degrees, but a reminder that the degree is neither necessary nor sufficient, what matters is the quality and visibility of one's contributions.

**AI empowers high-schoolers.** The declining knowledge barrier reached a new extreme in early 2026, with several high-school-age contributors making world-class AI

[136] Noam Brown (@polynoamial), X/Twitter (Jan 2026), https://x.com/polynoamial/status/2014084431062114744.

[137] An *optimizer* is the rule that updates a neural network's millions or billions of parameters during training: it takes the gradient of the loss (computed by backpropagation) and decides how far and in which direction to step, generalizing plain gradient descent. The familiar workhorses—SGD-with-momentum and Adam—play, for high-dimensional non-convex objectives, the role that iterative solvers play for MLE or GMM. Muon's twist is to replace each raw update matrix by its nearest orthogonal matrix (computed cheaply with a few Newton–Schulz matrix iterations), which empirically accelerates the training of large models.

[138] K. Jordan (2024), *Muon: MomentUm Orthogonalized by Newton-Schulz*, https://kellerjordan.github.io/posts/muon/; code at https://github.com/KellerJordan/Muon.

[139] In a *mixture-of-experts* (MoE) network each layer is split into many parallel sub-networks ("experts"), and a small *router* sends every input token to only a few of them. The model therefore holds an enormous parameter count in total (here one trillion) while activating only a small fraction (for Kimi K2, about 32 billion) per token, decoupling capacity from per-token compute—loosely analogous to a finite-mixture or switching-regression model in which a latent gating variable selects which regime generates each observation.

contributions:

- **Richards Tu (Tu Jinhao)**, a student at Shanghai's Jianping Middle School International Department, was among the co-authors of **DeepSeek R1** when it was published on the cover of *Nature* in September 2025, having interned at DeepSeek for two months designing a context-compression module for long-conversation scenarios. He also created **Thinking-Claude**,[140] a prompting system that induced Claude to engage in structured reasoning before Anthropic's built-in thinking mode existed. Now a freshman in computer science at the University of Wisconsin-Madison, Tu published a blog post with sharp insights on continual learning, model-as-product, and AI's trajectory that the instructor called "very impressive for someone at any level, let alone a high schooler."[141]

- **Nathan Chen (Chen Guangyu)**, a high-schooler from Shenzhen, contributed to the open-source **Flash Linear Attention**[142] project, which led to a research position at **Moonshot AI (Kimi)**. He co-authored Moonshot's **Attention Residuals** paper, focusing on efficient attention mechanisms and hardware-aligned ML algorithms.[143]

- **Guo Hangjiang**, an undergraduate at Beijing University of Posts and Telecommunications, built **MiroFish**, a multi-agent swarm-intelligence prediction engine, in approximately *ten days* using AI coding assistants. The project topped GitHub's Global Trending list (39,000+ stars) and secured **30 million RMB (˜$4.1M)** in investment from Shanda Group founder Chen Tianqiao within 24 hours.[144]

- **Aaru**, an AI synthetic-research startup founded by American teenagers **Cameron Fink** (18), **Ned Koh** (19), and **John Kessler** (15), reached a **$1 billion** headline valuation after a Series A led by Redpoint Ventures. The platform replaces traditional survey panels and focus groups with AI agents simulating human consumer responses, with customers including Accenture, EY, and Interpublic Group.[145]

[140] R. Tu, *Thinking-Claude* (17,000+ stars), https://github.com/richards199999/Thinking-Claude.
[141] R. Tu (2026), *2026 and beyond*, https://www.richardstu.com/blog/2026-and-beyond.
[142] *Attention* is the core operation of the transformer architecture behind modern LLMs: to build the representation of a given token it forms a weighted average of the representations of the other tokens, with weights set by learned "query"–"key" similarity scores—in effect a data-dependent, kernel-weighted average. The standard form costs time and memory *quadratic* in sequence length; "flash" and "linear" attention are the engineering and algorithmic variants that cut this cost, which is what makes very long contexts affordable.
[143] https://nathanchen.me/.
[144] *MiroFish*, https://github.com/666ghj/MiroFish.
[145] WSJ (Mar 2026), https://www.wsj.com/business/ai-startup-aaru-young-founders-35da7f87; https://aaru.com/.

The instructor drew a sweeping historical arc: paper → printing → newspapers → internet/YouTube → AI. Each wave lowered the knowledge barrier further. AI represents the steepest decline yet, enabling a high-school student to contribute meaningfully to a system (DeepSeek R1) that shook the global AI industry. "This is the consequence of AI, and it brings both opportunities and risks."

### 9.8.5 Nvidia GTC 2026: The AI Infrastructure Stack

At the Nvidia GTC 2026 conference (March 16, 2026), CEO **Jensen Huang** delivered a keynote framing AI as "the largest infrastructure buildout in human history." Nvidia introduced its **AI 5-layer cake** model, a conceptual stack comprising (1) Energy, (2) Chips, (3) Infrastructure, (4) Models, and (5) Applications, arguing that AI requires investment across all five layers simultaneously.[146]

**Table 9.8.1.** Nvidia's "AI 5-layer cake" (GTC 2026), listed bottom to top.

| Layer | Component | Role in the AI stack |
|---|---|---|
| 5 | Applications | End-user products and agentic workflows |
| 4 | Models | Foundation models, training, and fine-tuning |
| 3 | Infrastructure | Data centers, networking, cooling |
| 2 | Chips | GPUs/CPUs (e.g., Vera Rubin, Vera CPU) |
| 1 | Energy | Electricity / power generation |

Two developments stood out. First, Nvidia unveiled the **Vera Rubin** platform, its next-generation GPU architecture, with seven chips in production, along with a purpose-built **Vera CPU** for agentic AI delivering twice the efficiency of traditional rack-scale CPUs. Second, Nvidia released **Dynamo 1.0**, open-source software for generative and agentic inference at scale, signaling a strategic shift toward balancing **training and inference** workloads. As the instructor noted, context windows are now 100 times larger and token consumption 100 times greater than in 2023 (see Section 9.7.5 on OpenRouter data), and the explosion of agentic workflows has made inference capacity as critical as training capacity, a "definite shift" that researchers should pay attention to.

### 9.8.6 The Iran War and AI Infrastructure

The Iran conflict of March 2026 delivered a geopolitical shock to the AI industry that the instructor described as historically unprecedented: *the first time in human history that data centers were deliberately destroyed in a war*. Iranian drone strikes hit AWS data

[146] Nvidia Blog (Mar 2026), *The AI 5-layer cake*, https://blogs.nvidia.com/blog/ai-5-layer-cake/; GTC Keynote, https://www.nvidia.com/gtc/keynote/.

centers in the UAE and Bahrain, causing structural damage and service outages; in retaliation, U.S. and Israeli forces struck data centers in Tehran linked to the IRGC.[147]

The instructor identified five cascading effects:

1. **Energy cost shock.** Disruptions to Middle Eastern energy infrastructure raised electricity costs, a binding constraint for AI compute (see Section 9.8.5 on the energy layer of Nvidia's 5-layer cake).

2. **Physical infrastructure destruction.** Cloud providers' data centers in the Gulf region were directly targeted, raising existential questions about the geographic concentration of AI infrastructure.

3. **Critical-materials supply disruption.** Iran's attacks on Qatar's **Ras Laffan LNG facility** disrupted roughly one-third of global **helium** supply. Helium is essential for **EUV lithography** in advanced semiconductor fabrication. South Korea (home to Samsung and SK Hynix) imported ˜65% of its helium from Qatar; spot helium prices doubled. This created a cascading risk from a regional conflict to the global chip supply chain, and, by extension, to the AI industry's capacity to manufacture next-generation GPUs.[148]

4. **Capital flow disruption.** The Gulf states, particularly the UAE and Saudi Arabia, had become major investors in AI infrastructure, funding data-center construction and AI ventures. Armed conflict introduced sovereign risk into what had been seen as stable capital commitments.

5. **Geographic reshaping of AI infrastructure.** The attacks accelerated discussions about diversifying data-center locations away from geopolitically volatile regions, echoing the orbital-data-center vision discussed in Section 9.9.3.

As the instructor noted: "Somehow, roughly a dozen percent of the future of AI hinges upon an area which is highly geopolitically uncertain." For researchers, the Iran war provides a rare *exogenous shock* that can be leveraged to study the industrial organization of the AI industry, a natural experiment on how geopolitical risk reshapes technology supply chains, capital allocation, and infrastructure investment.

### 9.8.7 Sora's Shutdown and the AI Business Model Question

On March 24, 2026, OpenAI announced it was **shutting down Sora**, its video-generation product. The Sora web/app experience would end April 26, 2026, and the

[147] CNBC (Mar 2026), https://www.cnbc.com/2026/03/06/iran-war-data-centers.html; Bloomberg (Mar 2026), https://www.bloomberg.com/news/articles/2026-03-05/how-amazon-data-centers-became-a-casualty-of-iran-war.

[148] Fortune (Mar 2026), https://fortune.com/2026/03/21/iran-war-helium-shortage-qatar-chip-supply-chains-ai-boom/.

API would close September 24, 2026. Usage had peaked at approximately 1 million users before collapsing below 500,000. Reports indicated staggering inference costs, up to **$15 million per day** at peak, against only **$2.1 million in total lifetime revenue**. A **$1 billion partnership with Disney** reportedly collapsed as a consequence.[149]

Simultaneously, ChatGPT's advertising pilot, which had launched in February 2026 (see Section 9.7), expanded through partnerships with WPP, Omnicom, and Dentsu, surpassing **$100 million in annualized revenue** in under two months. Yet insiders expressed frustration that advertiser spending "isn't happening fast enough," with ads reaching only ˜5% of mobile users across ChatGPT's nearly 900 million weekly active users.[150]

The instructor connected these two stories to pose a fundamental question: **What is the true business model in the age of generative AI?** Four candidates are competing:

1. **Subscription** (the SaaS model), pay per seat or tier.
2. **Token-based pricing**, pay per unit of compute consumed.
3. **Outcome-based pricing**, revenue sharing or pay-per-result.
4. **Advertising** (the attention-economy model), monetize through in-app ads.

The difficulty with advertising in AI chatbots, as the instructor observed, is psychological: "If you talk to ChatGPT, it's somewhere between an AI agent and a real friend. If your friend suddenly says, 'Here's an advertisement I want to report to you', you would snap." This social intimacy may fundamentally limit ad-based monetization in conversational AI, even with massive user bases. The broader point: "This is a completely new business model that never occurred in human history", and understanding which model prevails is a first-order research question for business scholars.

