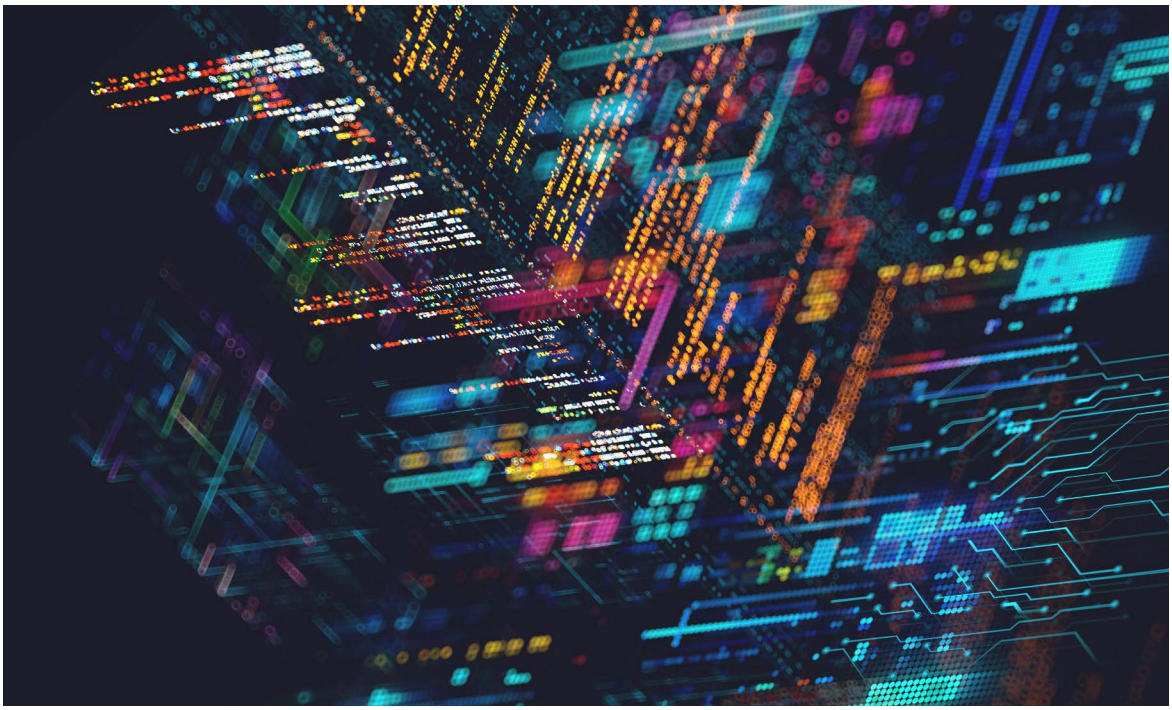
RAND

Research Report

CARTER C. PRICE, BRIEN ALKIRE, MOHAMMAD AHMADI

Algorithmic Advancement in Artificial Intelligence

A Survey of Advances with Projections for the Near Future



For more information on this publication, visit **www.rand.org/t/RRA3485-1**.

Published by the RAND Corporation, Santa Monica, Calif.

Cover: MF3d/Getty Images.

About This Report

With recent advancements in commercial products, such as OpenAI's ChatGPT, Anthropic AI's Claude, Meta's Llama, and other large language models, the topic of artificial intelligence (AI) has expanded in the public discourse. And, as AI capabilities develop, there has been increasing concern about their security implications. In this report, we survey algorithmic improvements from numerical analysis, operations research, and computer science; identify some common channels of advancement; and then describe the channels by which AI might advance. We also describe the implications that algorithmic improvement may have on AI advancement over the next few years and discuss some indicators that might point to such advancements. The purpose of this research is to present issues to consider regarding the future algorithmic advancement.

This work is intended to be of interest to both policymakers and a more general audience looking for information about algorithmic advancement in AI. However, portions of this report assume that the reader has familiarity with algorithms in general and machine learning algorithms in particular, and some of the content in the appendixes relies on an understanding of advanced mathematics, including numerical analysis.

The research in this report was conducted between October 2023 and August 2024. This predates the unveiling of DeepSeek-V3 in late December 2024.[1] DeepSeek-V3 purportedly outperforms similar open-source language models and performs comparably to leading closed-source models while requiring less compute for full training; it may provide an important example of an algorithmic advancement.[2] However, the authors made minor revisions and updates to the report through February 12, 2025.

Technology and Security Policy Center

RAND Global and Emerging Risks is a division of RAND that delivers rigorous and objective public policy research on the most consequential challenges to civilization and global security. This work was undertaken by the division's Technology and Security Policy Center, which explores how high-consequence, dual-use technologies change the global competition and threat environment, then develops policy and technology options to advance the security of the

[1] Cade Metz and Meaghan Tobin, "How Chinese A.I. Start-Up DeepSeek Is Competing with Silicon Valley Giants," *New York Times*, January 23, 2025.

[2] DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, et al., "DeepSeek-V3 Technical Report," arXiv, version 1, December 27, 2024.

United States, its allies and partners, and the world. For more information, contact tasp@rand.org.

Funding

This research was independently initiated and conducted within the Technology and Security Policy Center using income from operations and gifts from philanthropic supporters, which have been made or recommended by DALHAP Investments Ltd., Effektiv Spenden, Ergo Impact, Founders Pledge, Charlottes och Fredriks Stiftelse, Good Ventures, Jaan Tallinn, Longview, Open Philanthropy, and Waking Up Foundation. A complete list of donors and funders is available at www.rand.org/TASP. RAND donors and grantors have no influence over research findings or recommendations.

Acknowledgments

We thank the reviewers, Mary Lee and Neil Thompson, for their thoughtful reviews and constructive comments. We also appreciate the guidance of Jeff Alstott, Emma Westerman, and Casey Dugan and the support provided from the Technology and Security Policy Center throughout this study. We also thank Lennart Heim, Konstantin Pitz, Mauricio, Gabriel Kulp, and the participants in the Compute Working Group for their comments on an earlier draft of this work, and Nick Brown, who also provided substantive comments, particularly during the initial phases of this work. Finally, we thank Alison Hottes and Bryan Boling for their work managing the quality assurance process of this document. While we have benefited from the insights from many people, any errors in this document are solely the responsibility of the authors.

Summary

With recent advancements in commercial products, such as OpenAI's ChatGPT, Anthropic AI's Claude, Meta's Llama, and other large language models, the topic of artificial intelligence (AI) has expanded in the public discourse. And, as AI capabilities develop, there has been increasing concern about their security implications. In this report, we make evidence-based projections about the direction and pace of algorithmic advancements to help inform policymaking. We describe several possible channels for algorithmic improvement related to AI and explore the implications of how progress might be made along each of those channels.[3]

Key Findings

Our research on the direction and pace of algorithmic advancements revealed the following key findings:

- The two potentially high-impact channels for algorithmic improvement involve (1) generating synthetic data or pruning existing data to produce datasets better suited for training AI and (2) increasing data efficiency through improved algorithms that are either less computationally costly than transformers (such as Mamba) or more effective per iteration than transformers (such as Kolmogorov-Arnold Networks).[4] There is also potential for both improvements to happen more or less simultaneously.
- One wild-card channel would be the development of alternative criteria (which we loosely refer to in this report as *objective functions*) for training AI systems that better match commercially useful performance measures. [5]
- There are three near-term futures that depend on different levels of advancement along the two high-impact channels.
  - **If data limitations are binding:** A future scenario is possible in which the unavailability of additional data could prevent models from continuing to scale efficiently, and that could lead to small, focused AI systems dominating the market.
  - **If algorithms fail to scale:** In a future in which additional data can be obtained through synthetic generation (or some other mechanism)[6] but new algorithms are not able to efficiently extract meaningful performance gains by including those additional data, then work on large models could continue, but small AI systems would likely

[3] For the purposes of this report, changes to an algorithm are an *improvement* if they lead to enhanced performance measures or reduced effort and associated resource requirements (or both) for a given task.

[4] We make no claim that algorithmic improvements will or will not be widely adopted for commercial applications.

[5] Many models use the cross-entropy loss function as the primary objective for training. Some models pair that with reinforcement learning or reinforcement learning through human feedback to improve performance. Alternatives to these objectives could lead to substantial improvements.

[6] For instance, training on nontext modalities.

dominate.[7] Essentially, if there are diminishing returns to scale with additional data, then larger models might not be commercially viable.
  - **If algorithms continue to advance:** In a future in which data are abundant and algorithms are more efficient in using those data, then ever-larger models are likely to be a significant factor in AI research for the near term.

- One implication of algorithmic advancement is that export controls on hardware—such as restrictions on the export of high-end chips to China that were made in October 2022, October 2023, December 2024, and January of 2025[8]—could have muted effects, depending on the path of algorithmic advancement. As described in a 2024 Center for a New American Security report,[9] if algorithmic improvements continue to be widely available, then hardware-restricted actors (such as China) will be able to train models and be only a few upgrade cycles behind the frontier.

[7] For a detailed discussion of which domains might benefit from use of synthetic data, see Pablo Villalobos, Anson Ho, Jaime Sevilla, Tamay Besiroglu, Lennard Heim, and Marius Hobbhahn, "Will We Run Out of Data? Limits of LLM Scaling Based On Human-Generated Data," arXiv, version 2, June 4, 2024, pp. 7–9.

[8] Bureau of Industry and Security, "Commerce Strengthens Restrictions on Advanced Computing Semiconductors to Enhance Foundry Due Diligence and Prevent Diversion to PRC," Office of Congressional Affairs, January 15, 2025.

[9] Paul Scharre, "Future-Proofing Frontier AI Regulation," Center for a New American Security, March 13, 2024.

Contents

Figures

Figures

Chapter 1. Introduction

With recent advancements in commercial products—such as OpenAI's ChatGPT (which was released in 2018), [10] Anthropic AI's Claude, Meta's Llama, and other large language models (LLMs)—the topic of artificial intelligence (AI) has expanded in the public discourse. And, as AI capabilities develop, there has been increasing concern about their security implications. To assess the security implications associated with AI, policymakers need to have estimates of the direction and pace of algorithmic advancements. To that end, we seek to address the question: How will AI capabilities advance in the near future because of algorithms?

What Constitutes Algorithmic Improvement?

There are many ways to define what constitutes an algorithmic improvement and what distinguishes an improvement from a new algorithm, but none of the options are particularly robust. A study by Yash Sherry and Niel C. Thompson focuses on algorithms for problems with exact solutions that are globally optimal and defines an improvement in terms of solving the same problem with fewer operations.[11] Alternatively, research by Katja Grace takes a very different approach and evaluates a variety of algorithms, including machine learning algorithms, using a variety of performance measures. These measures include the Elo rating system,[12] the time required for problems of a given complexity, the size of problems that can be solved, and sample statistics, such as probabilities of detection.[13] The report by Grace focused on the empirical performance of algorithms, including both hardware and software advances, while the focus of our work in this report is on algorithmic improvements in the absence of hardware improvements. Hence, we are keenly interested in algorithmic performance on specific tasks relative to the effort and associated resources required.

For the purposes of this report, changes to an algorithm are an *improvement* if, for a given task, they lead to (1) enhanced performance measures or (2) reduced effort and associated resource requirements (or both). In different cases, the improvements could be more subjective (e.g., sample statistics on human preferences) or more objective (e.g., a reduction in the number

[10] Anam Nazir and Ze Wang, "A Comprehensive Survey of ChatGPT: Advancements, Applications, Prospects, and Challenges," *Meta-Radiology*, Vol. 1, No. 2, 2023.

[11] Yash Sherry and Neil C. Thompson, "How Fast Do Algorithms Improve?" *Proceedings of the IEEE*, Vol. 109, No. 11, November 1, 2021.

[12] An *Elo rating system* is a method of calculating the relative skill level of players in zero-sum games, such as chess, baseball, and pocket billiards. This rating system has more recently been applied to LLMs.

[13] Katja Grace, "Algorithmic Progress in Six Domains," Machine Intelligence Research Institute, Technical Report 2013-3, December 9, 2013.

of floating point operations [FLOPs][14] needed to perform a mathematical operation). The judgment of the authors will be used to identify what constitutes a task.

Dimensions of Improvement

There are several ways to describe algorithmic improvement in AI. One way to frame improvement would be with regard to the extensive or intensive margin. The intensive margin would include such things as reduced requirements for inputs (e.g., training data, training FLOPs,[15] or model parameters) or better performance with the same or fewer inputs. Essentially, the intensive margin is about efficiency. Improvements along the extensive margin would include new capabilities or areas of application—for example, the ability to solve a new class of problem that prior models were not able to solve.

Improvements can occur at different periods: during the training phase, while making post-training adjustments, or during inference.[16] Our focus in this report is on the intensive margin during the training phase. Our rationale is that training requires upfront costs that could be a barrier to the development of future models, and advancements along the extensive margin are generally harder to quantify. That said, some algorithmic changes might result in improvements along multiple dimensions or offer improvements along one dimension at the expense of another.

Approach and Limitations

To estimate the pace of algorithmic advancement, we first looked at a variety of algorithms in numerical analysis, operations research, and computer science to find the mechanisms for algorithmic advancement. We then grouped these mechanisms into broad classes and searched the computer science literature for discussions about their applicability to LLMs. Finally, we describe how these mechanisms could work in the near future to improve the algorithms behind LLMs and other foundation models.

Our approach was not comprehensive, and the algorithms that we assessed were selected by examining several textbooks. Thus, we might have missed some relevant mechanisms. Additionally, because of the rapid pace at which new research papers are published, our examination of the application of these mechanisms is necessarily incomplete. While this study

---

[14] In this report, *FLOPs* is the plural form of the abbreviation *FLOP*, which refers to one operation (e.g., addition, multiplication) performed on decimal (or floating point) numbers. *FLOPs per second (FLOP/s)* refers to the number of FLOPs that a processor can perform in one second. See Lennart Heim, "FLOP for Quantity, FLOP/s for Performance," blog, *.xyz, April 14, 2023, and Appendix A for a more detailed discussion of FLOPs and FLOP/s.

[15] See Appendix A.

[16] *Inference* refers to the post-training period when an AI model is introduced to new data and assessed on its ability to recognize patterns in and make inferences about the new dataset. See Appendix A for a more detailed discussion of different types of training and inference.

cannot be considered exhaustive, we do believe that the approach is sufficient to identify broad trends and make projections useful for exploring policy options.

As discussed in the preface, the research in this report was conducted between October 2023 and August 2024. An important limitation of this report is that the research was conducted before DeepSeek unveiled its DeepSeek-V3 language model in December 2024, which appears to be an important example of algorithmic improvement.[17] According to DeepSeek, their model "outperforms other open-source models and achieves performance comparable to leading closed models. . . . [And it] requires only 2.788M H800 GPU hours for its full training."[18] DeepSeek-V3 is described as a mixture-of-experts (MoE) language model that achieves efficient inference and cost-effective training by adopting multi-head latent attention and architectural changes to their previous model, implementing a new strategy for load balancing, and performing a multi-token prediction training objective for stronger performance. Model training was followed by supervised fine-tuning (SFT) and reinforcement learning stages to align its performance with human preferences.[19]

This report discusses similar mechanisms of algorithmic improvement but is not informed by the specific details of DeepSeek-V3. For instance, a consideration of DeepSeek-V3 was not a part of our assessment in Appendix D of the utility of reinforcement learning from human feedback (RLHF) in advancing AI algorithms. Details about the role of reinforcement learning with DeepSeek-V3 are described in a technical report published in January 2025.[20]

Organization of This Report

Chapter 2 describes the relevant literature on algorithmic advancement related to AI. Then, Chapter 3 presents the mechanisms that we have identified for algorithmic advancement and discusses how they might apply to AI systems. The final chapter describes how AI algorithms might advance in the near future and the implications these advancements could have.

We also include four appendixes: Appendix A provides background information on the computational effort associated with machine learning algorithms, which is intended to provide useful context for the interested reader; Appendix B includes details about how we identified the mechanisms of algorithmic improvements; Appendix C describes the specific implications of

---

[17] Cade Metz and Meaghan Tobin, "How Chinese A.I. Start-Up DeepSeek Is Competing with Silicon Valley Giants," *New York Times*, January 23, 2025.

[18] DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, et al., "DeepSeek-V3 Technical Report," arXiv, version 1, December 27, 2024.

[19] DeepSeek-AI et al., 2024.

[20] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, et al., "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning" arXiv, version 1, January 22, 2025.

algorithmic advancement on export control policies; and Appendix D contains a case study related to RLHF.

Chapter 2. Literature on Algorithmic Advancement

We are not the first to investigate algorithmic improvements relevant to AI. In this chapter, we explore existing literature on algorithmic improvement, especially studies that are relevant to the training phase of an AI system's development.

The report by Grace mentioned in Chapter 1 examined a handful of problem types on which there has been algorithmic progress, including Boolean satisfiability, chess and Go, large number factorization, physics simulations, mixed integer programming, scheduling, and a variety of machine learning problems. For each of these problems, Grace found literature summarizing performance progress and assessed the share of the progress that was attributable to algorithmic advancement. Using these examples, she determined that algorithmic advancement accounted for 50 to 100 percent of improved performance.[21]

Sherry and Thompson measured the pace of algorithmic innovation for 128 families of exact algorithms and 310 algorithmic improvements. With exact algorithms, the result of a specific problem solved by different algorithms within each family will be identical, so an improvement would be in the arithmetic operation count that an algorithm requires to reach the exact solution. Sherry and Thompson found that the pace and scale of improvement varied substantially; some algorithm families saw no substantive improvements, and others saw improvements that were substantially faster than the hardware advancement pace described in Moore's Law.[22] While their study provides an empirical assessment of algorithmic advancement, it does not provide a forecast that is relevant to the pace of advancement in AI.[23]

Using published characteristics of models from 2012 to 2023 and applying cross-entropy loss function to measure performance, Ho et al. estimated that 5 to 40 percent of LLM performance increases following pretraining were attributable to algorithmic improvements.[24] The paper identifies two key innovations that resulted in the majority of the performance increase: the

[21] Grace, 2013.

[22] Moore's Law is a projection, based on empirical observation, that the number of transistors per square inch on a microchip will double every two years. This increase in density relates to an increase in computing power.

[23] Yash Mohan Sherry and Neil C. Thompson, "Measuring the Pace of Innovation: Evidence From Algorithms," conference paper, SI 2020 IT and Digitization, National Bureau of Economic Research, July 2020.

[24] Anson Ho, Tamay Besiroglu, Ege Erdil, David Owen, Robi Rahman, Zifan Carl Guo, David Atkinson, Neil Thompson, and Jaime Sevilla, "Algorithmic Progress in Language Models," arXiv, version 1, March 9, 2024.

A *cross-entropy loss function* is a way to evaluate machine learning algorithms. In general, a cross-entropy loss function compares an actual data point(s) to the output from the machine learning model. In practice, these comparisons are aggregated to elicit specific behavior in a model. Essentially, these measures evaluate how well the model matches the training data.

introduction of the transformer (a deep learning architecture) and the scaling law from Hoffmann et al., 2022.[25]

In the Stanford Institute for Human-Centered AI's *2024 AI Index Report*, the authors collected information on AI advancement.[26] They note that AI performance has been approaching or surpassing human performance on nine technical performance benchmarks. However, they also note that "[p]erformance on these benchmarks has stagnated in recent years, indicating either a plateau in AI capabilities or a shift among researchers toward more complex research challenges."[27]

Leopold Aschenbrenner reviewed advancements in LLMs and projected the growth forward.[28] He estimates that there has been about half an order of magnitude of gains in model improvement per year attributable to algorithmic advancement and, if this trend continues into 2027, he predicts that AI systems will be able to do the work of AI researchers.

There is no clear consensus among these studies about the pace or direction of algorithmic advancement. Furthermore, although Aschenbrenner and the authors of the *2024 AI Index Report* discuss forward-looking paths for advancement, they have somewhat divergent interpretations of the trends. Specifically, they disagree about whether AI systems are plateauing at or near human levels of performance. Another key point of disagreement is about whether continued improvements in the performance of a cross-entropy loss function that is based on predicting the next token is sufficient to achieve material improvements in commercially relevant performance measures.[29]

We attempt to resolve these issues by approaching this problem slightly differently than earlier studies. By focusing on the mechanisms of improvement rather than the pace of

---

[25] In this context, a scaling law is an empirical relationship between the number of parameters, training computation, and model performance. Hoffman et al. trained more than "400 language models ranging from 70 million to over 16 billion parameters on 5 to 500 billion tokens" and found that, "for compute-optimal training, the model size and the number of training tokens should be scaled equally: for every doubling of model size the number of training tokens should also be doubled" (Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al., "Training Compute-Optimal Large Language Models," arXiv, version 1, March 29, 2022, p. 1). An earlier scaling law was presented in Jared Kaplan, Sam McCandish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei, "Scaling Laws for Neural Language Models," arXiv, version 1, January 23, 2020.

[26] Nestor Maslej, Loredana Fattorini, Raymond Perrault, Vanessa Parli, Anka Reuel, Erik Brynjolfsson, John Etchemendy, Katrina Ligett, Terah Lyons, James Manyika, Juan Carlos Niebles, Yoav Shoham, Russell Wald, and Jack Clark, *The AI Index 2024 Annual Report*, AI Index Steering Committee, Institute for Human-Centered AI, Stanford University, April 2024.

[27] Maslej et al., 2024, p. 82. This apparent stagnation represents, in part, a limitation of appropriate benchmarks for the directions relevant to AI advancement and a ceiling for possible performance on some of the benchmarks.

[28] Leopold Aschenbrenner, *Situational Awareness: The Decade Ahead*, June 2024.

[29] By *commercially relevant*, we mean AI systems that produce sufficient value that they are commercially viable, which is to say that the market for the output from those systems generates sufficient revenue to at least pay for the marginal cost of operating the model.

improvement, we are not treating advancements as exogenous but instead we present a summary of paths that have empirically led to algorithmic advances along the intensive margin, specify how these paths could be applied to AI systems, and then describe early indicators that could be a sign of how AI systems are likely to advance.

Chapter 3. Mechanisms for Algorithmic Advancement

Based on our review of canonical problem types in numerical analysis, operations research, and computer science described in Appendix B, we have identified the following key channels by which algorithms can improve:

- **Fewer iterations**: Reducing the number of iterations saves computational costs.
- **Stochasticity**: Injecting randomness can accelerate convergence by moving away from local optima, thereby improving performance.
- **Reducing precision**: Using fewer significant digits can reduce storage proportionally and the computational costs more than proportionally, in some contexts.[30]
- **Sparsity**: Specialized algorithms can take advantage of patterns of sparsity in data to work faster than a dense set and reduce storage costs.
- **Data tailoring**: Algorithms can be tailored to take advantage of the properties of data types.
- **Objective functions**: Alternative objective functions can allow for less computational costs or improved performance.
- **Complexity**: Alternative algorithms might trade the pace of convergence with the computational cost of each iteration.

In this chapter, we will discuss how each of these channels are or could be applied to AI and discuss the implications for the near future.

Channels Unlikely to Lead to Substantial Improvements

Reviewing the channels for improvement, we identified three that we think are unlikely to lead to significant algorithmic improvement.

Fewer Iterations

Models that use the empirically demonstrated scaling laws previously described are applying nearly the optimal amount of compute for a given corpus and parameter count, so we do not believe that a continued reduction in the number of forward and backward propagation iterations will yield significant improvements in the near term. Similarly, although data points can be fed through the model multiple times during training to improve performance, the performance effects can be tracked, so decreasing the iterations per data point is also unlikely to lead to substantial improvements.

[30] There is not an easily formulated relationship between precision and computational costs because this relationship will fundamentally be context dependent. However, Appendix B provides some empirical examples that were roughly quadratic improvements in computational costs.

However, if existing models are overfitted to their training data, then there could be some benefit to reducing the training iterations. For smaller scale machine learning problems, such techniques as k-cross fold validation are useful for reducing the risk of overfitting, but that would be very computationally expensive for something on the scale of an LLM. Other approaches to reducing the risk of overfitting, such as ensemble approaches, are being deployed in some capacity today. That said, even if overfitting is a concern, reducing the number of iterations likely would not dramatically increase performance, although it would reduce the computational costs proportional to the reduction in iterations.

Additionally, because the inference costs are proportional to the number of parameters in a model (not in the training cost of the model), reducing the number of iterations used in training would not materially affect the inference costs.

Stochasticity

Randomness (or quasirandomness) is already a factor in LLMs and other AI systems through stochastic gradient descent during pretraining, the selection of starting values for some diffusion models, and various other points in the architecture. In these cases, stochasticity is typically used to help the algorithms refrain from getting caught in a local optimum. Given that stochasticity is commonly used in many parts of AI systems already, it is not obvious from our review where additional stochasticity could lead to improvements in performance.

Reducing Precision

By reducing the number of bits to encode information, storage requirements are reduced proportionally to the degree of reduction, and the computational effort could be reduced by the square of the bit count, depending on the types of operations performed. This channel can apply to both the training and inference stages of modeling.[31] It is used in many LLMs, particularly for deployment on edge devices—but it leads to a one-time improvement. Thus, this form of quantization will not lead to repeated advancements but instead will allow frontier models to be scaled down for broader deployment.