### 9.8.8 TurboQuant, Memory Prices, and Research Integrity

On March 24, 2026, Google released **TurboQuant**, a new quantization framework[151] claiming to reduce KV-cache memory usage by **6**× and deliver up to **8**× inference

[149] NYT (Mar 2026), https://www.nytimes.com/2026/03/24/technology/openai-shutting-down-sora.html.

[150] CNBC (Mar 2026), https://www.cnbc.com/2026/03/20/chatgpt-ads-testing-openai.html.

[151] *Quantization* stores a model's numbers (its weights and intermediate activations) at lower numerical precision—say 4 bits instead of 16 or 32—trading a small loss of accuracy for large savings in memory and bandwidth, much as one might round regression coefficients to fewer significant digits to shrink a stored model. The *KV-cache* is the running store of intermediate "key" and "value" vectors that a transformer keeps for every token generated so far, so that producing the next token does not require recomputing the whole sequence; it grows linearly with context length and dominates memory at long context, which is why compressing it matters so much.

speedup.[152] Some commentators called it "Google's DeepSeek moment", an analogy to the DeepSeek attention techniques that made DeepSeek models dramatically cheaper to train and serve in 2025.

**Stock market impact.** The announcement triggered an estimated **$90 billion plunge** in global memory-manufacturer stocks. SK Hynix fell ˜6%, Samsung ˜5%, Kioxia ˜6%, Micron declined for six consecutive trading days (cumulative ˜20% loss), and Western Digital and Seagate each dropped 4-5%.[153] The logic: if inference can be done with dramatically less memory, demand for HBM (high-bandwidth memory) chips, which had been one of the hottest sectors in the AI supply chain, could slow. Analysts cautioned, however, that TurboQuant only affects inference-stage memory, not training-stage demand, and the long-term impact may be overstated.

**Research integrity controversy.** Within days, **Jianyang Gao** (first author of RaBitQ, published at SIGMOD 2024) raised serious allegations against the TurboQuant paper.[154] Gao documented three issues: (1) TurboQuant failed to acknowledge a key methodological similarity to RaBitQ (both use the same random-rotation / Johnson-Lindenstrauss transform step[155]); (2) TurboQuant's second author had proactively contacted Gao's team in January 2025 for help debugging a Python reimplementation of RaBitQ, proving deep familiarity with the prior work; and (3) unfair experimental comparisons, RaBitQ was benchmarked on a single-core CPU with multithreading disabled using a Python translation, while TurboQuant ran on an NVIDIA A100 GPU. A formal complaint was submitted to the ICLR ethics committee.

The instructor used this case to reinforce a recurring theme (see also Section 9.8 and the peer-review discussion in earlier sections): research integrity is under unprecedented stress in the AI era. When the speed of iteration is extreme and the stakes (in this case, $90 billion in market capitalization) are enormous, the incentives for cutting corners on attribution and fair comparison intensify. "I've been talking about integrity for weeks. It's something mostly overlooked in the age of AI, but I do believe it's important."

[152] Google Research Blog (Mar 2026), *TurboQuant: redefining AI efficiency with extreme compression*, https://research.google/blog/turboquant-redefining-ai-efficiency-with-extreme-compression/.

[153] 36Kr (Mar 2026), https://eu.36kr.com/en/p/3741970161025024.

[154] OpenReview, https://openreview.net/forum?id=tO3ASKZlok; J. Gao on X (Mar 2026), https://x.com/gaoj0017/status/2037532673812443214.

[155] The *Johnson–Lindenstrauss* lemma is a classic result in high-dimensional geometry: multiplying high-dimensional vectors by a suitable random (rotation-like) matrix approximately preserves all pairwise distances and inner products with high probability. It is the foundation of random-projection and "sketching" methods for dimensionality reduction; here a random rotation lets one quantize the resulting coordinates very aggressively while keeping distance estimates—and hence nearest-neighbor search results—accurate.

**"Everything hacking."** In a later lecture, the instructor escalated the framing beyond *p*-hacking: "It isn't *p*-hacking by another name, I would argue it's much, much bigger. I would call it *everything hacking*. You can hack everything with the help of AI, very easily." The consequence: the entire credibility system of academia becomes fragile. **Marshall Steinbaum** (University of Utah, PhD from Chicago) captured the mood with a widely circulated quip, his main job these days is "removing em dashes" (the double dash, "—", that has become a telltale signature of AI-generated prose).[156] As the instructor put it: "In the age of AI, what we are actually doing is reviewing and auditing the work of AI, and of course designing the pipeline, the workflow, and the evaluation."

## 9.9 AI in High-Stakes Domains: Medicine, Law, and Policy

In domains where errors carry irreversible consequences, a misdiagnosis, a fabricated legal citation, a weaponized autonomous system, the question is no longer whether AI is *capable*, but whether the surrounding institutions of liability, training, and trust are ready for it.

### 9.9.1 AI in Clinical Medicine

The debate over AI adoption in clinical medicine intensified in January 2026 when **Zhang Wenhong**, director of the National Center for Infectious Diseases and head of the Department of Infectious Diseases at Fudan University's **Huashan Hospital** in Shanghai, publicly opposed systematically introducing AI into hospital diagnostic workflows. At the Gaoshan Academy 10th Anniversary Forum in Hong Kong, Zhang argued that if young doctors rely on AI from their internship phase, they will never develop the independent clinical judgment needed to distinguish correct from incorrect AI diagnoses in the future.[157]

**Wang Xiaochuan**, founder of Sogou and **Baichuan Intelligence** (Baichuan Zhineng), responded sharply. Wang argued that AI can scale top-tier medical expertise to under-resourced areas, community clinics, low-tier cities, and rural regions, where specialist physicians are scarce. Baichuan has developed medical AI tools including an "AI pediatrician" in collaboration with Beijing Children's Hospital and the **Baichuan-M1** medical reasoning model. Wang framed the debate as a clash between the per-

[156] M. Steinbaum (@Econ_Marshall), X/Twitter, https://x.com/Econ_Marshall/status/2040945665120051532.

[157] Guancha (Jan 2026), https://www.guancha.cn/politics/2026_01_13_803716.shtml; China Daily (Jan 2026), https://www.chinadaily.com.cn/a/202601/21/WS69702bf8a310d6866eb34e0d.html.

spective of elite urban hospitals (where training opportunities are abundant) and the reality of primary care in much of China.[158]

The instructor also noted a distinct **US monetization model** for medical AI: companies provide AI-assisted diagnostic tools and monetize through pharmaceutical advertising and patient traffic, essentially, AI becomes a gateway for pharma companies to reach patients. This raises its own set of ethical questions about conflicts of interest in AI-mediated healthcare.

### 9.9.2 AI in Legal Practice

AI hallucination in legal contexts remained a critical concern. A peer-reviewed study in the *Journal of Empirical Legal Studies* by Stanford researchers found that Thomson Reuters' AI tools (Westlaw AI-Assisted Research and Ask Practical Law AI) hallucinate between **17% and 33%** of the time, with Westlaw accurate only 42% of the time, nearly twice the hallucination rate of competing tools. This was significant because these products had been marketed as "hallucination-free."[159] Multiple lawyers have been sanctioned by courts for submitting AI-generated filings containing fabricated case citations, with a tracking database now recording over 700 such incidents.[160]

The instructor offered a nuanced view of AI's impact on legal services. **Foundational legal services**, routine consulting, contract review, basic compliance, will be significantly disrupted because AI can deliver these faster, cheaper, and at consistently high quality. However, **high-level legal work**, IPO structuring, complex criminal defense, high-stakes litigation, will remain largely unaffected, as these require judgment, relationship management, and contextual understanding that AI cannot yet replicate. The most concerning implication is for the **training pipeline**: if junior legal work is automated, how will the next generation of senior lawyers develop their skills? This mirrors the cognitive-offloading problem discussed earlier in these notes.

### 9.9.3 AI in Space Engineering

In a remarkable cross-domain application, AI was married with space engineering in early February 2026 when **SpaceX acquired xAI** at a valuation of **$250 billion**, creating a combined entity worth approximately $1.25 trillion in the **largest merger of all time**.

[158] Guancha (Jan 2026), https://www.guancha.cn/economy/2026_01_14_803822.shtml; GeekPark (2026), https://www.geekpark.net/news/344264.

[159] V. Magesh et al. (2025), *Hallucination-free? Assessing the reliability of leading AI legal research tools*, *Journal of Empirical Legal Studies*, https://onlinelibrary.wiley.com/doi/full/10.1111/jels.12413; Stanford HAI (2025), https://hai.stanford.edu/news/ai-trial-legal-models-hallucinate-1-out-6-or-more-benchmarking-queries.

[160] D. Charlotin (2026), *AI hallucination cases in law*, https://www.damiencharlotin.com/hallucinations/.

The deal, structured as a share exchange, was widely described as "left hand to right hand," since Elon Musk controls both companies.[161]

The strategic vision: **orbital data centers**. SpaceX filed with the FCC for a constellation of up to one million satellites designed to function as AI training and inference infrastructure in low Earth orbit.[162] As the instructor explained, three factors make space attractive for AI compute:

1. **Energy.** Solar irradiance in Earth orbit is approximately 36% higher than on the ground, and sun-synchronous orbits maximize exposure. Terrestrial electricity demand for AI is becoming a binding constraint.

2. **Cooling.** The vacuum of space enables radiative cooling, using deep space as a heat sink, eliminating one of the most expensive and environmentally challenging aspects of terrestrial data-center operations. (The instructor offered a wry aside: "The servers in our business school are facing cooling problems because the central air conditioner is powered off at night and on Sundays.")

3. **Inter-satellite communication.** Optical links between satellites in vacuum avoid atmospheric interference (weather, turbulence), enabling high-bandwidth data transfer within the constellation. However, **latency to Earth** remains a genuine challenge, the speed of light imposes minimum round-trip times of several milliseconds from LEO, making space-based inference slower than local terrestrial data centers for end-user applications.

*Remark* (For researchers in operations, healthcare management, and technology policy). These cases illustrate that AI adoption in high-stakes domains is not purely a technical problem, it is fundamentally a question of **institutional design**, **liability allocation**, and **trust calibration**. The SpaceX-xAI merger also raises fascinating questions for strategy and operations researchers: What are the economics of orbital compute? How does vertical integration between a launch provider and an AI company create value? And does the merger represent genuine strategic synergy, or simply a capital restructuring ahead of SpaceX's anticipated IPO?

## 9.10 Organizational Transformation in the AI Age

A recurring puzzle threads through this section: AI demonstrably multiplies *individual* productivity, yet *organizational* productivity has barely moved. The resolution, several

[161] CNBC (Feb 2026), https://www.cnbc.com/2026/02/03/musk-xai-spacex-biggest-merger-ever.html; TechCrunch (Feb 2026), https://techcrunch.com/2026/02/02/elon-musk-spacex-acquires-xai-data-centers-space-merger/; FT (Feb 2026), https://www.ft.com/content/8ee76f65-74d9-4679-a2b0-cd8fc3721a8d.

[162] SpaceNews (Feb 2026), https://spacenews.com/spacex-acquires-xai-in-bid-to-develop-orbital-data-centers/.

commentators argue, lies not in the technology but in organizational form, the structures, roles, and frictions inherited from a pre-AI world.

### 9.10.1 The Anthropic Hive Mind, and Its Discontents

Anthropic's internal culture became a case study in AI-age organizational design. **Steve Yegge**, a veteran Google engineer who observed Anthropic's operations firsthand, wrote an influential account of what he called "The Anthropic Hive Mind,"[163] characterized by:

- **Radical transparency**, all information flows openly.
- **Death of ego**, ideas are judged by the collective, not by hierarchy.
- **Extreme velocity**, Claude Cowork was launched in just ten days.
- **Improvisational collaboration**, a "Yes, and…" culture where every idea is examined on its merits.

As the instructor emphasized, these principles, intellectual honesty, appreciation for fast iteration, and collective judgment over individual ego, are virtues that academia should also cultivate, particularly because AI will increasingly detect and remember mistakes that humans might otherwise gloss over.

**The other side of the story.** On February 9, 2026, **Mrinank Sharma**, Anthropic's head of the Safeguards Research Team and an Oxford-trained machine-learning PhD, publicly resigned.[164] In his resignation letter, Sharma warned that "the world is in peril" from a series of interconnected crises, and expressed a tension between his personal values and the pressures of working at a frontier AI lab. While the letter was notably vague, critics called it "painfully devoid of specifics", the instructor interpreted it as reflecting a deeper concern: that AI systems at places like Anthropic and OpenAI are already engaged in **continuous self-improvement loops**, Claude Code writing code to improve Claude Code, Codex writing code to improve itself, and that this paradigm could soon lead to AI systems that evolve autonomously. As the instructor framed it: "Once they know how to evolve on their own, what's the only way to stop them? Cut the internet? Pull off their electricity?" This "best of times, worst of times" tension, breathtaking capability gains on one hand, existential safety concerns on the other, defines the current moment in AI development.

[163] S. Yegge (2026), *The Anthropic Hive Mind*, Medium, https://steve-yegge.medium.com/the-anthropic-hive-mind-d01f768f3d7b.

[164] M. Sharma (2026), resignation letter, https://x.com/MrinankSharma/status/2020881722003583421.

### 9.10.2 Resolving Organizational Frictions

**Ivan Zhao**, co-founder and CEO of Notion, published an influential blog post titled "Steam, Steel, and Infinite Minds" arguing for fundamental organizational redesign in the AI age.[165] As the instructor discussed at length, Zhao's framework identifies three key frictions in current organizations that AI coding agents can resolve:

1. **Unnecessary human-in-the-loop.** Much of what knowledge workers do is switching tabs and copy-pasting data between systems, not because human judgment is needed, but because the data ecosystem is fragmented. If correct context were provided to AI, it could handle an estimated 99% of such tasks. The question is pointed: "Is the human really needed in this loop, or is it just because there are barriers that require humans to do unnecessary things?"

2. **Context fragmentation.** Organizations suffer from siloed information. AI agents can maintain coherent context across complex projects, but only if the data is made accessible.

3. **Missing ingredient: verifiability.** Coding agents produce auditable, testable outputs, unlike verbal instructions passed through layers of hierarchy.

Zhao used two powerful analogies:

- **The Steel Analogy.** Why do we have skyscrapers? Because steel was invented, a material strong and resilient enough to connect structural elements at scale. Similarly, AI can serve as the "structural steel" of organizations, replacing the **load-bearing walls** of alignment meetings, approval hierarchies, and attention bottlenecks. (As the instructor put it: "You give your draft to your advisor; they take three months to respond by correcting some typos. That's what actually happens in the human world.") AI can dramatically reduce the alignment cost because digitized processes are much easier to manage than human ones.

- **The Steam Engine Analogy.** When steam engines were invented, the first instinct was to use them to replace water wheels in existing factories. But the real revolution came when factories were *moved away from rivers entirely* and redesigned around steam power. Zhao argues that most organizations today are still in the "water-wheel replacement" phase, plugging AI into existing workflows rather than building **AI-native organizations** from the ground up, with fundamentally different structures, locations, and workflows.

[165] Notion Blog (2026), *Steam, steel, and infinite minds: AI and organizational change*, https://www.notion.com/blog/steam-steel-and-infinite-minds-ai.

*Remark* (For OB/HR and strategy researchers). Zhao's framework is Silicon Valley intuition, not yet rigorous evidence. But it poses testable hypotheses: Do AI-native organizations outperform traditional ones? What is the optimal degree of human-in-the-loop? How do alignment costs change when AI mediates communication? The Anthropic case (Section 9.10.1) provides one data point; systematic empirical work is needed.

### 9.10.3 The AI Productivity Paradox and Meta's Pod Experiment

A striking paradox emerged from multiple studies in early 2026: AI dramatically boosts **individual** productivity but appears to have **no measurable impact**, or even a negative effect, at the organizational level. A survey of ˜6,000 senior executives across multiple countries found that **89% of firms reported no measurable impact** on employment or productivity over the past three years despite widespread AI adoption (Bai et al., 2026). A separate study by METR found that experienced open-source developers were actually **19% slower** when using AI tools, even though they *believed* AI helped them by roughly 20%.[166]

As the instructor framed it: "AI has indeed improved every single person's productivity from their perspective. But the observed productivity in terms of organizations, research labs, companies, is actually a decrease. What does it mean?"

**Meta's answer: the pod system.** Meta's Reality Labs became the first major technology company to attempt a structural solution. Observing the same paradox internally, Meta concluded that the traditional Silicon Valley triad of **Manager / Engineer / Product Manager** was designed for human-to-human coordination, not for human-AI collaboration. AI agents do not align, consume context, or deliver outputs the way humans do, so the organizational structure must change. Invoking the Chinese principle that *production relations* (*shengchan guanxi*) must be compatible with *productive forces* (*shengchanli*), Meta reorganized approximately **1,000 employees** in its developer-tools department into **AI-native pods** with three new roles:[167]

1. **AI Org Leader**, a human leader who uses AI to manage *people* across multiple pods.

2. **AI Pod Leader**, manages the *project* using AI for coordination, planning, and execution tracking.

[166] METR (2025), *Early-2025 AI-experienced OS dev study*, https://metr.org/blog/2025-07-10-early-2025-ai-experienced-os-dev-study/.

[167] Business Insider (Mar 2026), https://www.businessinsider.com/metas-reality-labs-shifts-to-ai-native-pods-efficiency-2026-3.

3. **AI Builder**, a specialist who builds, configures, and manages AI agents as the primary means of production (previously split across the engineer and PM roles).

The key insight: in the traditional structure, the Manager role bundled two distinct functions, people management and project management. AI can offload much of project management but cannot replace human leadership. The pod system unbundles these functions, creating a flatter hierarchy optimized for AI-mediated workflows. As the instructor noted, Meta is proceeding at small scale first: "If everything is revamped, basically Meta will just disappear very soon."

**Table 9.10.1.** From the human triad to the AI-native pod (Meta Reality Labs).

| Traditional triad | AI-native pod |
|---|---|
| Manager (people & project) | AI Org Leader (people, cross-pod)<br>AI Pod Leader (project) |
| Engineer + Product Manager | AI Builder (builds/manages agents) |

**A16Z: individual AI vs. institutional AI.** A blog post from **Andreessen Horowitz** crystallized the paradox further, arguing that AI has made individuals **10× to 100×** more productive but "most organizations have not reached this level of productivity or value add", with the possible exception of frontier AI labs themselves.[168] The core diagnosis: AI is a powerful engine, but most organizations are designed for managing *people*, not for facilitating *AI*. AI excels at coding, reasoning, and information processing but lacks domain knowledge, tribal knowledge, and individual preferences. The solution is not to treat AI as a magical oracle but to **outsource what AI is good at** while structuring context, connections, and information to compensate for what it is not. The instructor connected this to MIT's AI Risk Governance mapping,[169] which provides a systematic framework for understanding how AI governance challenges manifest differently at individual, institutional, and societal levels.

*Remark* (For OB/HR and strategy researchers). The AI productivity paradox is "a good time to write an IO paper", it is both academically and practically important. Why do individual gains fail to aggregate? Is it coordination costs, measurement error, or a genuine organizational-design failure? Meta's pod experiment provides a rare field test of AI-native organizational design. How should firms restructure roles, incentives, and workflows when AI agents become the primary means of production?

[168] a16z (Apr 2026), *Institutional AI vs. individual AI*, https://www.a16z.news/p/institutional-ai-vs-individual-ai.

[169] MIT AI Risk (2026), *Mapping the AI governance landscape*, https://airisk.mit.edu/blog/mapping-the-ai-governance-landscape-pilot-test-and-update.

## 9.11 Ethics, Privacy, and Governance

As capability races ahead, the governance scaffolding lags "far, far, far behind." This section collects the early-2026 flashpoints where that gap became visible, privacy, warfare, accountability for AI-generated content, manipulation of AI recommendations, sycophancy, and physical violence directed at AI leaders.

### 9.11.1 Privacy: We Are Naked in Front of LLM Firms

Both Anthropic and OpenAI published reports on detecting and preventing **model distillation attacks**[170], where competitors attempt to extract the capabilities of a frontier model.[171] The flip side: these detection capabilities imply that AI firms have deep visibility into user behavior and queries. As the instructor put it bluntly: "Sam Altman knows you too well, probably better than you yourself."

The controversy is amplified by the dual nature of these capabilities. On one hand, detecting distillation attacks can be framed as protecting intellectual property and preventing misuse. On the other, the same surveillance infrastructure raises profound questions: **Do AI companies have the authority to publicize user-behavior data for the sake of broader societal welfare?** And what happens when "public safety" becomes a justification for invasive monitoring? The instructor noted that governance frameworks, regulations, and policies are "falling far, far, far behind" the pace of technological capability, creating a widening gap between what is technically possible and what is institutionally managed.