Channels with Potential to Lead to Some Improvements

One channel (sparsity) is likely to result in some sustained and repeatable improvements, but not on the scale of orders of magnitude.

[31] For example, see Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei, "The Era of 1-Bit LLMs: All Large Language Models Are in 1.58 Bits," arXiv, version 1, February 27, 2024.

Sparsity

The scaling laws described in both the Kaplan and Hoffmann et al. articles mentioned in Chapter 2 applied to dense neural networks. If sparsity can be introduced (for instance, through pruning or regularization) in a way that does not substantially deteriorate performance, the inference FLOPs would decline proportionally. Furthermore, if sparsity patterns were known in advance of training, then mathematical techniques might be developed to exploit those patterns, and training FLOPs could also be reduced proportionally.

MoE, a type of dynamic compute graph, is another approach to exploiting sparsity. Much like random forests are mixes resulting from a variety of classification and regression trees, MoE mix results from a variety of smaller models. A study by Xu Owen He found that using a large number of small experts (more than 1 million) could result in higher accuracy for a fixed computational cost than a comparably sized non-MoE LLM.[32] Similarly, Tal Shnitzer et al. found that performance could be improved by identifying the "best model" for a given task from a pool of experts and then applying that model to the task.[33]

Advancements related to sparsity should be expected to result in incremental improvements or refinements to a system rather than improvements on the scale of orders of magnitude.

Channels with Potential to Lead to Substantial Improvements

There are three channels that have the potential to achieve large improvements in algorithmic performance. For these channels, we describe the research we reviewed and some potential indicators for whether a specific channel will result in sustained and repeatable advancement.

Data Tailoring

There have been several studies assessing options for pruning data or otherwise improving the quality of data used for models. These have been found to produce comparable results with substantially less data. In different contexts, studies have found that models trained on datasets that were 20 to 99 percent selectively pruned resulted in minimal reductions in performance.[34] Additionally, Wang et al. developed an approach to efficiently generate high-quality synthetic

---

[32] Xu Owen He, "Mixture of a Million Experts," arXiv , version 1, July 4, 2024.

[33] Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin, "Large Language Model Routing with Benchmark Datasets," arXiv, version 1, September 27, 2023.

[34] The selection processes for the pruning are described in the specific papers and future studies might find generalizable approaches for the pruning process. See Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos, "Beyond Neural Scaling Laws: Beating Power Law Scaling via Data Pruning," *Advances in Neural Information Processing Systems 35*, proceedings of the 36th International Conference on Neural Information Processing Systems, 2022; and Raphaël Pestourie, Youssef Mroueh, Chris Rackauckas, Payel Das, and Steven G. Johnson, "Physics-Enhanced Deep Surrogates for Partial Differential Equations," *Nature Machine Intelligence*, Vol. 5, December 2023.

data for use in reinforcement learning (RL).[35] These methods distill large datasets into smaller samples that allow for more-efficient training.

Alternatively, there are methods that tailor data by synthetically generating examples to improve performance. For example, researchers at Google DeepMind described an approach in which synthetic training data were generated for geometric proofs and used to fine-tune a model that achieved a silver medal score on the International Mathematical Olympiad test.[36] This type of fine-tuned model shows that synthetic data can be used to produce highly effective models for a narrow class of problems. It is plausible to believe that similar approaches could be used to generate synthetic data for other narrow problems if there is sufficient commercial interest to warrant the attention. However, these examples are limited to narrow types of problems, and there has not yet been a published approach on a generalized synthetic generative tool for training frontier models. This is one approach that can be thought of as operating on the extensive margin, because training on the tailored data results in a model with new capabilities, such as producing geometry proofs.

The development of a generalized data pruning approach would be an indicator that training costs could become 1 percent or less of the training costs based on existing scaling laws.[37] In other words, models that use this approach might be able to scale (data permitting) 100 times for the same cost as a prior generation model. However, while a method to generate generalized synthetic data could result in models that were highly capable for a variety of sophisticated knowledge tasks, such an approach would result in a dataset that covered a large variety of topics and epistemologies.

Fundamentally, if a data curation (either pruning or generating) approach could select the precise quantity and disposition of the data to be fed into a pretraining algorithm to optimize the information gain, then a new class of scaling laws could be developed to maximize computational efficiency. Consider the following examples: If a system were trained on the writings of Jack Torrance from *The Shining,* there would be no marginal benefit from an additional sentence, page, volume, or library because of the repetition of "All work and no play makes Jack a dull boy."[38] The marginal information content is zero. Similarly, a dataset consisting of every grammatically correct English language sentence of at most twenty words could be used directly to train a model about the validity of English language sentences, but

---

[35] Zhilin Wang, Yi Dong, Olivier Delalleau, Jiaqi Zeng, Gerald Shen, Daniel Egert, Jimmy J. Zhang, Makesh Narsimhan Sreedhar, and Oleksii Kuchaiev, "HelpSteer2: Open-Source Dataset for Training Top-Performing Reward Models," arXiv, version 1, June 12, 2024.

[36] Google DeepMind, "AI Achieves Silver-Medal Standard Solving International Mathematical Olympiad Problems," July 25, 2024.

[37] One potential area for exploration is using a measurement of information entropy to either prune existing data or generate synthetic data.

[38] *The Shining*, dir. Stanley Kubrick, Warner Bros., 1980.

without a mechanism that pruned for factual content, this model would not be useful for providing factual statements.

Objective Functions

For any optimization problem, the goal is to find the input value (or values) that maximize (or minimize) the objective function. A common objective function used in machine learning is the cross-entropy loss function, which calculates the difference between the predicted value and the actual value for examples in the training set. For LLMs, these values are based on the next word or token in a sequence. While the cross-entropy loss function is a useful measure of performance, it is not the precise objective function of users during the inference stage of an LLM. Users might want factual information, stylistic content, or something else that could be more or less correlated with the cross-entropy loss function. Thus, there is an inherent misalignment between an LLM's actual commercial relevance and the performance measures it achieves during pretraining and fine-tuning.[39] Techniques, such as RLHF, have been found to result in superior performance but are very expensive to implement.[40] See Appendix D for a case study of this technique.

The invention of alternatives to the cross-entropy loss function that are both efficiently computable and closer to users' preferences would be an indicator of a faster pace in AI development. The magnitude of this effect is fully dependent on the details, so we do not have an estimate for the likely effect.

Complexity Tradeoffs

Alternative algorithms to transformers—such as Mamba,[41] which is subquadratic in computational complexity, or Kolmogorov-Arnold Networks,[42] which require many fewer iterations because of better performance per iteration—have been found to perform better than similarly sized transformers.[43] Alternative algorithms such as these could be less computationally intensive to train for a given size; therefore, a better performing model could be developed for a fixed budget of compute than with a transformer-based model.

---

[39] A recent advancement that has shown promise is the development of *transfer learning*, whereby pretrained models gather knowledge from one task and apply it to another task (Emmanuella Budu, "What Does Pre-Training a Neural Network Mean?" Baeldung, July 21, 2022).

[40] Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei, "Deep Reinforcement Learning from Human Preferences," arXiv, version 4, February 17, 2023.

[41] Albert Gu and Tri Dao, "Mamba: Linear-Time Sequence Modeling with Selective State Spaces," arXiv, version 1, December 1, 2023.

[42] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark, "KAN: Kolmogorov–Arnold Networks," arXiv, version 2, May 2, 2024.

[43] As one reviewer of this report noted, adoption of Mamba appears to be slow, and that could be, in part, because of the significant effort that would be required to deviate from existing methods.

So far, these algorithms have been demonstrated to work well on small scales and in limited contexts. What is not known is the degree to which the performance of these alternatives can scale. Additionally, there could be a substantial incumbency bias for transformers because hardware development and other components of AI systems have been optimized for that architecture over the past few years. Thus, even if alternatives have superior performance in an abstract sense, they might be less likely to be pursued because the costs of switching grow as more investments are made around the existing architecture.

If the performance of these alternative models scales efficiently, it would be an indicator that model training costs will decline substantially, particularly for larger models. It is plausible that these types of approaches could reduce the cost of training models by at least an order of magnitude. Though, for context, an order of magnitude would amount to only a few years' worth of improvement at the pace of AI advancement since the introduction of the transformer.

Summary of Advancement Channels

The bottom line is that, in the next few years, there are many plausible paths by which LLMs could achieve substantially better performance on a fixed compute budget. In particular, if data curation is systematized and transformer alternatives scale, LLMs and large multimodal models performing equivalently to today's state-of-the-art models could plausibly be trained for multiple orders of magnitude less compute, and larger models could become exponentially better than today's frontier models. However, if the barriers to advancement previously discussed (e.g., learning efficiency, data constraints) are not surmounted, progress in the largest models could slow.

Chapter 4. Conclusions and Early Indicators

Based on the different early indicators described at the end of the last chapter, we have specified three distinct possible trajectories for the near-term advancement of AI systems.

Possible Futures

Data Limitations Are Binding

If synthetic data generation does not lead to the ability to meaningfully scale future model training much past the stock of easily accessible, high-quality public data,[44] or if alternative architectures are not able to train more efficiently than existing models, then we would not expect substantial performance improvements of frontier models in the near future. However, the introduction of new datasets could lead to some focused improvements. This would mean that the computational demand for training frontier models would not continue to grow and, therefore, we should expect a greater relative demand for computational budget for inference.

In this environment, there could still be substantial advancement on smaller models, particularly those models that are tailored to specific problems or modalities.

Algorithms Fail to Scale

If synthetic data generation leads to the meaningful scaling of datasets, but alternative architectures are not able to train more efficiently than existing approaches, we would expect that the frontier models could continue to grow based on the ability to produce additional synthetic data. However, the cost of those models would not be worthwhile for most fields. In particular, removing data constraints from LLM training would not reduce the cost of inference.

For example, if synthetic datasets could be generated along the lines of AlphaGeometry and AlphaProof for a broad variety of fields,[45] but the efficiency by which the models learned from those datasets did not substantially improve, it would still take tens or hundreds of millions (or more) of generated examples to train models to master a given field. In the absence of substantial transfer learning across fields, general models could require tens or hundreds of billions (or more) of examples to learn. That would make inference on the larger generalized models more expensive than a model specialized for a given task. In that case, specialized models would likely be preferred by users, and larger models might not be commercially viable.

---

[44] Villalobos et al., 2024.

[45] Trieu H. Trinh, Yuhuai Wu, Quoc V. Le, He He, and Thang Luong, "Solving Olympiad Geometry Without Human Demonstrations," *Nature*, Vol. 625, January 18, 2024.

Larger models could be developed, but performance improvements would come from larger datasets and increased computational spending rather than a more-efficient use of data. If this is the case, we expect to see more work in advancing small models because larger models would not provide a substantial improvement relative to their computational costs.

Scaling Continues

If there are advancements in both synthetic data generation and the efficiency of model training, we would expect substantial returns to scale and continued competition to build larger models. Furthermore, improvements to computational efficiency during training can also be expected to reduce computational costs during inference. In this environment, small models could still have a role for niche tasks, such as on edge devices, but efficiency would tend to increase with scale, which might outweigh even the small amount of costs and time needed to develop niche models.

Recommendations for Policymaking

While the pace and trajectory of algorithmic advancement is highly uncertain, there are indicators—such as those identified in the previous sections—that can be used to inform policymaking related to AI. Therefore, policymakers should consider investing in technology scanning capabilities related to algorithmic advancement, particularly in the areas of synthetic data generation, data pruning, and the scalability of transformer alternatives. By monitoring these types of advancements, policymakers can have some foresight into which of the futures discussed in this report is most likely in the near term.

Additionally, one question this study raises but does not address is the adequacy of the objective functions used in existing AI systems. The cross-entropy loss function is conceptually very useful for predicting the next token, but algorithms seeking to eke out incremental improvements in that metric are outperformed in some tasks when RLHF is applied to the pretrained model. Scaling RLHF for training LLMs poses coordination challenges,[46] so any technology scan should also include progress on post-training adjustments, such as RL, that are more easily scalable.[47]

When considering international AI competition, improvements in algorithmic efficiency might reduce the efficacy of policies that restrict access to computation. We discuss this in more detail in Appendix C.

Additional study will be needed because of the pace of algorithmic improvements and the breadth of active research efforts at the frontier.

---

[46] Jian Hu, Xibin Wu, Zilin Zhu, Xianyu, Weixun Wang, Dehao Zhang, and Yu Cao, "OpenRLHF: An Easy-to-Use, Scalable and High-Performance RLHF Framework," arXiv, version 4, November 24, 2024.