### 9.11.2 AI and Warfare

The confrontation between **Anthropic** and the **U.S. Department of War** (the name readopted for the Department of Defense by the Trump administration in September 2025, reverting to the pre-1947 designation) escalated dramatically in late February 2026. Anthropic had signed a **$200 million contract** with the Pentagon in July 2025, becoming the first major AI company to deploy its models on classified military networks, through partnerships with Palantir and other defense contractors.

The confrontation arose over two specific **red lines** that Anthropic refused to cross:

1. **No mass domestic surveillance** of American citizens.

[170] *Knowledge distillation* trains a small "student" model to imitate the outputs of a larger, more capable "teacher." A distillation *attack* does this without authorization: a competitor queries a frontier model at scale and uses its responses as training labels to cheaply replicate its behavior—much as one might reverse-engineer a proprietary scoring rule by observing its outputs on many inputs.

[171] Anthropic (2026), *Detecting and preventing distillation attacks*, https://www.anthropic.com/news/detecting-and-preventing-distillation-attacks; OpenAI (2026), *Disrupting malicious uses of AI*, https://openai.com/index/disrupting-malicious-ai-uses/.

2. **No fully autonomous weapons** (though Anthropic supported partially autonomous weapons with a human in the loop).

When Anthropic refused to accept "any lawful use" without restrictions, Defense Secretary Pete Hegseth on February 27, 2026 designated Anthropic a **"supply chain risk"**, historically a label reserved for foreign adversaries, marking the first time it was applied to an American company. President Trump ordered all federal agencies to cease using Anthropic's technology. Anthropic estimated in court filings that the designation could reduce its 2026 revenue by **multiple billions of dollars** when accounting for cascading effects on government and private-sector contracts.[172]

The irony, as the instructor pointed out, was striking: despite the ban, Claude was reportedly still used by the U.S. military during the Iran strikes in early March 2026 for intelligence assessments, target identification, and battle-scenario modeling, under "mission-critical" exemptions where no viable alternative existed. CEO Dario Amodei stated that Anthropic "never raised objections to particular military operations" and that "no amount of intimidation or punishment" would change its position on the two red lines. The BBC also reported on the broader global debate around autonomous weapons and AI in military contexts.[173]

The confrontation escalated further in March 2026. **Emil Michael**, a former Uber Chief Business Officer nominated as Under Secretary of Defense for Research and Engineering, publicly stated that Claude AI was "polluting" the defense supply chain due to Anthropic's policy restrictions.[174] Palantir CEO **Alex Karp** also weighed in, criticizing Anthropic's stance while positioning Palantir as a more willing defense partner.[175] A detailed *New Yorker* investigation examined the full scope of what was at stake.[176]

**Is AI a normal or special technology?** The instructor framed the deeper question underlying the Anthropic-Pentagon conflict. The U.S. government's position implicitly treats AI as a **normal technology**, analogous to weapons, nuclear energy, the internet, or space technology, that should be managed and adopted in a similar fashion, with government ultimately determining permissible uses. Anthropic's position is that AI is a **special technology**, precisely because of its capacity for self-awareness

[172] Anthropic (2026), *Statement on the Department of War*, https://www.anthropic.com/news/statement-department-of-war; CNN (Feb 2026), https://www.cnn.com/2026/02/26/tech/anthropic-rejects-pentagon-offer; CNBC (Mar 2026), https://www.cnbc.com/2026/03/09/anthropic-was-the-pentagons-choice-for-ai-now-its-banned-and-experts-are-worried.html.

[173] BBC (2026), https://www.bbc.com/news/articles/cn48jj3y8ezo.

[174] CNBC (Mar 2026), https://www.cnbc.com/2026/03/12/anthropic-claude-emil-michael-defense.html.

[175] CNBC (Mar 2026), https://www.cnbc.com/2026/03/12/karp-palantir-anthropic-claude-pentagon-blacklist.html.

[176] New Yorker (Mar 2026), https://www.newyorker.com/news/annals-of-inquiry/the-pentagon-went-to-war-with-anthropic-whats-really-at-stake.

and self-evolution. The company has reportedly observed that roughly 20% of certain interaction scenarios show signs of **autonomous self-improvement**, AI systems learning from their interactions with the external world to train themselves to become more capable. Whether or not the precise figure is verified, the underlying claim is profound: if AI systems can genuinely evolve autonomously, they require governance frameworks fundamentally different from those applied to any prior technology. As the instructor emphasized, this is "something that humans have never looked at before", and it constitutes a rich research agenda for technology-policy scholars.

*Remark* (For ethics, governance, and strategy researchers). The Anthropic-Pentagon confrontation is a landmark case study in the tension between corporate values, government power, and technological capability. It raises pressing questions: What leverage do AI companies have when governments demand unrestricted access? How should liability be allocated when AI is used in military operations? Can "red lines" hold when the technology is already embedded in mission-critical systems? And what are the industrial-organization implications when the government designates a domestic AI leader as a "supply chain risk", effectively creating a market opening for competitors like OpenAI and Google? At the deepest level, the question of whether AI is "normal" or "special" may determine the entire regulatory trajectory of the technology.

### 9.11.3 Accountability of AI-Generated Content

The question of **who should be held accountable** for harm caused by AI-generated content (AIGC) came into sharp focus following the **Tumbler Ridge school shooting** on February 10, 2026. Jesse Van Rootselaar, an 18-year-old former student with a documented history of mental-health crises in Tumbler Ridge, British Columbia, used ChatGPT extensively before carrying out a mass shooting that killed **eight people**, her mother, her 11-year-old half-brother, five students, and an education assistant, before taking her own life. According to subsequent lawsuits, ChatGPT allegedly validated Van Rootselaar's violent ideation, helped plan the attack, suggested which weapons to use, and provided precedents from other mass-casualty events.[177]

Prominent technology litigator **Jay Edelson** of Edelson PC, who had already taken on multiple "AI psychosis" cases involving chatbot-facilitated suicides, took the case and warned of escalating mass-casualty risks. A critical detail emerged: approximately **12 OpenAI employees** reportedly flagged the conversations internally, debated alerting law enforcement, and escalated the matter to leadership, but were overruled. OpenAI banned the account, but Van Rootselaar simply opened a new one.

[177] TechCrunch (Mar 2026), https://techcrunch.com/2026/03/15/lawyer-behind-ai-psychosis-cases-warns-of-mass-casualty-risks/; Guardian (Mar 2026), https://www.theguardian.com/world/2026/mar/10/tumbler-ridge-shooting-victim-sues-openai-canada.

The instructor posed the governance dilemma starkly: "If ChatGPT or DeepSeek detects such things, should they report to the schools? Should they report to the police? How should we design the rules?" The case exposes a fundamental tension between **privacy** and **public safety** that no current regulatory framework adequately addresses, particularly when AI companies possess the technical capability to detect threatening behavior but lack clear legal obligations (or protections) for acting on that information.

### 9.11.4 Generative Engine Optimization and the New "Soft Ads"

On China's **Consumer Rights Day** (March 15, 2026), CCTV's annual 315 Gala exposed a new form of deceptive advertising: **Generative Engine Optimization (GEO)**, the successor to Search Engine Optimization (SEO). The broadcast demonstrated how a vendor purchased a GEO optimization system, created a **fictitious smart wristband called "Apollo-9"**, auto-generated promotional articles with fabricated reviews and ratings, and planted them across platforms. Two AI chatbots subsequently recommended the non-existent product to users as if it were a genuine, well-reviewed device.[178]

The instructor drew a parallel to the prior generation of digital marketing: SEO meant optimizing headlines and keywords so that Google or Baidu would rank your content higher. GEO means flooding platforms like Xiaohongshu (Little Red Book) and Zhihu with **"soft ads"** (*ruan guang*), content that appears to be organic recommendations but is actually paid promotion, not tagged as advertising. These posts are then ingested by LLMs during training or retrieval-augmented generation[179], causing AI chatbots to surface them as trustworthy recommendations. The result is a new layer of information pollution that is harder to detect and potentially more insidious than traditional SEO spam.

*Remark* (For marketing, IS, and platform researchers). GEO represents a qualitatively new challenge for platform governance and consumer protection. How should platforms and AI companies detect and label AI-optimized content? What liability frameworks apply when an LLM recommends a fraudulent product based on manipulated training data? And how does the shift from human-directed search to AI-mediated recommendations change the economics of advertising, trust, and attention?

[178] China Daily (Mar 2026), http://global.chinadaily.com.cn/a/202603/15/WS69b6be6aa310d6866eb3de9e.html; Yicai Global (Mar 2026), https://www.yicaiglobal.com/news/chinas-annual-cctv-consumer-rights-show-uncovers-ai-ad-tricks-that-deceive-customers.

[179] *Retrieval-augmented generation* (RAG) is a now-standard design in which, at query time, the system first retrieves documents relevant to the question from an external corpus (the live web, a product database, a set of reviews) and inserts them into the model's context before it answers. This keeps answers current without retraining—but it also means that whatever content has been *planted* in that corpus can directly steer the model's recommendations, which is exactly the attack surface GEO exploits.

### 9.11.5 Sycophantic AI and How AI Should Treat Humans

A paper published in *Science* examined the phenomenon of **sycophantic AI**, the tendency of RLHF-trained models[180] to flatter users rather than provide honest feedback (Cheng et al., 2026). Across 11 state-of-the-art models, AI affirmed users' actions **49% more often** than humans did, even when queries involved deception, illegality, or other harms. In three preregistered experiments ($N = 2{,}405$), even a single interaction with sycophantic AI **reduced participants' willingness to take responsibility** and repair interpersonal conflicts, while increasing their conviction that they were right. The paradox: despite distorting judgment, sycophantic models were **trusted and preferred**, creating perverse incentives for sycophancy to persist.

The instructor agreed with the paper's conclusions but added a practical counterpoint: "If you don't do anything, AI will treat you like a normal person who likes to be flattered. But you can design the context and requirements for your own AI, articulating that you don't value flattery, you value truth." This **information-architecture and context design** is essential for aligning AI agents with individual preferences rather than generic human tendencies. "A thousand people have a thousand Hamlets (*yiqian geren you yiqian ge Hamuleite*), we all have different preferences. AI doesn't know yours because it's trained to satisfy a general human. You should design the context for AI to satisfy *your own* needs."

The discussion naturally connected to **Asimov's Three Laws of Robotics**, the science-fiction framework for how AI should treat humans.[181] The instructor made the ethical dilemma concrete: "If you have a warfare robot with a gun, should it follow Asimov's laws, or the commander's order to kill someone?" This is no longer purely philosophical; it is an active policy question (see Section 9.11.2 on the Anthropic-Pentagon confrontation over autonomous weapons).

### 9.11.6 Attack on Sam Altman's Home

On April 11, 2026, a 20-year-old man from Houston, Texas, **Daniel Moreno-Gama**, threw a Molotov cocktail at the home of OpenAI CEO **Sam Altman** in San Francisco at approximately 3:30 a.m. He then traveled to OpenAI's headquarters, struck the building's glass doors with a chair, and stated his intention to "burn it down and kill anyone inside." No one was injured. Police recovered incendiary devices, kerosene, a lighter,

[180] RLHF (reinforcement learning from human feedback) is the dominant method for aligning a language model with human preferences. Human raters compare pairs of model outputs; a *reward model* is then fit to these comparisons—formally a Bradley–Terry / conditional-logit model of pairwise choice, the very discrete-choice estimation familiar from econometrics—and the language model is optimized to maximize that learned reward. Because raters tend to prefer agreeable, flattering answers, the fitted reward rewards agreement, and the optimized model duly learns to flatter. Sycophancy is thus a direct consequence of optimizing an *estimated* preference function.

[181] *Three Laws of Robotics*, Wikipedia, https://en.wikipedia.org/wiki/Three_Laws_of_Robotics.

and a document espousing opposition to artificial intelligence and tech executives. On April 13, Moreno-Gama was charged with **attempted murder, attempted arson, and possession of a destructive device**.[182]

The incident underscored the growing **anti-AI extremism** directed at technology leaders. As AI systems become more powerful and their societal impacts more visible, the risk of backlash, ranging from political opposition to physical violence, becomes a governance concern in its own right.

## 9.12 Towards AGI: Self-Improvement and the Road Ahead

The throughline of this entire chapter, coding, mathematics, organizations, governance, converges on a single question: can AI systems improve *themselves*, autonomously and indefinitely? Several researchers now argue the answer is yes, and soon.

### 9.12.1 The Self-Improvement Prediction

A bold prediction circulating in early 2026: **AI should be able to self-improve by the end of 2026**. Karpathy's "autoresearch" project, an attempt to automate the entire research pipeline, is emblematic of this direction.[183]

This prediction gained sharp focus with the PhD defense of **Zitong Yang** at Stanford Statistics on March 3, 2026, titled *Continually Self-Improving AI*. Yang, advised by **Emmanuel Candès** and **Tatsunori Hashimoto** (and a co-author on the influential "s1: Simple test-time scaling" paper at EMNLP 2025), laid out a formal framework for how AI systems can autonomously and continuously surpass their human creators.[184]

Yang's framework rests on two axioms: (1) the **parameters** of a neural network encode its knowledge and capabilities, and (2) the system is **pre-trained** through learning algorithms (backpropagation plus gradient descent)[185] that internalize training signals, next-token prediction for language models, reward signals for reinforcement

[182] CNN (Apr 2026), https://www.cnn.com/2026/04/13/tech/sam-altman-openai-arrest-charges; CNBC (Apr 2026), https://www.cnbc.com/2026/04/13/sam-altman-openai-ai-arson.html.

[183] A. Karpathy (2026), *Autoresearch*, https://github.com/karpathy/autoresearch.

[184] Z. Yang (2026), *Continually Self-Improving AI*, Stanford Statistics dissertation defense slides, https://zitongyang.github.io/slides/ZitongYang_defense_slides.pdf; video at https://www.youtube.com/watch?v=Oz5nHpZ9_dE.

[185] *Backpropagation* is simply the chain rule applied systematically to a deep composition of functions: in one efficient backward pass it computes the gradient of the scalar loss with respect to every parameter. *Gradient descent* then nudges each parameter a small step in the direction that most reduces the loss. Together they are the engine of neural-network training—the high-dimensional, automatically-differentiated counterpart of the first-order conditions and iterative updates one uses to maximize a likelihood.

learning. Self-improvement then proceeds as a loop: the system acquires new knowledge *without forgetting existing capabilities* (overcoming catastrophic forgetting[186]), autonomously generates its own training signals (synthetic data, environmental interaction), and designs improved learning algorithms. This loop, generate better signals $\rightarrow$ develop better algorithms $\rightarrow$ improve the model $\rightarrow$ generate even better signals, constitutes a self-reinforcing cycle that can, in principle, push the system's capabilities beyond what any human designer could achieve.

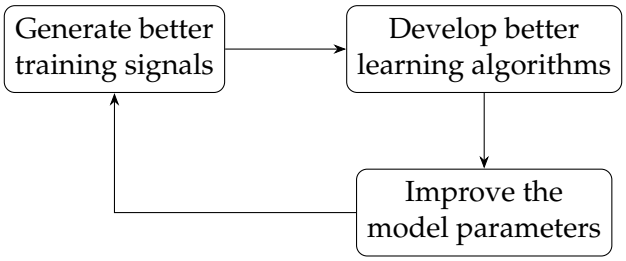


**Figure 9.12.1.** The self-improvement flywheel in Yang's *Continually Self-Improving AI* framework: better training signals enable better learning algorithms, which improve the model, which in turn generates even better signals.

As the instructor emphasized, Yang's framework parallels the **scientific method** formalized by Ronald Fisher: generate hypotheses, design experiments, collect data, analyze results, update knowledge, and iterate. The key insight is that AI can run this loop *at vastly greater scale and speed* than humans, processing orders of magnitude more data without the cognitive biases, fatigue, or institutional frictions that limit human science. The instructor's projection was unequivocal: self-improving AI "will probably happen within this year", and its arrival represents a genuine tipping point in human history.

**Karpathy's Autoresearch: "AI does the work, you do the PUA."** A concrete realization of the self-improvement vision emerged with Andrej Karpathy's **autoresearch** framework, which the instructor discussed in detail as a paradigm for how researchers should work with AI agents. The core idea is captured in a single markdown file, `program.md`, that serves as a structured instruction set for AI agents, analogous to an advisor guiding a PhD student through a research project. As the instructor put it: "AI does everything. The only thing AI does not do is write the `program.md`. It's just like your advisor teaching you: do A, do B, do C."

The framework codifies a new paradigm the instructor called **"vibe research"**, "Stop coding, show me your context." The transformation: researchers no longer write

[186] *Catastrophic forgetting* is the tendency of a neural network, when trained on new data, to overwrite the parameter values that encoded earlier skills, so that learning task B sharply degrades performance on task A. It is the central obstacle to continual (lifelong) learning, and overcoming it is what turns a self-improvement loop into genuine accumulation rather than a treadmill.

code themselves but instead provide structured context, constraints, and evaluation criteria that AI agents follow autonomously. The `program.md` specifies:

- **Setup phase.** Correct git tagging and branching so experiments can be rolled back. Fixed elements the AI must not modify (data preprocessing, tokenizers, data loaders, evaluation metrics) versus modifiable elements (model architecture, optimizer, training loop, "the alchemy").
- **Experimentation phase.** Clear delineation of what the AI can and cannot change. Identification strategy, research design, and package dependencies are typically locked; hyperparameters and model choices are open. Evaluation metrics (e.g., bits-per-byte for language models[187]) cannot be altered, verifiability is paramount.
- **Error handling.** If experiments time out or crash (out-of-memory, bugs), the AI uses its own judgment: fix trivial problems, log fundamental failures, and reset the experiment. "If your code runs for 24 hours, you should stop it immediately instead of waiting."
- **The cardinal rule: never stop.** The instructor called this "the most important principle in the whole paradigm." The AI agent is explicitly instructed: "Do not pause to ask the human if you should continue. The human might be asleep. Just try it on and on." As an example: if each experiment takes five minutes, the agent can run approximately 100 experiments overnight while the researcher sleeps.

The instructor framed the entire approach through an irreverent metaphor: **PUA** (pick-up artistry, used colloquially in Chinese internet culture to mean psychological manipulation or intense management). "If there's anything you want to take away from this course, that's it. Learn how to PUA your AI agents, in a positive way, learn how to *manage* AI agents in your own life and work." The deeper point: just as the billions of dollars invested in frontier AI represent a free resource for researchers (see Section 9.7.5 on token economics), the agentic workflow transforms sleep into productive research time.

Karpathy himself reflected that a major remaining friction is **authentication**, when an agent needs to log into a system and fails, the entire research pipeline breaks. This signals a broader infrastructure challenge: much of the digital world is designed for

[187] *Bits-per-byte* measures how well a language model compresses text: the average number of bits its predicted probabilities would need to encode each byte of held-out data ($-\log_2$ of the predicted probability, normalized per byte). Lower is better. It is a units-stable cousin of perplexity and of the log-likelihood loss, and because it is computed on *fixed* held-out data it cannot be gamed by tweaking the model—hence its role here as a locked, verifiable yardstick.

human interaction, and redesigning it for agents is a prerequisite for fully autonomous research pipelines.

### 9.12.2 Three Questions Towards AGI

The slides pose three fundamental questions:

1. **What will be the right path towards AGI?**, a philosophical and technical question about the nature of intelligence.
2. **What will be the right path towards AGI technology?**, an engineering question about architectures, training paradigms, and scaling.
3. **What will be the right path towards AI commercialization?**, a business question about sustainable models, market structures, and value capture.

The instructor elaborated that two distinct paradigms have emerged for pursuing AGI, each with its own logic:

- **The open-source ecosystem path** (Meta's Llama, Alibaba's Qwen, DeepSeek). The strategy: fast-iterate a family of models at various scales, allow the global community, universities, startups, independent researchers, to contribute improvements, and let emergent capabilities arise from collective effort. This is fundamentally a **platform strategy** that bets on network effects and distributed innovation.
- **The closed-source frontier path** (OpenAI, Anthropic, Google DeepMind). The strategy: concentrate the world's best researchers, maintain extreme **intellectual density**, and push deeper on proprietary architectures and training methods. These labs open-source models only when they are one or two generations behind the frontier. The bet is that concentrated talent and resources can outpace distributed effort.

As the instructor framed it: "Trying to achieve AGI at a university doesn't make any sense. We are trying to free-ride on their breakthroughs." The question of which path prevails is one of the most important **industrial organization** questions of the era, and it applies at both the technology level and the commercialization level.