[47] A potential example is Anthropic's Constitutional AI (Anthropic, "Constitutional AI: Harmlessness from AI Feedback," policy memo, December 15, 2022).

Appendix A. Background on the Computational Effort Associated with Machine Learning Algorithms

This appendix provides a primer on the computational effort associated with machine learning algorithms. It provides basic information (with examples) about types of machine learning algorithms and measures of computation for training and for inference.

Machine Learning Algorithms

Portions of this report assume that the reader has familiarity with algorithms in general and machine learning algorithms in particular. Nonetheless, in this appendix, we provide some summary level information and references by which the interested reader can acquire the necessary background. An *algorithm* is a well-defined procedure for transforming a set of input values to a set of output values and can be thought of as a tool for solving a well-specified computational problem.[48] Machine learning is considered a branch in the field of AI and computer science concerned with the use of data and algorithms to imitate the way that humans learn.[49]. There are varying definitions of AI. For example, in their book *Artificial Intelligence: A Modern Approach*, AI researchers Peter Norvig and Stuart Russell organize definitions of AI into four broad categories: thinking humanly, acting humanly, thinking rationally, and acting rationally.[50] AI can be further subdivided into types, such as narrow AI, which focuses on algorithms for specific tasks, and artificial general intelligence, a state in which machines acquire an intelligence equal to or surpassing humans and possess a self-aware consciousness. A comprehensive treatment of AI requires delving into a vast array of subjects, including philosophy, and will be avoided here. The focus of this report is on practical aspects of machine learning, which "involves creating models by training an algorithm to make predictions or decisions based on data [and] encompasses a broad range of techniques that enable computers to learn from and make inferences based on data without being explicitly programmed for specific tasks."[51]

The life cycle of a machine learning algorithm can be categorized into two broad phases: training and inference. During the training phase, the algorithm processes data to acquire the

[48] For an introduction to algorithms, see Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to Algorithms*, 3rd ed., MIT Press, 2009.

[49] Cole Stryker and Eda Kavlakoglu, "What Is Artificial Intelligence (AI)?" IBM, August 9, 2024.

[50] Stuart Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed., Pearson Education Limited, 2014, p. 2.

[51] Stryker and Kavlakoglu, 2024.

expertise that it will need to make predictions or classifications. During the inference phase, the algorithm makes predictions or classifications from novel data. A few examples are explored in the following sections.

Supervised Learning Applications

The applications described in this section are examples of *supervised learning*, meaning that each feature set in the training data had a corresponding label that specified the desired output. Supervised learning usually requires human analysts to provide or at least validate the labels for the training data. For instance, consider developing a training set for an image classification system. Imagery information (e.g., pixel values or information derived from Fourier-domain representations of the pixel values) might provide the feature sets, and human analysts might apply their judgment to label each feature set as belonging to a tank versus a truck.

Least Squares Data Fitting

Linear algebra procedures for least squares data fitting should be familiar to anyone who has completed a course in linear algebra, and, perhaps, this might not seem to be an example of a machine learning algorithm. However, consider that training consists of fitting a curve to a dataset comprised of a discrete set of inputs, called the *feature set*, and the corresponding outputs, called the *labels*. Once the model parameters of the curve have been adequately fit to the training dataset (as measured by the sum of the squared errors), then the curve can be used to make predictions of the output from new input data, which is inference. There is a closed-form expression for solving a linear least squares problem and iterative approaches for solving nonlinear least squares problems (we will have more to say about types of solutions later in this section).[52]

Logistic Regression

This is the process of fitting the parameters logistic function to a training set using maximum likelihood criterion and is procedurally similar to least squares data fitting. The characteristic "s" shape of a logistic function is useful for classifying features into binary categories. That is, the label for each corresponding feature set is associated with one of two categories, such as categorizing image features as either belonging to a "tank" or "truck."[53]

Linear Classification and Support Vector Machines

Linear classification is the process of finding the equations of hyperplanes that separate (or classify) input feature sets into their own unique regions (each label pertains to a distinct region).

---

[52] For more information, see Stephen Boyd and Lieven Vandenberghe, "Least Squares Data Fitting," in *Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares*, Cambridge University Press, 2018.

[53] Stephen Boyd and Lieven Vandenberghe, "Statistical Estimation, " in *Convex Optimization*, Cambridge University Press, 2004.

It is not always possible to perfectly classify features into distinct regions; hence, the user might have to accept some misclassifications. The support vector machine is a variation of linear classification that adds a heuristic to minimize the number of misclassified points. Linear classification and support vector machine algorithms can be generalized to use nonlinear classification if the nonlinear function is affine in the parameters that define it (e.g., polynomial classification). Support vector machines have been used in spam filtering applications.[54]

Neural Networks

Neural networks consist of layers of interconnected logistic functions. The mathematical behavior of the output of a logistic function to its inputs provides a useful model of the behavior of the output of a neuron in the human brain via its axon terminals to electrical inputs on its dendrites. The human brain, which has about 86 billion neurons, represents learning as patterns of electrical signaling through networks of neurons. Artificial neural networks are similarly used to represent learning as patterns of mathematical signaling through networks of logistic functions. Neural networks consist of an input layer, an output layer, and usually one or more so-called "hidden" layers in between. The output of a neuron from one layer is connected to the input of each neuron of the subsequent layer, and there is a *weight* associated with each connection (*weights* are numerical values in an artificial neural network that are similar to the synapses in biological neural networks). The numerical values of the features are provided to the input layer. During learning, the values of the weights are adjusted so that the outputs of the neurons in the output layer statistically correspond to the desired label.[55] Neural networks are used for a wide variety of applications, including image classification and natural language processing in AI models, such as LLMs. This is an intensely active area of research and there are many variations of neural networks.

Unsupervised Learning Applications

There are also examples of machine learning algorithms that are *unsupervised*, meaning that the training set consists of only features and no labels. With unsupervised learning, we want the machine to learn some relationship between the inputs. We explore some examples in the following sections.

Multivariate Gaussian Model Fitting

A multivariate Gaussian probability distribution is entirely defined by the mean and covariance. Hence, we can use the sample mean and sample covariance associated with a set of input data to fit a Gaussian probability distribution and use it to make predictions for novel data.

---

[54] Stephen Boyd and Lieven Vandenberghe, "Geometric Problems, " in *Convex Optimization*, Cambridge University Press, 2004.

[55] A classical treatment of neural networks is available in the "Deep Learning" chapter of Russell and Norvig, 2014.

There are no labels in this application, just collections of features that are assumed to be well modeled as Gaussian random variables. Applications include anomaly detection, for instance, to improve engine maintenance. Features might include temperature and vibration measurements associated with an engine. The model is trained using samples from engines that are in working condition. Then, the model can be used to detect characteristics that are statistically inconsistent with a working engine.

Cluster Analysis

In these applications, an algorithm is used to collect natural groupings of input data into distinct clusters without the aid of labels. One example is the K-means algorithm for categorizing inputs into the number K distinct clusters based on the Euclidean distance of each input data point to a mean value that defines each cluster. Typically, the K clusters are initialized to randomly selected mean values. Then, each data point from the input is assigned to the closest cluster. Once all the data points have been assigned, the mean value of each cluster is updated, and the algorithm iterates until either the maximum change in any given mean is below some threshold value or a maximum number of iterations is exceeded. A notional application might be market segmentation: A company collects information about the users of its products and uses cluster analysis to bin users into distinct use cases so it can optimize its business strategy based on user needs.

Summary of Learning Applications

As we have seen, in supervised learning, the model parameters are trained using specific labels, whereas with unsupervised learning, there are no labels, and the model parameters are trained to find relationships in the input data. Another variation is RL, during which the model parameters are trained to maximize rewards or minimize some type of penalty, also without the use of labels. In this variation, the model might incorporate a neural network, but rather than having labels, the model perceives its environment and takes actions based on the outcomes of trial and error. Google used this approach to automate the cooling of its data centers.[56]

Computational Effort and Resources for Training

One important measure of the computational effort required for training or inference is the number of FLOPs that are needed. Most computational systems represent real numbers using the Institute of Electrical and Electronics Engineers Standards Association's standard for floating-point arithmetic (IEEE-754, 2008).[57] Adding, subtracting, multiplying, or dividing two floating-

[56] Chris Gamble and Jim Gao, "Safety-First AI for Autonomous Data Centre Cooling and Industrial Control," Google DeepMind, August 17, 2018.

[57] Institute of Electrical and Electronics Engineers, "754-2008: IEEE Standard for Floating-Point Arithmetic," August 29, 2008.

point numbers all require about the same amount of computational effort. We call this effort one *FLOP*. Comparing two numbers might require a bit less effort, but we usually count the effort as one FLOP. The amount of effort for computing a square root is typically counted as approximately 5 FLOPs, and trigonometric functions are somewhere in the range of 15 to 20 FLOPs, depending on the method and the range of the variables.

Similarly, FLOPs per second (FLOP/s) provides a useful measure of the computational performance of a computing system and is used for important benchmarks for high-performance computing. For instance, in November of 2023, the Frontier system at Oak Ridge National Laboratory achieved 1.194×10^{18} FLOP/s using a combination of 8,699,904 combined central processing unit (CPU) and graphics processing unit (GPU) cores (Top500, 2023).[58] Also, computational performance tends to scale linearly with input power; hence, FLOP/s per watt (FLOP/s/W) is also used as a measure of performance for computing systems. The Frontier system performance is 52.59×10^9 FLOP/s/W.[59]

Compared with the multiple instruction and multiple data architecture of CPUs, the single instruction and multiple data architecture of GPUs are better suited to the types of computations associated with neural networks, and GPUs are widely used in the training and inference of large neural networks, including LLMs. As an example, consider the NVIDIA A100 Tensor Core GPU with 32-bit floating-point arithmetic. According to NVIDIA, each device is capable of 152×10^{12} FLOP/s (NVIDIA, 2021).[60] Typically, hundreds or thousands of these devices are employed to carry out computations in parallel for the purposes of training LLMs.[61]

The amount of effort required for machine learning varies by algorithm and approach. There is a closed-form expression for the optimal solution to a linear least squares problem, and it requires $mn^2 + (1/3)n^3$ FLOPs using Cholesky factorization, where n is the number of model parameters and m is the number of samples in the training dataset.[62] Logistic regression and support vector machine training do not have closed-form expressions, and iterative methods are required. Depending on the specific approach, each iteration requires approximately n^3 FLOPs to compute the error function and one or two derivatives. Logistic regression and support vector machine training is a convex optimization problem,[63] so there is a bound on the number of iterations that are needed to find a globally optimal solution. That bound is polynomial in the

---

[58] "Frontier Remains No. 1 in the TOP500 but Aurora with Intel's Sapphire Rapids Chips Enters with a Half-Scale System at No. 2," Top500, undated.

[59] "Frontier Remains No. 1," undated.

[60] NVIDIA, "NVIDIA A100 Tensor Core GPU," data sheet, June 2021.

[61] Executive Order 14110 set reporting requirements for (1) any computing cluster in a single data center having a theoretical maximum computing capacity of 10^{20} FLOP/s or greater for training AI and (2) any model that was trained using 10^{26} or more FLOPs (Executive Order 14110, "Safe, Secure, and Trustworthy Development and Use of Artificial Intelligence," Executive Office of the President, October 30, 2023).

[62] Boyd and Vandenberghe, 2018, pp. 191, 231.

[63] For more information about convexity and its implications, see Boyd and Vandenberghe, 2004.

number of parameters, number of samples in the training dataset, and number of desired digits of accuracy. In practice, 20 to 50 iterations are required.[64]

For many supervised machine learning algorithms, including neural networks which are emphasized in this report, training involves the minimization of a so-called "loss" function. While there are several variations, commonly used loss functions are the squared error between the model prediction and the label, or a variation called *log loss*, which is derived from maximum likelihood estimation. Training of a neural network involves minimizing the loss function typically using a variation of the gradient descent method.[65] It also involves evaluating both the loss function using a technique called *forward pass* and its derivative using a technique called *backward pass* (these names derive from the direction of the computation relative to the input layer and output layer). The computational effort of each pass (forward and backward) is $4n$ FLOPs for a total of $8n$, where n is the number of parameters in the neural network.[66] The calculations could be conducted in parallel and the effort divided across computing devices. As an example, consider a fully connected neural network with an input layer and an output layer each having 49,152 neurons and 96 hidden layers having 12,288 neurons each. Then the total number of parameters would be

$$n = 2 \times 49{,}152 \times 96 \times 12{,}288 \approx 116 \times 10^9.$$

This is about two-thirds of the 174.6×10^9 parameters associated with OpenAI's Generative Pretrained Transformer 3 (GPT-3) LLM.[67] Hence, training an LLM such as GPT-3 requires about $8 \times 174.6 \times 10^9 = 1.3968 \times 10^{12}$ FLOPs per iteration.