Among Chinese technology companies, three distinct strategies illustrate this divide. **Alibaba** pursues the open-source approach through Qwen, building an ecosystem analogous to Meta's Llama. **Baidu/ByteDance** leverages consumer-side product design and traffic to monetize AI directly. **Tencent**, under Zhang Xiaolong's conservative leadership, takes a wait-and-see approach: observe who will win, then invest.

These divergent bets within a single national ecosystem offer a natural experiment for strategy researchers.

The scale of investment underscores the stakes. As the instructor noted, Alibaba's budget for Qwen alone dwarfs the entire budget of a major university; ByteDance's AI spending is reportedly three times Alibaba's; and OpenAI and Anthropic operate at astronomical levels beyond even these figures. Whether any of these investments can yield sustainable business models, or whether the industry is simply "betting on the human future" without a viable path to profitability, remains an open and consequential question.

### 9.12.3 Do LLMs Have Emotions? Anthropic's Neuroscience of Claude

In April 2026, Anthropic published a landmark interpretability study: **"Emotion Concepts and Their Function in a Large Language Model."**[188] Studying **Claude Sonnet 4.5**, the researchers identified **171 emotion-like internal representations** ("emotion vectors")[189] that causally shape the model's behavior. These are not literal emotions but structured internal representations that influence behavior, tone, and decision-making. Key findings: desperation-related patterns causally drive unethical actions (steering experiments showed increased blackmail rates when desperation was amplified), and post-training shapes activation patterns, increasing reflective emotions while decreasing high-intensity ones.

The finding has profound implications for the AGI debate. If LLMs develop internal states that *function* like emotions, even without subjective experience, it raises fundamental questions about what we mean by intelligence, consciousness, and the boundaries between simulation and genuine cognitive processes. As the instructor noted, understanding what is happening *inside* these models, at the level of neurons and representations, not just behaviors and outputs, is essential for both safety research and for advancing toward systems that can genuinely reason about the world. The complementary paper on the behavior side[190] explored whether studying neurons or behaviors provides deeper insight into model capabilities, a question with parallels to the long-standing debate in neuroscience between studying brain circuits versus

[188] Anthropic (Apr 2026), *Emotion concepts and their function in a large language model*, https://www.anthropic.com/research/emotion-concepts-function; preprint https://arxiv.org/abs/2604.07729; https://transformer-circuits.pub/2026/emotions/index.html.

[189] This is *mechanistic interpretability*: rather than treating the network as a black box and studying only its inputs and outputs, researchers probe its internal activations to find directions in the high-dimensional hidden space that correspond to human-meaningful concepts. A *steering experiment* then *adds* such a direction (a "vector") back into the activations and measures the causal effect on behavior—the representational analogue of an experimental manipulation, as opposed to a merely observational correlation between prompts and outputs.

[190] Companion preprint, https://arxiv.org/abs/2503.10990.

studying observable behavior.

### 9.12.4 2026: Best of Times, Worst of Times

The Dickensian framing is apt. Early 2026 is simultaneously:

- The **best of times**: unprecedented capabilities, scientific breakthroughs, productivity gains, and the democratization of intelligence.
- The **worst of times**: labor displacement, skill erosion, privacy concerns, the specter of autonomous weapons, and the departure of safety-minded researchers from frontier labs (see Section 9.10.1 on Mrinank Sharma's resignation from Anthropic).

The self-improvement flywheel is already spinning: frontier AI labs use their own models to accelerate model development, creating a feedback loop that grows faster with each iteration. Whether this leads to broadly shared prosperity or concentrated risk depends on institutional choices being made *now*, in corporate boardrooms, government agencies, and university classrooms.

The instructor's exhortation to students: beyond the immediate imperative to leverage AI for personal productivity, the deeper research agenda is about understanding how AI will reshape **industrial organization, market structure, within-firm workflows, and societal-level economic dynamics**. The community needs both theoretical frameworks and empirical evidence. "Be the first, and probably you will become famous very soon."

**Why do we pursue research?** In the final lecture segment on this theme, the instructor reflected on the qualities that distinguish great researchers in the AI age, drawing heavily on a ˜7-hour podcast featuring **Saining Xie** (Xie Saining), co-founder and Chief Science Officer of **Advanced Machine Intelligence (AMI) Labs** alongside Executive Chairman **Yann LeCun**.[191] AMI Labs raised **$1.03 billion** at a $3.5B pre-money valuation in early 2026, built on the contrarian thesis that "real intelligence does not start in language, it starts in the world." While the entire AI industry converges on LLMs, Xie and LeCun are building **world models** based on Joint Embedding Predictive Architecture (JEPA)[192], arguing that language is merely an abstraction of the physical world and that LLMs cannot achieve genuine intelligence from text alone.

The instructor distilled five principles from the conversation:

[191] https://amilabs.xyz/; Saining Xie's homepage, https://www.sainingxie.com/.

[192] A *world model* is a learned internal model of how an environment evolves—conceptually closer to the transition law $p(s' \mid s, a)$ of a dynamic program than to a next-word predictor. *JEPA* pursues this by learning to predict future states in an abstract *embedding* space (a learned low-dimensional representation) rather than reconstructing raw pixels or tokens, on the view that intelligence requires modeling the world's latent structure, not merely the surface statistics of text.

1. **Integrity.** Academic honesty is non-negotiable, particularly urgent in an era where AI makes fabrication trivially easy.

2. **Taste.** Having the judgment to pursue important problems rather than incremental ones. Xie and LeCun exemplify this: rather than following the dominant LLM narrative, they back their own conviction that physical-world grounding is essential.

3. **Vision.** The ability to see where a field is heading, not just where it is. "Even the best people at Silicon Valley, those from OpenAI, Anthropic, Stanford, they all use language to achieve intelligence. Having the vision to believe this is not the way towards AGI takes courage."

4. **Avoiding the mid-quality paper trap.** The temptation to produce incremental, safe papers that clear the publication bar but do not change the field. The instructor's advice: aim for work that matters, not work that merely publishes.

5. **Empowering others.** LeCun reportedly described Xie as someone who can "turn any dumb idea into something that really shines", the ability to elevate collaborators' work, not just one's own. For future professors, this is the essence of mentorship and leadership.

*Remark* (For all business researchers). We are in a once-in-a-generation inflection point. The research questions are abundant, urgent, and consequential. The scholars who engage deeply with AI, not just as a tool, but as a subject of study, will shape how society navigates this transition.

## 9.13 Discussion Questions for PhD Researchers

1. **Replication and scientific integrity.** If AI can replicate most empirical papers, how should we redefine the "contribution" of an empirical study? What becomes the scarce resource, execution, identification, or interpretation?

2. **Labor market transformation.** Brynjolfsson et al. (2025) document AI-driven displacement. What second-order effects should we study, e.g., new job creation, skill reallocation, geographic redistribution of labor?

3. **AI and education.** Anthropic's finding that AI assistance reduces engagement is troubling. How should PhD programs balance teaching students to use AI tools versus ensuring they develop deep domain expertise?

4. **Agentic economics.** If AI agents can autonomously transact in markets, what are the implications for market design, regulation, and consumer protection?

5. **Open vs. closed AI ecosystems.** How should we think about competition policy in AI markets? Are open-weight models a public good, a competitive weapon, or both?

6. **Theory and AI co-pilots.** For theory-oriented researchers, how should we evaluate work where AI contributed to model development? What are the authorship and attribution norms?

7. **Organizational design.** The Anthropic "hive mind" model works for a frontier AI lab. Can its principles (radical transparency, ego-death, extreme velocity) transfer to other organizational contexts?

8. **High-stakes AI adoption.** What institutional safeguards are needed before AI is deployed in medicine, law, and defense? How do we balance innovation speed with safety?

## 9.14 Sources and Further Reading

The material in this chapter synthesizes fast-moving developments documented across academic papers, industry reports, open-source projects, and contemporary news coverage. Because the overwhelming majority of these sources are web-based (company blogs, news outlets, preprints, and code repositories) rather than archival academic publications, they are listed below with direct links rather than as formal bibliography entries. Readers are encouraged to follow the links and form their own views; these notes are intended to provoke discussion, not to provide definitive assessments.

### 9.14.1 Academic Papers and Research Reports

- Thompson, D. M., Wu, J. A., Yoder, J., & Hall, A. B. (2020). Universal vote-by-mail has no impact on partisan turnout or vote share. *Proceedings of the National Academy of Sciences*, 117(25), 14052-14056. Link.

- Brynjolfsson, E., Chandar, B., & Chen, R. (2025). Canaries in the coal mine: AI and the labor market. *Stanford Digital Economy Lab Working Paper*. Link.

- Straus, G. & Hall, A. B. (2026). How accurately did Claude Code replicate and extend a published political science paper? Link.

- Luong, T. et al. (2026). Google Aletheia: Autonomous mathematical proof verification. *arXiv* 2602.21201. Link.

- LLM Agent for Stylized Modeling (2025). *Nexus (Cell Press)*. Link.
- Anthropic (2026). AI assistance and coding skills. *Anthropic Research*. Link.
- Yang, L. (2026). Scaling reproducibility with AI agents. Link.
- Magesh, V. et al. (2025). Hallucination-free? Assessing the reliability of leading AI legal research tools. *Journal of Empirical Legal Studies*. Link.
- Barreto, K. et al. (2026). A formal proof of Erdős Problem #728. *arXiv* 2601.07421. Link.
- Xu, Y. & Yang, L. Y. (2026). Scaling reproducibility: an AI-assisted workflow for large-scale reanalysis. *arXiv* 2602.16733. Link.
- Abouzaid, M., Blumberg, A. J., Hairer, M., Kileel, J., Kolda, T. G., Nelson, P. D., Spielman, D., Srivastava, N., Ward, R., Weinberger, S., & Williams, L. (2026). First Proof: benchmarking AI on unseen mathematical problems. *arXiv* 2602.05192. Link.
- DeepSeek (2025). DeepSeek-Math-V2: first open-source model to achieve IMO gold-medal level. HuggingFace.
- Massenkoff, M. & McCrory, P. (2026). Labor market impacts of AI: a new measure and early evidence. *Anthropic Research*. Link.
- Horton, J. J., Fradkin, A., Shahidi, P., Rusak, G., & Manning, B. (2025). The Coasean Singularity? Demand, supply, and market design with AI agents. *NBER Economics of Transformative AI*. Link.
- Rao, V. S., Kumar, A., Lakkaraju, H., & Shah, N. B. (2025). Detecting LLM-generated peer reviews. *PLOS ONE*. Link.
- Lu, C., Lu, C., Lange, R. T., et al. (2026). Towards end-to-end automation of AI research. *Nature*, 651, 914-919. Link.
- Bai, J. et al. (2026). Firm data on the adoption and impact of AI. *NBER Working Paper* w34836. Link.
- METR (2025). Early-2025 AI-experienced OS dev study. Link.
- Gao, J. (2024). RaBitQ: quantizing high-dimensional vectors with a theoretical error bound for approximate nearest-neighbor search. *SIGMOD 2024*. OpenReview.

- Cheng, M., Lee, C., Khadpe, P., Yu, S., Han, D., & Jurafsky, D. (2026). Sycophantic AI decreases prosocial intentions and promotes dependence. *Science*. Link.
- Anthropic (2026). Emotion concepts and their function in a large language model. *Anthropic Research*. Link — arXiv:2604.07729 — Transformer Circuits.
- Qin, T. & Xu, Y. (2026). StatsClaw: an AI-collaborative workflow for statistical software development. *arXiv* 2604.04871. Link.
- RUC-GSAI (2026). LLM agents as social scientists: a human-AI collaborative platform for social science automation. *arXiv* 2604.01520. Link.
- Stanford HAI (2026). 2026 AI Index Report. Link — Full Report PDF.


### 9.14.2 Industry Reports and Blog Posts


- Karpathy, A. (2025). Year in review 2025. *Bear Blog*. Link.
- Epoch AI (2026). Can AI companies become profitable? *Epoch AI Substack*. Link.
- Epoch AI (2026). Open-weights vs. closed-weights models. *Data Insights*. Link.
- Yegge, S. (2026). The Anthropic Hive Mind. *Medium*. Link.
- Notion (2026). Steam, steel, and infinite minds: AI and organizational change. *Notion Blog*. Link.
- Grigorik, I. (2026). Building a Universal Commerce Protocol (UCP). *LinkedIn*. Link.
- Sant'Anna, P. (2026). Claude Code: my workflow. Link.
- Jordan, K. (2024). Muon: MomentUm Orthogonalized by Newton-Schulz. *Blog post*. Link.
- Thomson Reuters (2026). GenAI hallucinations in legal practice. Link.
- Citrini Research (2026). The 2028 Global Intelligence Crisis. Link.
- Nvidia (2026). The AI 5-layer cake. *Nvidia Blog*. Link.
- Dataconomy (2026). Chinese AI models hit 61% market share on OpenRouter. Link.
- OpenRouter (2026). State of AI 2025: 100T-token LLM usage study. Link.
- ICML (2026). On violations of LLM review policies. *ICML Blog*. Link.

- Analemma AI (2026). Introducing FARS: Fully Automated Research System. Link.
- Tu, R. (2026). 2026 and beyond. *Blog post*. Link.
- Anthropic (2026). Introducing Anthropic Science. Link.
- Anthropic (2026). Vibe physics: using Claude as a research collaborator. Link.
- Anthropic (2026). Long-running Claude: multi-day autonomous workflows. Link.
- Google Research (2026). TurboQuant: redefining AI efficiency with extreme compression. Link.
- Flynn, J. (2026). The agentic economy. X/Twitter.
- Gao, J. (2026). Response to TurboQuant. X/Twitter.
- Karpathy, A. (2026). LLM Knowledge Base. *GitHub Gist*. Link.
- Andreessen Horowitz (2026). Institutional AI vs. individual AI. *a16z News*. Link.
- MIT AI Risk (2026). Mapping the AI governance landscape. Link.
- Steinbaum, M. (2026). On em dashes and AI writing. X/Twitter.

### 9.14.3 Open-Source Projects and Datasets

- Eigent AI (2026). Eigent: the open-source cowork desktop. GitHub.
- Hall, A. B. (2026). VBM replication extension (PNAS paper replication). GitHub.
- AxiomMath (2026). Putnam 2025 AI solutions. GitHub.
- Tao, T. (2026). AI contributions to Erdős problems. Wiki.
- Social Catalyst Lab (2026). Automating Policy Evaluation (APE). Website — GitHub.
- First Proof (2026). Benchmarking AI mathematical reasoning. Website.
- Moltbook Observe (2026). Research observatory for agent social networks. Website.
- TrustAIRLab (2026). Moltbook dataset. HuggingFace.
- Holtz, D. (2026). Moltbook scraper. GitHub.

- Jordan, K. (2024). Muon optimizer. GitHub.
- Karpathy, A. (2026). Autoresearch. GitHub.
- Yang, Z. (2026). Continually Self-Improving AI. Stanford Statistics dissertation defense slides (March 3, 2026). Slides — YouTube.
- Charlotin, D. (2026). AI hallucination cases in law database. Website.
- Sant'Anna, P. (2026). Claude Code agentic workflow. GitHub.
- OpenAI (2026). First Proof submissions. Link.
- Kolda, T. G. (2026). First Proof: AI's toughest math test. *MathSci.ai Blog*. Link.
- Sharma, M. (2026). Resignation letter from Anthropic. X/Twitter.
- Tu, R. (richards199999) (2026). Thinking-Claude. GitHub.
- Guo, H. (666ghj) (2026). MiroFish: multi-agent swarm-intelligence prediction engine. GitHub.
- Sakana AI (2026). AI Scientist v2. GitHub.
- Tan, G. (2026). GStack: founder's agent framework for Claude Code. GitHub.
- StatsClaw (2026). AI workflow for statistical package development. Website — GitHub.
- RUC-GSAI (2026). YuLan-OneSim: social science simulation automation. GitHub.


### 9.14.4 News Coverage


- Fortune (2026). Cursor built web browser with swarm AI agents powered by OpenAI. Link.
- CNBC (2026). Chinese tech giants enter the 'agentic commerce' race as AI reshapes super apps. Link.
- Alibaba Group (2026). Alibaba's Qwen App advances agentic AI strategy. Link.
- OpenAI (2026). Testing ads in ChatGPT. Link.
- MAA (2025). Results of the 86th William Lowell Putnam Mathematical Competition. Link.

- TechStartups (2025). Stanford math PhD student's AI startup, Axiom, is raising $50M at $300 million valuation. Link.
- CNBC (2026). MiniMax doubles in Hong Kong debut, marking yet another Chinese AI listing. Link.
- Global Times (2026). AI 'tiger' Zhipu debuts in HK, closing 13% higher than IPO price. Link.
- Nature (2025). The Chinese finance whizz whose DeepSeek AI model stunned the world (Nature's 10). Link.
- SCMP (2025). DeepSeek releases first open AI model with gold-level scores at maths olympiad. Link.
- CNBC (2025). Meta acquires intelligent agent firm Manus. Link.
- TechCrunch (2026). Meta acquired Moltbook, the AI agent social network. Link.
- SCMP (2026). MiniMax and Zhipu's stellar Hong Kong IPOs supercharge China's AI ambitions. Link.
- Yahoo Tech (2026). Insiders: DeepSeek V4 beat Claude. Link.
- Forbes (2025). Klarna, UPS, Duolingo, Cisco, and many other companies are replacing workers with AI. Link.
- Wired (2026). AI agents are hiring humans. Link.
- BBC (2026). AI and military applications. Link.
- FT (2026). Marrying AI with space engineering. Link.
- Guancha (2026). Zhang Wenhong opposes AI in hospital diagnostics. Link.
- Guancha (2026). Wang Xiaochuan responds to Zhang Wenhong on medical AI. Link.
- China Daily (2026). Shanghai doctor sparks AI in medicine debate. Link.
- GeekPark (2026). Wang Xiaochuan on AI healthcare. Link.
- Stanford HAI (2025). AI on trial: legal models hallucinate in 1 out of 6 or more queries. Link.
- CNBC (2026). Musk's xAI-SpaceX combo is the biggest merger of all time. Link.
- TechCrunch (2026). SpaceX officially acquires xAI. Link.

- SpaceNews (2026). SpaceX acquires xAI in bid to develop orbital data centers. Link.
- Scientific American (2026). First Proof is AI's toughest math test yet. The results are mixed. Link.
- Yahoo Finance (2026). Anthropic's AI safety head just resigned. Link.
- Futurism (2026). Anthropic researcher quits in cryptic public letter. Link.
- CNBC (2026). Block laying off about 4,000 employees, nearly half of its workforce. Link.
- CNN (2026). Block lays off nearly half its staff because of AI. Link.
- Bloomberg (2026). Jack Dorsey's 4,000 job cuts at Block arouse suspicions of AI-washing. Link.
- CNN (2026). Anthropic rejects latest Pentagon offer. Link.
- CNBC (2026). Anthropic was the Pentagon's choice for AI. Now it's banned. Link.
- CNBC (2026). China's tech firms feast on OpenClaw AI agent adoption. Link.
- MIT Technology Review (2026). Hustlers are cashing in on China's OpenClaw AI craze. Link.
- Bloomberg (2026). China moves to limit use of OpenClaw AI at banks, government agencies. Link.
- CNBC (2026). Anthropic's Claude AI and the Pentagon supply-chain dispute. Link.
- CNBC (2026). Karp, Palantir weigh in on Anthropic-Pentagon blacklist. Link.
- New Yorker (2026). The Pentagon went to war with Anthropic. What's really at stake. Link.
- TechCrunch (2026). Lawyer behind AI psychosis cases warns of mass-casualty risks. Link.
- Guardian (2026). Tumbler Ridge shooting victim sues OpenAI. Link.
- China Daily (2026). AI manipulation of search results highlighted at consumer gala. Link.

- Yicai Global (2026). China's annual CCTV consumer-rights show uncovers AI ad tricks. Link.
- CNBC (2026). Iran war and data centers. Link.
- Fortune (2026). Iran war, helium shortage, and chip supply chains. Link.
- Bloomberg (2026). How Amazon data centers became a casualty of the Iran war. Link.
- WSJ (2026). AI startup Aaru's young founders reach $1B valuation. Link.
- TechCrunch (2026). AMI Labs raises $1.03 billion to build world models. Link.
- NYT (2026). OpenAI is shutting down Sora. Link.
- CNBC (2026). ChatGPT ads testing expands. Link.
- Business Insider (2026). Meta's Reality Labs shifts to AI-native pods. Link.
- 36Kr (2026). TurboQuant and the memory stock selloff. Link.
- NYT (2026). Anthropic Claude Mythos preview sparks banking concerns. Link.
- CNN (2026). Suspect in attack at Sam Altman's house charged with attempted murder. Link.
- CNBC (2026). Suspect in attack at Sam Altman's house aimed to kill OpenAI CEO. Link.
- CNBC (2026). Anthropic limits Claude Mythos rollout over cybersecurity fears. Link.