How many iterations are needed? Unfortunately, training a neural network is a nonconvex optimization problem and, as a result, there is no polynomial bound on the number of iterations and no guarantee that the algorithm will converge to a globally optimal solution. Fortunately, the performance of neural networks on their intended tasks tends to have excellent performance compared with competing methods (including humans) despite the implications of suboptimal training. Typically, LLMs, such as GPT-3, are trained with one iteration each on every piece of data in their corpus. The data are divided into chunks of text called *tokens*. OpenAI reported that

---

[64] See "Unconstrained Minimization" and "Interior-Point Methods" in Boyd and Vandenberghe, 2004.

[65] See "Unconstrained Minimization" in Boyd and Vandenberghe, 2004.

[66] Dzmitry Bahdanau suggests that, theoretically, $6n$ total FLOPs are needed but suggests that 8n is a better estimate to use because of practical details that almost always apply (Dzmitry Bahdanau, "The FLOPs Calculus of Language Model Training," Medium, January 9, 2022).

[67] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al., "Language Models Are Few-Shot Learners," arXiv, version 4, July 22, 2020, p. 46.

GPT-3 was trained on approximately 300 billion tokens, occupying 570 gigabytes of storage.[68] Hence, the training effort required is about $300 \times 10^9 \times 8 \times 174.6 \times 10^9 = 4.2 \times 10^{23}$ FLOPs.

Suppose the computations are parallelized across 1,024 NVDIA A100 GPU devices, each capable of 152×10^{12} FLOP/s. Then the computational effort associated with the FLOPs for training an LLM such as GPT-3 would require $4.2 \times 10^{23} / (1{,}024 \times 152 \times 10^{12}) = 2{,}698{,}396.4$ seconds, which is about 31 days.

Other Factors and Resources Contributing to Training Time

FLOPs account for only a portion of the actual training time. Lucia Mocz estimates total training time as the time required for the FLOPs, plus a factor Mocz refers to as *bandwidth* and a factor for *overhead* (which we cover in the next paragraph).[69] *Bandwidth* accounts for the time required to move the data from input storage and through the parallel processing architecture. Let T_B denote the total time associated with bandwidth. Mocz estimates this as

$$T_B = N_B N_U S(N_n - 1) / R$$

where N_B is the number of bytes used to represent each model's parameters, N_U is the number of update transfers, S is the size of the update for each processing node, N_n is the number of processing nodes in the parallel processing architecture, and R is the data transfer rate in bytes per unit time. The training data for LLMs are loaded or updated in batches. Hence, if N_T denotes the total number of tokens in the training data, and B denotes the batch size in tokens, then the number of updates is $N_U = N_T / B$. The size of the update per processor is calculated as $S = N_B N_p / N_n$. As a numerical example, consider again a model such as GPT-3 with $N_T = 300 \times 10^9$ tokens, $N_p = 174.6 \times 10^9$ parameters, and processed on a system employing $N_n = 1{,}024$ GPUs. According to OpenAI, this model was trained with a batch size of $B = 3.2 \times 10^6$ tokens. Hence, the number of updates is $N_U = 300 \times 10^9 / (3.2 \times 10^6) = 93{,}750$. Assume that we use $N_B = 2$ bytes to represent each parameter; then, the size of the update for each processing node is

$$S = 2 \times 174.6 \times 10^9 / 1{,}024 = 341{,}015{,}625 \text{ bytes.}$$

Mocz assumes a transfer rate of $R = 2 \times 10^{11}$ bytes per second, though the source of this assumption is not explained. Using this example data, we find that the bandwidth contributions to training time is

$$T_B = 2 \times 93{,}750 \times 341{,}015{,}625 \times 1{,}023 / (2 \times 10^{11}) \approx 327{,}535 \text{ seconds,}$$

---

[68] Brown et al., 2020, p. 46.

[69] Lucia Mocz, "Performance Bottlenecks in Deploying LLMs—a Primer for ML Researchers," Medium, May 10, 2023.

which is about 3.8 days.

Mocz describes *overhead factors* as relating to additional computational costs associated with synchronization, coordination, and communication during training that are not related to the bandwidth or FLOPs, and she finds that the overhead does not scale with model size. No details are provided for estimating overhead related delays, but in the examples for an LLM with 65 billion parameters and 1.4 trillion tokens, it is suggested that the overhead delays are between 6 and 11 days.

Hence, if we add together the 31 days required for FLOPs, with the 3.8 days of bandwidth, and 6 to 11 days overhead delay for our GPT-3 example (174.6 billion parameters and 300 billion tokens), then the total training time is estimated to be between 41 and 46 days.

Another factor that contributes to training time is *sample efficiency*. An algorithm is sample efficient if it can get the most out of every training sample. A related concept is the *sample complexity*, which, in machine learning, is "how many examples are required to guarantee a probably approximately correct solution,"[70] and the sample complexity depends on the desired accuracy and confidence that is needed in a given application.

Computational Effort and Resources Required for Inference

For simple machine learning models (such as least squares, logistic regression, and support vector machines), inference involves a simple inner product of an input vector with the vector of model parameters, which requires approximately $2m$ FLOPs, where m is the number of model parameters (more precisely, it requires m multiplications and $m - 1$ additions which is approximately $2m$ for large m).

For neural networks, including LLMs, interference requires conducting a single forward pass; hence, it requires about $4n$ FLOPs where n is the number of model parameters. Consider again our GPT-3 example with 174.6×10^9 parameters. If we apply the same architecture using 1,024 A100 processors for inference, then the estimated time for the computations would be

$$4 \times 174.6 \times 10^9 / (1{,}024 \times 152 \times 10^{12}) \approx 4.5 \times 10^{-6} \text{ seconds}$$

for a single inference. If we have a 40 token prompt (which is likely about 30 words), this should take approximately 1.8×10^{-4} seconds. This amount of effort is trivial compared with the effort required for training. As a second example, consider a desktop computer with a performance specification of 20×10^9 FLOP/s/W and an input power of 75 watts. The estimated time for the computations would be

[70] Shai Shalev-Shwartz and Shai Ben-David, *Understanding Machine Learning: From Theory to Algorithms*, Cambridge University Press, 2014, p. 44.

$$4 \times 174.6 \times 10^9 / (20 \times 10^9 \times 75) \approx 0.467 \text{ seconds}$$

for a single inference, or about 19 seconds for a 40-token (30-word) prompt.

Measuring Performance for Tasks

Evaluating the loss function used for training a machine learning model with the training data provides a measure of how well the model fits the data for a given task, but this does not provide a useful measure of how well the model will generalize to new data for that task. For this reason, datasets are often divided into subsets for training and cross-validation and a third subset for fine-tuning. Evaluating the loss function with the cross-validation set provides an estimate of how well the model will generalize to new data. And comparing performance of the model with the training and cross-validation sets can provide useful insights about whether the model is poorly fit (called *bias*), overfit (called *variance*), or appropriately fit. The comparison can also yield insights about the size of the training dataset and diminishing returns on increasing the size.

For detection and classification problems, we may use sample statistics related to probability of success with a cross-validation set as a measure of performance. For instance, probability of detection versus false alarm rate, or probabilities of Type I errors (false positives) and Type II errors (false negatives) using sample statistics. In some cases, we might want to compare these algorithm-obtained measures with human performance on the same task or compare the performance of two competing algorithms. For instance, we could take the probability of detection and false alarm rate sample statistics of detecting a target from radar imagery using an algorithm and compare it with human analysts.

In machine learning applications to zero-sum games, we might measure the performance of two algorithms or the performance of an algorithm versus a human using a relative rating system, such as the Elo system.[71] Elo is a method of calculating the relative skill of two players. It was invented for chess and is intended to predict the outcome of a match assuming a normal distribution: A player with a rating that is 100 more than their opponent has a 64-percent chance of winning; with a rating that is 200 more, the chance of winning increases to 76 percent. The Elo system is used to compare the chess performance of AI algorithms with humans or with other algorithms. It is also used to assess algorithms performing the board strategy game Go.

A wide variety of techniques are used to evaluate the performance of LLMs. The performance of deployed LLMs has been measured on standardized exams that are designed for humans, such as college entrance examinations, software coding challenges, and bar exams. GPU utilization metrics are used, such as counting the number of prompt and completion tokens. In some applications, we are interested in measures of human preferences, which are highly subjective but can be measured using survey techniques. For instance, we can provide a set of

[71] Grace, 2013.

prompts to two competing algorithms and survey a group of human reviewers to measure their preferences.[72]

Of course, we are also interested in the performance of an algorithm for a given task relative to the amount of resources needed for training or inference. Questions such as the following are relevant to such an assessment: For a given level of performance on a specific task, how many FLOPs were needed to train the model? How much data storage is required? How long did it take?

Finally, we might also be interested in the flexibility of a machine learning algorithm to perform a variety of tasks, how well it performs in each, and how many resources are required compared with other alternatives.

[72] Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, et al., "RLAIF: Scaling Reinforcement Learning from Human Feedback with AI Feedback," arXiv, version 2, December 1, 2023.

Appendix B. Survey of Mechanisms for Algorithmic Advancement

This appendix describes the mechanisms that we identified for algorithmic advancement from various computational fields. We reviewed classes of algorithms from numerical analysis, operations research, and computer science to identify paths by which algorithms changed; our intent was to identify key mechanisms that could be relevant to advances in AI. We identified the classes of algorithms by reviewing standard textbooks from these fields,[73] so this appendix should be thought of as more of a survey than a comprehensive and exhaustive examination of the algorithms in these spaces.

For each specific class of algorithm, we reviewed specific algorithms described in the textbooks and compared them to understand the specific mechanism that explains the distinctions. We then sorted these mechanisms into groups that could be useful for exploring paths for AI algorithms to advance (as discussed in Chapter 3).

Numerical Analysis

Approximation and Interpolation

The goal of approximation is to closely match the behavior of a function with a computationally simpler function. Relatedly, interpolation is the process of estimating function values between data points.[74] We found that the primary distinction between different approximation and interpolation algorithms related to either the types of data used or the data quality and the objective function (or error measure) used.

Data Types and Data Quality

There are a variety of classes of interpolation that vary based on the desired fit of the solution. For example, Lagrange interpolation finds a polynomial that exactly matches a set of data points. Hermite interpolation extends Lagrange interpolation by matching both the position of the data points but also some number of derivatives at those points. Thus, if more information about the data points is available, then Hermite interpolation can use that to better match all available information. This improved fit comes at a cost. For n data points, Lagrange interpolation will produce a polynomial of degree $n - 1$ or less, while Hermite interpolation will have a polynomial of degree $(m + 1) \times (n - 1)$, where m is the number of derivatives included.

[73] We specifically used Anthony Ralston and Philip Rabinowitz, *A First Course in Numerical Analysis*, 2nd ed., Dover, 2001; J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, 3rd ed., Springer, 2002; Richard L. Burden and J. Douglas Faires, *Numerical Analysis*, 9th ed., Brooks/Cole, 2011; and Cormen et al., 2009.

[74] See "Interpolation and Polynomial Approximation" in Burden and Faires, 2011.

Additionally, it is important to note that, as the degree of the polynomial increases, the stability of the values might decrease, and this can be particularly problematic for any extrapolation using the polynomials.

Relatedly, Fourier transforms are a way of identifying the frequencies in a set of data. So, for data on frequencies (e.g., waves of sound, light, or matter) Fourier transforms can find the best fitting wave function as opposed to the best fitting polynomial for Lagrange or Hermite interpolation.[75]

Objective Function

One commonly used approach in approximation is the least squares approach in which the objective of the algorithm is to identify the function that minimizes the sum of the square of the error. This is widely used in statistics, at least in part because it is not computationally intensive. Alternative approaches include Chebyshev polynomials for which the goal is to minimize the maximum error or minimize the sum of the error. In each of these cases, the algorithm differs because the goal of the approach is different. In some cases, the result will be relatively close, but in other cases, the outputs of the algorithms can differ wildly. Ultimately, the selection of the objective function should be made based on the intended use of the analysis.

Systems of Linear Equations

Systems of linear equations take the form of $Ax = b$, where A is a matrix, and x and b are either vectors or matrices. These methods are foundational to other numerical analysis methods, including least squares approximation and solutions to partial differential equations.[76] These methods can be either direct or iterative. The algorithms for solving these equations differ based on the structure of the matrices (specifically, the structure related to sparsity), stability in the accuracy of solutions, or the number of iterations involved.