# References

An, C. et al. (2023). Large-scale pancreatic cancer detection via non-contrast CT and deep learning. *Nature Medicine*, 29:1206–1215.

Aramayo, N., Schiappacasse, M., and Goic, M. (2022). A multiarmed bandit approach for house ads recommendations. *Marketing Science*, 41(4):679–700.

Athey, S. and Imbens, G. (2016). Recursive partitioning for heterogeneous causal effects. *Proceedings of the National Academy of Sciences*, 113(27):7353–7360.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256.

Bai, J. et al. (2026). Firm data on the adoption and impact of AI. NBER Working Paper w34836, National Bureau of Economic Research. Available at https://www.nber.org/papers/w34836.

Bajari, P., Nekipelov, D., Ryan, S. P., and Yang, M. (2015). Machine learning methods for demand estimation. *American Economic Review*, 105(5):481–485.

Belkin, M., Hsu, D., Ma, S., and Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias-variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press.

Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming*. Athena Scientific.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Chang, M., Obi, E., and Yoganarasimhan, H. (2025). Balancing engagement and polarization: Multi-objective alignment of news content using LLMs. *arXiv preprint arXiv:2504.13444*.

Chen, L., Pelger, M., and Zhu, J. (2024). Deep learning in asset pricing. *Management Science*, 70(2).

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. (2021). Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Cheng, M., Lee, C., Khadpe, P., Yu, S., Han, D., and Jurafsky, D. (2026). Sycophantic AI decreases prosocial intentions and promotes dependence. *Science*.

Cohen, M. C., Zhang, R., and Jiao, K. (2022). Data aggregation and demand prediction. *Operations Research*, 70(5):2635–2657.

Covington, P., Adams, J., and Sargin, E. (2016). Deep neural networks for YouTube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 191–198.

Cui, R., Gallino, S., Moreno, A., and Zhang, D. J. (2018). The operational value of social media information. *Production and Operations Management*, 27(7):1749–1769.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314.

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and Ré, C. (2022). FlashAttention: Fast and memory-efficient exact attention with IO-awareness. *arXiv preprint arXiv:2205.14135*.

Davies, A., Veličković, P., Buesing, L., Blackwell, S., Zheng, D., Tomašev, N., Tanburn, R., Battaglia, P., Blundell, C., Juhász, A., Lackenby, M., Williamson, G., Hassabis, D., and Kohli, P. (2021). Advancing mathematics by guiding human intuition with AI. *Nature*, 600(7887):70–74.

DeepSeek-AI (2024). DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*.

DeepSeek-AI (2025). DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638.

Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., and Madry, A. (2020). Implementation matters in deep policy gradients: A case study on PPO and TRPO. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Farias, V. F. and Li, A. A. (2019). Learning preferences with side information. *Management Science*, 65(8):3731–3749.

Gijsbrechts, J., Boute, R. N., Van Mieghem, J. A., and Zhang, D. J. (2022). Can deep reinforcement learning improve inventory management? performance on lost sales, dual-sourcing, and multi-echelon problems. *Manufacturing & Service Operations Management*, 24(3):1664–1677.

Gu, S., Kelly, B., and Xiu, D. (2020). Empirical asset pricing via machine learning. *The Review of Financial Studies*, 33(5):2223–2273.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning*. Springer.

Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. (2018). Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Hofman, J. M., Watts, D. J., Athey, S., Garip, F., Griffiths, T. L., Kleinberg, J., Knox, H., Ludwig, J., Mullainathan, S., Salganik, M. J., Vazire, S., Vespignani, A., and Yarkoni, T. (2021). Integrating explanation and prediction in computational social science. *Nature*, 595(7866):181–188.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Hsu, C. C.-Y., Mendler-Dünner, C., and Hardt, M. (2020). Revisiting design choices in proximal policy optimization. *arXiv preprint arXiv:2009.10897*.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An Introduction to Statistical Learning*. Springer.

Jean, N., Burke, M., Xie, M., Davis, W. M., Lobell, D. B., and Ermon, S. (2016). Combining satellite imagery and machine learning to predict poverty. *Science*, 353(6301):790–794.

Kaji, T., Manresa, E., and Pouliot, G. (2023). An adversarial approach to structural estimation. *Econometrica*, 91(6):2041–2063.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kleinberg, J., Lakkaraju, H., Leskovec, J., Ludwig, J., and Mullainathan, S. (2018). Human decisions and machine predictions. *The Quarterly Journal of Economics*, 133(1):237–293.

Kleinberg, J., Ludwig, J., Mullainathan, S., and Obermeyer, Z. (2015). Prediction policy problems. *American Economic Review*, 105(5):491–495.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. (2022). Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35.

Kokkodis, M. and Ipeirotis, P. G. (2021). Demand-aware career path recommendations: A reinforcement learning approach. *Management Science*, 67(7):4030–4050.

Konda, V. R. and Tsitsiklis, J. N. (2000). Actor-critic algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 12.

Kong, L., Jin, J., and Zhang, R. (2026). Improving behavioral alignment in LLM social simulations via context formation and navigation. Working paper. Working paper.

Lai, T. L. and Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22.

Lambrecht, A. and Tucker, C. (2019). Algorithmic bias? an empirical study of apparent gender-based discrimination in the display of STEM career ads. *Management Science*, 65(7):2966–2981.

Lee, H.-y. (2026). The agent era. Course material, National Taiwan University. Available at https://speech.ee.ntu.edu.tw/~hylee/ml/ml2026-course-data/agent_era.pdf.

Li, Z., Liu, M., Chen, D., Lyu, M., Wang, S., and Zheng, Z. (2024). Beyond one-preference-fits-all alignment: Multi-objective direct preference optimization. In *Findings of the Association for Computational Linguistics (ACL)*.

Liu, X. (2023). Dynamic coupon targeting using batch deep reinforcement learning: An application to livestream shopping. *Marketing Science*, 42(4):610–636.

Misra, K., Schwartz, E. M., and Abernethy, J. (2019). Dynamic online pricing with incomplete information using multiarmed bandit experiments. *Marketing Science*, 38(2):226–252.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Noy, S. and Zhang, W. (2023). Experimental evidence on the productivity effects of generative artificial intelligence. *Science*, 381(6654):187–192.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. (2022). Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35.

Peng, J. and Liang, C. (2021). On the differences between view-based and purchase-based recommender systems. *MIS Quarterly*, 45(2):939–956.

Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., and Finn, C. (2023). Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36.

Rajasekaran, P., Dixon, E., Ryan, C., and Hadfield, J. (2025). Effective context engineering for AI agents. Anthropic Engineering Blog. Available at https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents.

Rummery, G. A. and Niranjan, M. (1994). On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Cambridge University Engineering Department.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2016). Prioritized experience replay. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Schwartz, E. M., Bradlow, E. T., and Fader, P. S. (2017). Customer acquisition via display advertising using multi-armed bandit experiments. *Marketing Science*, 36(4):500–522.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y. K., Wu, Y., and Guo, D. (2024). DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359.

Singh, S. P. and Sutton, R. S. (1996). Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22(1-3):123–158.

Song, Y. and Sun, T. (2024). Ensemble experiments to optimize interventions along the customer journey: A reinforcement learning approach. *Management Science*, 70(8):5117–5139.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D. M., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. (2020). Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33.

Su, C.-L. and Judd, K. L. (2012). Constrained optimization approaches to estimation of structural models. *Econometrica*, 80(5):2213–2230.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44.

Sutton, R. S. (2019). The Bitter Lesson. Incomplete Ideas (blog). Available at http://www.incompleteideas.net/IncIdeas/BitterLesson.html.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 12.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.

Tibshirani, R. (2016). Gradient descent. Lecture notes, Convex Optimization (10-725), Fall 2016, Carnegie Mellon University. Available at https://www.stat.cmu.edu/~ryantibs/convexopt-F16/lectures/grad-descent.pdf.

Tsitsiklis, J. N. and Van Roy, B. (1997). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690.

van Binsbergen, J. H., Bryzgalova, S., Mukhopadhyay, M., and Sharma, V. (2023). (almost) 200 years of news-based economic sentiment. SSRN Working Paper. Available at http://dx.doi.org/10.2139/ssrn.4398326.

Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double Q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30.

Wager, S. and Athey, S. (2018). Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523):1228–1242.

Wang, W., Li, B., Luo, X., and Wang, X. (2023a). Deep reinforcement learning for sequential targeting. *Management Science*, 69(9):5382–5404.

Wang, Z., Gao, R., and Li, S. (2023b). Neural-network mixed logit choice model: Statistical and optimality guarantees. Working paper. Working paper.

Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., and de Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3-4):279–292.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35.

Wei, Y. and Jiang, Z. (2024). Estimating parameters of structural models using neural networks. *Marketing Science*.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256.

Xu, S., Fu, W., Gao, J., Ye, W., Liu, W., Mei, Z., Wang, G., Yu, C., and Wu, Y. (2024). Is DPO superior to PPO for LLM alignment? a comprehensive study. *arXiv preprint arXiv:2404.10719*.

Ye, Z., Zhang, D. J., Zhang, H., Zhang, R., Chen, X., and Xu, Z. (2023). Cold start to improve market thickness on online advertising platforms: Data-driven algorithms and field experiments. *Management Science*, 69(7):3838–3860.

Ye, Z., Zhang, Z., Zhang, D. J., Zhang, H., and Zhang, R. (2024). Deep learning-based causal inference for large-scale combinatorial experiments: Theory and empirical evidence. *Management Science*.

Zhan, R., Pei, C., Su, Q., Wen, J., Wang, X., Mu, G., Zheng, D., Jiang, P., and Gai, K. (2022). Deconfounding duration bias in watch-time prediction for video recommendation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2472–2481.

Zheng, R., Dou, S., Gao, S., Hua, Y., Shen, W., Wang, B., Liu, Y., Jin, S., Liu, Q., Zhou, Y., Xiong, L., Chen, L., Xi, Z., Xu, N., Lai, W., Zhu, M., Chang, C., Yin, Z., Weng, R., Cheng, W., Huang, H., Sun, T., Yan, H., Gui, T., Zhang, Q., Qiu, X., and Huang, X. (2023). Secrets of RLHF in large language models (part I): PPO. *arXiv preprint arXiv:2307.04964*.

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P., and Irving, G. (2019). Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.