Sparsity

There are a variety of special methods for directly solving linear equations more quickly than the most basic Gaussian elimination.[77] Many of these methods rely on sparsity within the matrix.

---

[75] Richard Haberman, "Infinite Domain Problems—Fourier Transform Solutions of Partial Differential Equations," in *Elementary Applied Partial Differential Equations with Fourier Series and Boundary Value Problems*, 2nd ed., Prentice-Hall, 1987.

[76] See "Boundary-Value Problems for Ordinary Differential Equations" and "Numerical Solutions to Partial Differential Equations" in Burden and Faires, 2011.

[77] Yousef Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, 1996. Many of these methods are described in the Linear Algebra Package (LAPACK). LAPACK contains optimized functions for various classes of matrices and other linear algebra structures. For more information, see E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, "LAPACK: A Portable Linear Algebra Library for High-Performance Computers," *Supercomputing '90: Proceedings of the 1990 ACM/IEEE Conference on Supercomputing*, May 1990.

If there are patterns in the nonzero values of the matrix (e.g., a banded matrix has nonzero values in diagonal bands but the matrix's elements are otherwise zero and an upper-triangular matrix has zeros in every cell below the main diagonal), then specially designed algorithms can take advantage of those.

By taking advantage of these symmetries, the computational cost is reduced loosely proportionally to the square of the number of nonzero elements, and the storage costs are proportional to the number of nonzero elements.

Iterative Methods

Instead of solving the whole problem at once, iterative methods are simpler steps that are repeated to solve the system of equations. Each step in the iteration lowers the error, and, if the process continues n steps, then the exact solution of x is found (the solution with zero error). However, in some cases, the goal might not be to get the exact solution but rather find the solution such that the error is below a given threshold. In those cases, some degree of accuracy can be sacrificed for a reduction in the computational cost.

Differential Equations

Differential equations are a class of problems for which the rate of change of a system is used to estimate the state of the system based on either the initial state of the system or on the state of the system at the boundaries. The distinction between algorithms for differential equations will typically come from trade-offs related to the stability of the solution and the functional forms used to approximate the solution.

Stability

For initial value problems (IVP), one concern is the stability of the solution. There is a trade-off between the resolution and the stability. In other words, for small step sizes, the solutions tend to be less stable because the errors compound. This lack of stability manifests when small differences in the initial conditions result in relatively larger differences as the step size decreases. IVP algorithms trade off between resolution and stability.

Functional Forms

For partial differential equations, one class of algorithms are finite element methods in which the domain is partitioned by a mesh, and, for each cell in the mesh, a basis function is evaluated. In some cases, it is possible to take advantage of symmetries in the underlying problem to reduce dimensions (e.g., using rotational symmetry to reduce a problem of three spatial dimensions into two spatial dimensions in cylindrical coordinates).

Another source of efficiency can be found through a careful selection of the basis functions. For example, if the basis functions are orthogonal to all but the adjacent elements, then the matrix will be sparse and, therefore, solvable quickly relative to a dense matrix.

These approaches will generally reduce the computational cost proportional to the reduction in dimensions for symmetries and the degree of sparsity if orthogonality can be applied to introduce systematic sparsities.

Nonlinear Equations

Nonlinear equations take the form $F(x) = b$, in which x and b can be vectors or matrices and F is a function. While F is assumed to be continuous and differentiable for many algorithms, in some cases a suitable subgradient can substitute for a derivative in the algorithm.[78]

Newton's method, a root-finding algorithm, converges quadratically when the estimate of x is sufficiently near a solution to the nonlinear equation, but it requires the calculation of first and second derivatives, which could be computationally expensive for some functions.[79] Alternatively, quasi-Newton methods converge more slowly (generally super-linearly) but rely on estimates of the first derivative rather than a functional evaluation. Thus, for these nonlinear algorithms, there is a trade-off between the rate of convergence and computational complexity.

Operations Research

Constrained Optimization

Constrained optimization is a class of problems that can generally be written as minimize($F(x)$), in which x is a vector subject to constraints such as $G(x) \leq a$ or $H(x) = 0$. Algorithms in this space vary based on the functional forms involved.

Functional Forms

There are a variety of special cases of constrained optimization that depend on the functional form of F, G, and H.

If F and G are convex functions and H is a linear function, then this is a convex optimization problem that can be solved with polynomial time complexity using an interior point method.[80] Otherwise, the problem is nonconvex and has no known polynomial-time solution. The case in which x is constrained to be integer-valued is also nonconvex.

Stochasticity

Constrained optimization problems may use stochastic factors.[81] One example of this for optimization would be random (or quasi-random) sampling. The user inputs the number of cases

---

[78] Vladimir F. Dem'yanov and Leonid V. Vasil'ev, *Nondifferentiable Optimization*, Optimization Software Inc., 1985.

[79] Boyd and Vandenberghe, 2004, pp. 484-496.

[80] Boyd and Vandenberghe, 2004.

[81] Boyd and Vandenberghe, 2004, pp. 305–317.

that they would like to test (n), a set of n vectors is generated, $x_1 - n$ that meet the constraints in G and H, the function F is evaluated for each x_i, and the maximum is selected from that set as an approximation of the maximum. As n grows, the gap between the actual optima and the estimated optima decreases. Alternatively, because a truly randomly generated set of vectors might not be evenly distributed across the solution space, quasi-random numbers are used to ensure that the solution space is spanned.

Traveling Salesman Problem

The traveling salesman problem seeks to identify the shortest route that connects a series of points in a closed loop. This problem is also known to be NP-hard, so there are a variety of algorithms and heuristics that are used to provide exact or approximate solutions. Many approaches seek to minimize the distance travelled by generating an initial solution and then iteratively improving upon it. Some of these iterative approaches use metaheuristics (such as simulated annealing) that use randomness. For these cases, the key mechanisms for improvement would be trade-offs related to the iterations and the application of stochasticity to move away from local optima.

Computer Science

Compression

The goal of compression is to reduce the number of bytes required to store a file. In general, compression can be categorized as either lossless or lossy. With lossless compression, no information content is lost, and the original data can be fully recovered.[82]

[82] Khalid Sayood, *Introduction to Data Compression*, Morgan Kaufmann Publishers, 1996.

Appendix C. Implications for Hardware Export Controls

Through a series of executive orders and updated export control rules that were instituted in October 2022, October 2023, December 2024, and January of 2025,[83] the United States imposed constraints on the types of chips that can be sold to entities in the People's Republic of China. One rationale for these hardware restrictions is to help the United States retain dominance in the AI space. However, this raises an important question: Does the pace of algorithmic advancement mean that the hardware constraints are likely to be less effective in helping the United States and allied nations retain dominance in AI? [84]

Constraints on computing power most acutely affect the ability to train large foundation models, but they also reduce the ability to conduct experimentation using smaller models. In this appendix, we explore the question of the effectiveness of hardware constraints through the projections of the three futures described in Chapter 4.

In the Data Limitations Are Binding and Diminishing Returns to Scale scenarios, because the largest frontier models do not perform much better than smaller, more-focused models, the demand for computing capacity for training would likely be focused on the smaller models. Thus, the hardware export restrictions would primarily affect the advancement of foreign frontier models by limiting the ability to experiment. In practice, that could reduce the ability of models to innovate, but if ideas related to algorithms continue to be shared in scientific forums,[85] the net effect of a hardware ban on the ability of targeted countries to develop near-frontier models would likely be minimal. There is evidence that researchers in China have been able to identify and deploy the algorithmic advances made by frontier firms, so even the reduced computational budget for experimentation might not be effective.[86]

In the Scaling Continues scenario, the performance of the largest frontier models grows rapidly, and there is significant demand for computing power at the training stage. At the same

[83] Bureau of Industry and Security, "Commerce Strengthens Restrictions on Advanced Computing Semiconductors to Enhance Foundry Due Diligence and Prevent Diversion to PRC," Office of Congressional Affairs, January 15, 2025.

[84] The advancements claimed by DeepSeek-V3 in December of 2024 might suggest that hardware constraints are less important than previously thought. DeepSeek-V3 was unveiled in December of 2024 (well after we conducted our research for this report), and their technical report describes improvements with multi-head latent attention, a new strategy for load balancing, and a multi-token prediction training objective, as well as SFT and reinforcement stages after initial training (DeepSeek-AI et al., 2024).

[85] We observe that some frontier AI model companies (such as OpenAI) are publishing relatively little about their algorithmic advances.

[86] Zhiyuan Zeng, Qinyuan Cheng, Zhangyue Yin, Bo Wang, Shimin Li, Yunhua Zhou, Qipeng Guo, Xuanjing Huang, and Xipeng Qiu, "Scaling of Search and Learning: A Roadmap to Reproduce o1 from Reinforcement Learning Perspective," arXiv, version 1, December 18, 2024.

time, the performance of the largest models of the prior generation could be duplicated with much less computing power. In this environment, the export controls would greatly restrict the ability for frontier AI models to be developed in the targeted countries. However, depending on the nature of the scaling involved and the growth of algorithmic efficiency, models developed in targeted countries could be as large as the frontier models developed in countries with full access to hardware a year or two prior, that is, if researchers in the targeted countries were aware of current advancements.[87]

Another set of considerations for assessing the efficacy of hardware constraints relates to the delivery mechanism for an AI system. Specifically, whether a model is open (either open-source or open-weight) or whether inference is delivered through a closed-source model will determine who bears the computational costs for inference. With a closed-source model, the developer is responsible for obtaining the computational capacity; with an open model, the user or another party can run the model on their own hardware. Similarly, if developments in algorithms push toward more test-time compute,[88] then the burden for delivering the computational capacity will depend on whether a model is open or closed. Hardware constraints may push toward more open models to the extent that open models can be supported through a business case.

The bottom line is that the efficacy of hardware constraints on the ability of targeted countries to develop AI depends, in large part, on the nature of algorithmic advances. The more those advances are biased toward larger models through relaxing data constraints, generating synthetic data, and more efficiently leveraging data, the more impactful hardware constraints will be on AI work in targeted countries.

[87] More-recent estimates related to DeepSeek estimate this timeline to be between seven and ten months (Dario Amodei, "On DeepSeek and Export Controls," blog, January 2025).

[88] Test-time compute is a class of approaches where the response to a prompt is refined iterative during inference. Test-time compute requires much more computation during the inference phase than model that rely on the initial outputs of an LLM.

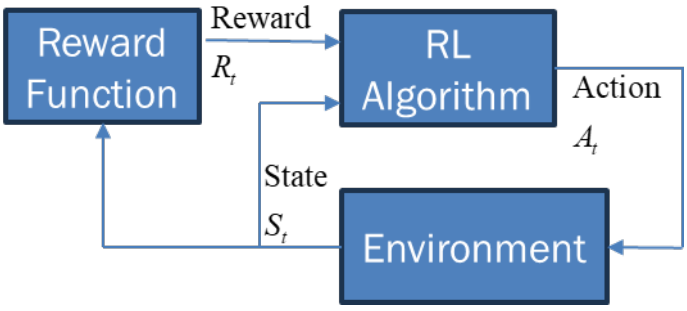
Appendix D. Case Study of Reinforcement Learning from Human Feedback

In this appendix, we discuss several studies that examine the efficacy of various approaches to RL for AI models.

Background and Context

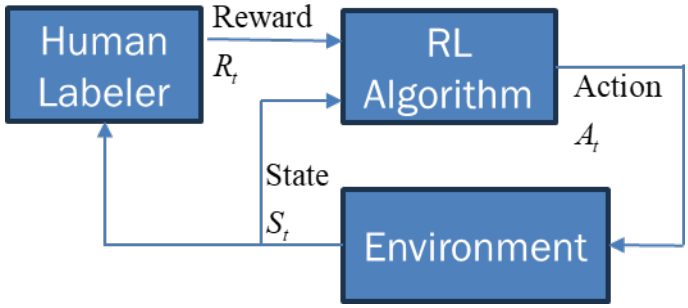
In RL, the model parameters are trained to maximize rewards or minimize some type of penalty without the use of labels. Figure D.1 provides a diagram of the approach. Rather than having labels, the model perceives its environment and takes actions based on the outcomes of trial and error. At each time step t, the RL algorithm generates an action denoted A_t, which updates the state of the environment, denoted S_t. The state of the environment is provided as an input variable for a reward function which generates a reward denoted R_t. The reward and state are used to update the parameters of the RL.

Figure D.1. Reinforcement Learning



But what happens if there is no clear reward function or if the reward function is difficult to assess? For example, consider the task of training a robot to cook an egg or drive a car. What numerical reward could you provide after short time steps that would incentivize the robot to learn the task? Notionally, we could have a human supervisor monitor the state of the environment after each time step and employ human judgment to generate a numerical reward or label to use for reinforcement, as shown in Figure D.2. Unfortunately, most robotic tasks require large numbers of time steps that are very short in duration and would require vast numbers of human generated samples, which would be too time-consuming and expensive to be practical.

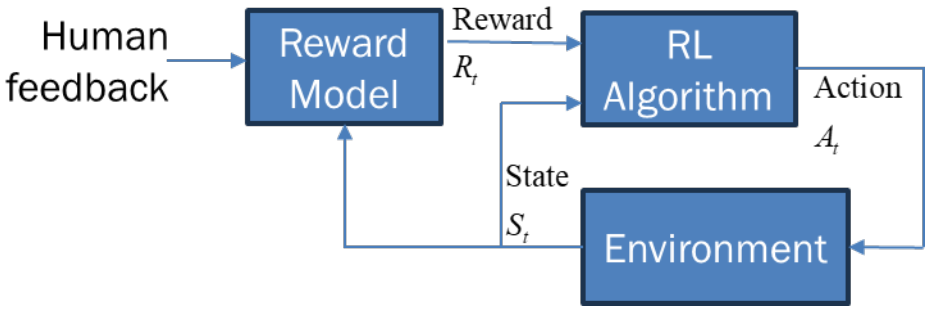
Figure D.2. Human-Supervised Reinforcement Learning



Reinforcement Learning from Human Feedback to Improve Sample Efficiency

An alternative to the supervised RL approach is to substitute the human supervisor with a reward model that has been trained on a smaller set of human samples, as shown in Figure D.3.

Figure D.3. Reinforcement Learning with Human Feedback

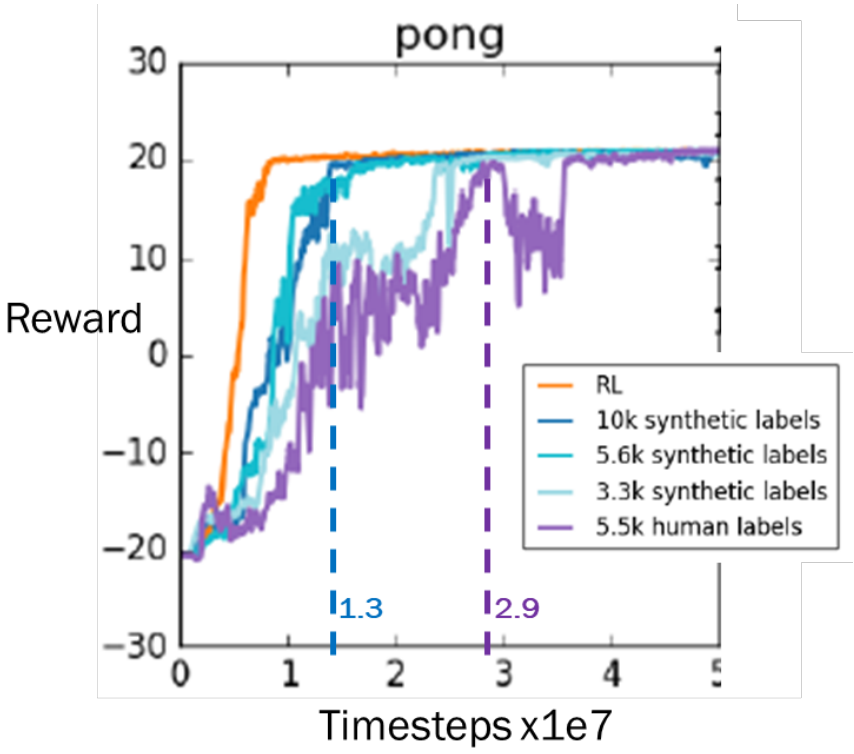


Christiano et al. compared the supervised approach with RLHF for training a deep neural network to play Atari video games (such as Pong) and for simulated robotic locomotion tasks. Quantitative reward functions exist for these applications, but the authors demonstrate how RLHF can be used without access to the reward functions and compare the results of RLHF with RL and with supervised RL approaches. In their research, human supervisors do not directly provide quantitative rewards. Instead, human supervisors are provided with pairs of short video clips (usually 1 to 2 seconds long) of changes to the state of the environment produced from an action. The authors refer to the video clip activity as a *trajectory*. The human supervisor judges whether either of the trajectories is useful for accomplishing the task and, if so, which of the two trajectories is preferred. The results of the human preference samples are used to generate a reward.[89]

[89] Christiano et al. point out that their approach does not require human supervisors with expertise in performing the task. Instead, their approach only requires humans who can judge useful trajectories (Christiano et al., 2023).

Figure D.4 shows the results from Christiano et al. for the Atari game Pong. The horizontal axis of the figure shows the time step, and it appears that there are about 50-million time steps. The vertical axis shows the numerical reward; the paper does not provide any additional details about the units or an interpretation, but, presumably, the reward is related to score in the game. The different color curves denote the reward for different options in training the algorithm. The orange curve shows the performance of RL using the true reward, and we see that the reward asymptotically reaches a maximum of 20 at approximately 10-million time steps. The purple line shows the performance of the human-supervised approach using 5,500 samples of human labels. From the plot, we see the human-supervised approach asymptotically reaches the reward peak of 20 at approximately 29-million time steps. The other colored lines all correspond to RLHF without access to the true reward and trained using 5,500 samples of human labels. We see that with 3,300 synthetic labels, the RLHF performance is already similar to that of the human-supervised approach. Furthermore, with 5,500 or more synthetic labels, the reward of RLHF reaches a maximum of 20 in approximately 13-million time steps compared with 20-million time steps for the human-supervised approach. Hence, the number of time steps needed to reach the maximum reward is reduced by 55 percent. That is, the sample efficiency of RLHF for this example is improved by 55 percent compared with the human-supervised approach.

Figure D.4. Reinforcement Learning from Human Feedback Performance in Learning Pong



SOURCE: Adapted from Christiano et al., 2023, p. 8.

Christiano et al., 2023 state that generating the human labels for the Atari example required about 5 hours of human labor. Furthermore, they indicate that the cost of training the RLHF was about 1-GPU day, and the cost of computing resources and human labor were about equal. This suggests that there would be diminishing returns from generating more samples to train the reward function for this example.

The Christiano et al. paper has results for Atari games in which RLHF and the human-supervised approach fail to reach the reward obtained with RL (e.g., for the game breakout). They also have results for Atari games in which the human-supervised or RLHF approaches reached a higher reward than RL. Unfortunately, very little information is provided in the paper to explain these results. Furthermore, because many real-world problems have more dimensions than an Atari game, the findings in their paper might not scale.

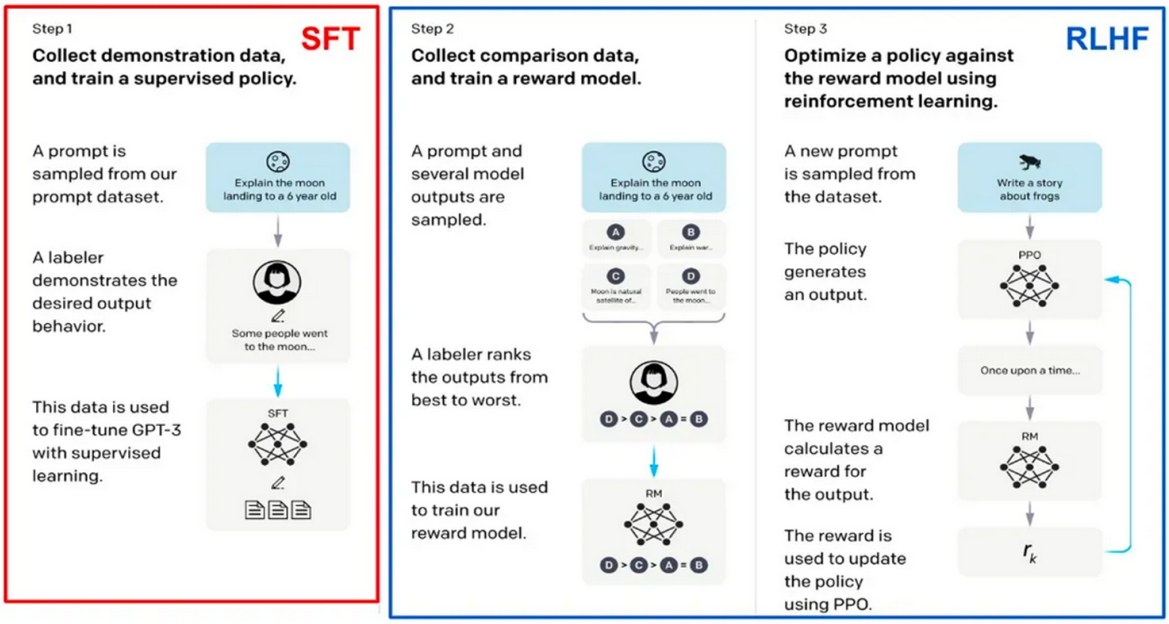
Reinforcement Learning from Human Feedback to Align Behaviors with Human Preferences and Values

In the previous section, we provided an example in which RLHF could be used for RL without access to a true reward function and suggested that this can improve sample efficiency compared with using human labels. In this section, we provide an example showing how RLHF can be used as an alternative (or in addition to) SFT for aligning an LLM with human preferences in a summarization task. The summaries generated using the RLHF approach are preferred by human judges to those generated using SFT alone, even with a smaller model for RLHF.

First, we provide a high-level description of SFT and RLHF as applied to LLMs. The red box on the left in Figure D.5 provides an overview of SFT. Assume you have an LLM such as GPT-3 that has been pretrained, and you want to optimize the resulting policy for downstream tasks. In particular, suppose you want to optimize the LLM policy for generating human reference summaries and align it with human preferences. Retraining the policy from scratch could be cost prohibitive. The process for SFT is to generate a dataset of prompts for summaries, have human labelers demonstrate the desired response, and then fine-tune the policy using supervised learning. The blue box on the right in Figure D.5 provides an overview of RLHF for this application. Several outputs of the pretrained LLM (or, alternatively, the pretrained LLM that has had SFT applied to it) are generated for each sample prompt. Human labelers rank the outputs from best to worst. The samples of prompts and ranked outputs are then used to train a reward model. Next, policy is updated using RL with the trained reward model. Typically, a proximity policy optimization approach is used for the RL, which optimizes the policy for the designated task while ensuring the optimized policy does not move too far away from the original policy. For instance, one approach is to generate a prompt from the dataset and responses from the reward model and policy. Then, a measure known as the *Kullback-Liebler divergence* is assessed

over sequences of the tokens of the responses, and a gradient is used to update the weights of the policy to move them in the direction of the reward.[90]

Figure D.5. Overview of Supervised Fine-Tuning and Reinforcement Learning from Human Feedback as Applied to a Large Language Model



SOURCE: Reproduced from Cameron R. Wolfe, "Understanding and Using Supervised Fine-Tuning (SFT) for Language Models," *Deep (Learning) Focus* blog, September 11, 2023, adapting a figure from Ryan Lowe and Jan Leike, "Aligning Language Models to Follow Instructions," OpenAI, January 27, 2022.

Stiennon et al., 2022 surveyed human judges to learn their preferences for human reference summaries produced by a pretrained LLM.[91] They asked the judges to assess summaries from three sources: (1) an LLM with no SFT and no RLHF, 2) an LLM with SFT and no RLHF, and 3) an LLM with both SFT and RLHF. The LLM had been pretrained on a large corpus of text. The researchers used four different model sizes in each case: 1.3-billion parameters, 2.7-billion parameters, 6.7-billion parameters, and 12.9-billion parameters. The SFT was based on the Reddit too long; didn't read (TL;DR) dataset of summaries (which they note required 320-GPU days for the 6.7-billion parameter model). They trained a reward model based on human preferences between pairs of summaries that required thousands of hours of labeler labor. Human

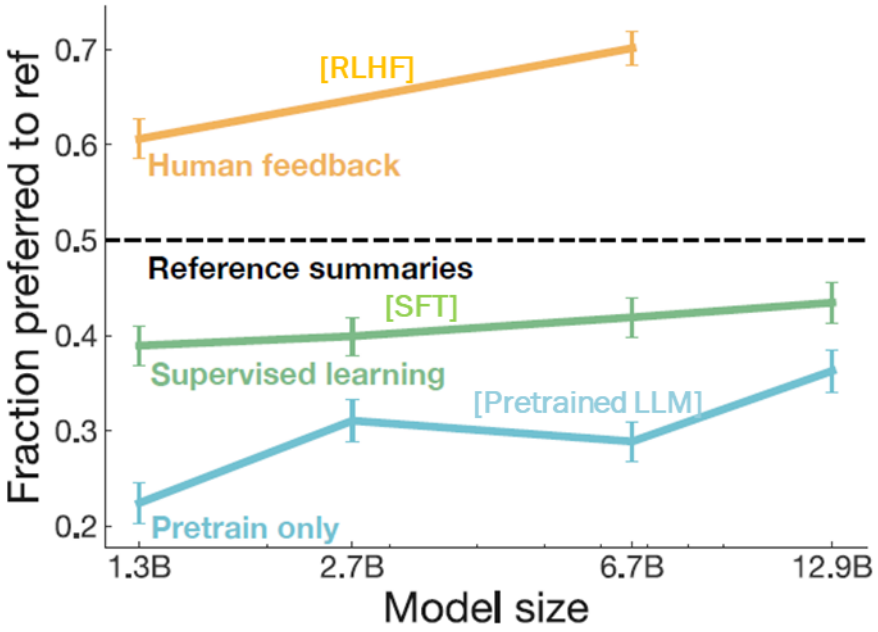
[90] For more details, see Nathan Lambert, Louis Castricato, Leandro von Werra, and Alex Havrilla, "Illustrating Reinforcement Learning from Human Feedback (RLHF)," Hugging Face, December 9, 2022.

[91] Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano, "Learning to Summarize from Human Feedback," arXiv, version 3, February 15, 2022.

judges were provided summaries from the pretrained LLM, both the SFT and RLHF variants, and asked to indicate their preferences.

Figure D.6 shows the results from Stiennon et al. The horizontal axis shows the model size. The vertical axis shows the fraction of summaries (averaged) that judges preferred from the pretrained LLM (cyan colored line), SFT (green colored line), and RLHF (orange colored line) variants. Note that their RLHF variant used the SFT variant as the baseline (that is, the RLHF variant had also undergone SFT). Apparent from the results is the fact that the authors assess only the RLHF variant for model sizes of 1.3- and 6.7-billion parameters. We observe from the results that the human judges had a strong preference for the RLHF-generated summaries compared with those generated from the SFT or pretrained LLM. In fact, summaries generated by the 1.3-billion parameter RLHF model were preferred to the summaries generated by the 12.9-billion parameter pretrained LLM and SFT models. These results show that RLHF improved the alignment of the LLM for the summary task, even when using a model that is an order of magnitude smaller.

Figure D.6. Human Judges Preferred Reinforcement Learning from Human Feedback Summaries, Even with Smaller Model Sizes



SOURCE: Reproduced from Stiennon et al., 2022, p. 2.

Other research suggests that, although RLHF might be very effective at aligning an LLM to human preferences, there could be performance trade-offs involved. In particular, Kirk et al.,

2024[92] define out-of-distribution (OOD) generalization as a metric on the reliability of a model outside the distribution of the training data. They define output diversity (OD) as a measure of the diversity of the output distribution in open-ended domains such as storytelling. The results of their research suggest that LLMs trained with RLHF improve OOD compared with SFT but at the expense of OD performance.

Summary

In sum, research suggests that RLHF algorithmic improvements

- include improved sample efficiency
- include improved performance in aligning policies with human preferences, even for smaller model sizes
- are application-specific and involve trade-offs
- require human labelers and compute resources.

How similar are these improvements to improvements from computer hardware advances? Improvements in sample efficiency and improvements to alignment with human preferences for smaller model sizes have some similarities to the speedups in compute that result from hardware improvements. But RLHF provides some additional, qualitative improvements to LLMs that do not have a clear analog to hardware advances.

[92] Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu, "Understanding the Effects of RLHF on LLM Generalisation and Diversity," arXiv, version 3, February 19, 2024.

Abbreviations

AI	artificial intelligence
GPT-3	Generative Pretrained Transformer 3
FLOP	floating point operation
FLOP/s	floating point operations per second
GPU	graphics processing unit
LLM	large language model
MoE	mixture-of-experts
RL	reinforcement learning
RLHF	reinforcement learning from human feedback
SFT	supervised fine-tuning

About the Authors

Carter C. Price is a senior mathematician at RAND and a professor at Pardee RAND Graduate School. His research focuses on empirical analysis of the relationship between economic growth and inequality and recent trends in economic distributions. He holds a Ph.D. in applied mathematics.

Brien Alkire is a senior operations researcher at RAND and serves as director of PAF Fellows in RAND Project AIR FORCE. His research focuses on space, cyber and electronic warfare technology and strategy, command and control, and artificial intelligence. He holds a Ph.D. in electrical engineering.

Mohammad Ahmadi is a Ph.D. student in the technology applications and implications stream at Pardee RAND Graduate School and an assistant policy analyst at RAND. His research interests include artificial intelligence, machine learning, Islamic terrorism, and homeland security. He has an M.P.S.A. in policy analysis.

References

Amodei, Dario, "On DeepSeek and Export Controls," blog, January 2025.

Anderson, E., Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, "LAPACK: A Portable Linear Algebra Library for High-Performance Computers," *Supercomputing '90: Proceedings of the 1990 ACM/IEEE Conference on Supercomputing*, May 1990.

Anthropic, "Constitutional AI: Harmlessness from AI Feedback," policy memo, December 15, 2022. As of February 28, 2025: https://www.anthropic.com/news/constitutional-ai-harmlessness-from-ai-feedback

Aschenbrenner, Leopold, *Situational Awareness: The Decade Ahead*, June 2024.

Bahdanau, Dzmitry, "The FLOPs Calculus of Language Model Training," Medium, January 9, 2022.

Boyd, Stephen, and Lieven Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

Boyd, Stephen, and Lieven Vandenberghe, *Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares*, Cambridge University Press, 2018.

Brown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al., "Language Models Are Few-Shot Learners," arXiv, version 4, July 22, 2020.

Budu, Emmanuella, "What Does Pre-Training a Neural Network Mean?" Baeldung, last updated February 13, 2025.

Burden, Richard L., and J. Douglas Faires, *Numerical Analysis*, 9th ed., Brooks/Cole, 2011.

Bureau of Industry and Security, "Commerce Strengthens Restrictions on Advanced Computing Semiconductors to Enhance Foundry Due Diligence and Prevent Diversion to PRC," Office of Congressional Affairs, January 15, 2025.

Christiano, Paul F., Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei, "Deep Reinforcement Learning from Human Preferences," arXiv, version 4, February 17, 2023.

Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to Algorithms*, 3rd ed., MIT Press, 2009.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, et al., "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning" arXiv, version 1, January 22, 2025.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, et al., "DeepSeek-V3 Technical Report," arXiv, version 1, December 27, 2024.

Dem'yanov, Vladimir F., and Leonid V. Vasil'ev, *Nondifferentiable Optimization*, Optimization Software Inc., 1985.

Executive Order 14110, "Safe, Secure, and Trustworthy Development and Use of Artificial Intelligence," Executive Office of the President, October 30, 2023.

"Frontier Remains No. 1 in the TOP500 but Aurora with Intel's Sapphire Rapids Chips Enters with a Half-Scale System at No. 2," Top500, undated. As of March 14, 2025: https://top500.org/news/frontier-remains-no-1-in-the-top500-but-aurora-with-intels-sapphire-rapids-chips-enters-with-a-half-scale-system-at-no-2/

Gamble, Chris, and Jim Gao, "Safety-First AI for Autonomous Data Center Cooling and Industrial Control," Google DeepMind, August 17, 2018. As of March 14, 2025: https://deepmind.google/discover/blog/safety-first-ai-for-autonomous-data-centre-cooling-and-industrial-control/

Google DeepMind, "AI Achieves Silver-Medal Standard Solving International Mathematical Olympiad Problems," July 25, 2024. As of March 4, 2025: https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/

Grace, Katja, "Algorithmic Progress in Six Domains," Machine Intelligence Research Institute, Technical Report 2013-3, December 9, 2013.

Gu, Albert, and Tri Dao, "Mamba: Linear-Time Sequence Modeling with Selective State Spaces," arXiv, version 1, December 1, 2023.

Haberman, Richard, "Infinite Domain Problems—Fourier Transform Solutions of Partial Differential Equations," in *Elementary Applied Partial Differential Equations with Fourier Series and Boundary Value Problems*, 2nd ed., Prentice-Hall, 1987.

He, Xu Owen, "Mixture of a Million Experts," arXiv, version 1, July 4, 2024.

Heim, Lennart, "FLOP for Quantity, FLOP/s for Performance," blog, *.xyz, April 14, 2023. As of March 3, 2025: https://blog.heim.xyz/flop-for-quantity-flop-s-for-performance/

Ho, Anson, Tamay Besiroglu, Ege Erdil, David Owen, Robi Rahman, Zifan Carl Guo, David Atkinson, Neil Thompson, and Jaime Sevilla, “Algorithmic Progress in Language Models,” arXiv, version 9, March 9, 2024.

Hoffmann, Jordan, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al., “Training Compute-Optimal Large Language Models,” arXiv, version 1, March 29, 2022.

Hu, Jian, Xibin Wu, Zilin Zhu, Xianyu, Weixun Wang, Dehao Zhang, and Yu Cao, “OpenRLHF: An Easy-to-Use, Scalable and High-Performance RLHF Framework,” arXiv, version 4, November 24, 2024.

Institute of Electrical and Electronics Engineers, “754-2008: IEEE Standard for Floating-Point Arithmetic,” August 29, 2008.

Kaplan, Jared, Sam McCandish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei, “Scaling Laws for Neural Language Models,” arXiv, version 1, January 23, 2020.

Kirk, Robert, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu, “Understanding the Effects of RLHF on LLM Generalisation and Diversity,” arXiv, version 3, February 19, 2024.

Lambert, Nathan, Louis Castricato, Leandro von Werra, and Alex Havrilla, “Illustrating Reinforcement Learning from Human Feedback (RLHF),” Hugging Face, December 9, 2022.

Lee, Harrison, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, et al., “RLAIF: Scaling Reinforcement Learning from Human Feedback with AI Feedback,” arXiv, version 2, December 1, 2023.

Liu, Ziming, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark, “KAN: Kolmogorov–Arnold Networks,” arXiv, version 2, May 2, 2024.

Lowe, Ryan, and Jan Leike, “Aligning Language Models to Follow Instructions,” OpenAI, January 27, 2022.

Ma, Shuming, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei, “The Era of 1-Bit LLMs: All Large Language Models Are in 1.58 Bits,” arXiv, version 1, February 27, 2024.

Maslej, Nestor, Loredana Fattorini, Raymond Perrault, Vanessa Parli, Anka Reuel, Erik Brynjolfsson, John Etchemendy, Katrina Ligett, Terah Lyons, James Manyika, Juan Carlos Niebles, Yoav Shoham, Russell Wald, and Jack Clark, *The AI Index 2024 Annual Report*, AI Index Steering Committee, Institute for Human-Centered AI, Stanford University, April 2024.

Metz, Cade, and Meaghan Tobin, "How Chinese A.I. Start-Up DeepSeek Is Competing with Silicon Valley Giants," *New York Times*, January 23, 2025.

Mocz, Lucia, "Performance Bottlenecks in Deploying LLMs—a Primer for ML Researchers," Medium, May 10, 2023.

Nazir, Anam, and Ze Wang, "A Comprehensive Survey of ChatGPT: Advancements, Applications, Prospects, and Challenges," *Meta-Radiology*, Vol. 1, No. 2, 2023.

NVIDIA, "NVIDIA A100 Tensor Core GPU," data sheet, June 2021. As of March 14, 2025: https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-us-nvidia-1758950-r4-web.pdf

Pestourie, Raphaël, Youssef Mroueh, Chris Rackauckas, Payel Das, and Steven G. Johnson, "Physics-Enhanced Deep Surrogates for Partial Differential Equations," *Nature Machine Intelligence*, Vol. 5, December 2023.

Ralston, Anthony, and Philip Rabinowitz, *A First Course in Numerical Analysis*, 2nd ed., Dover, 2001.

Russell, Stuart, and Peter Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed., Pearson Education Limited, 2014.

Saad, Yousef, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, 1996.

Sayood, Khalid, *Introduction to Data Compression*, Morgan Kaufmann Publishers, 1996.

Scharre, Paul, "Future-Proofing Frontier AI Regulation," Center for a New American Security, March 13, 2024.

Shalev-Shwartz, Shai, and Shai Ben-David, *Understanding Machine Learning: From Theory to Algorithms*, Cambridge University Press, 2014.

Sherry, Yash, and Neil C. Thompson, "How Fast Do Algorithms Improve?" *Proceedings of the IEEE*, Vol. 109, No. 11, November 1, 2021.

Sherry, Yash Mohan, and Neil C. Thompson, "Measuring the Pace of Innovation: Evidence From Algorithms," conference paper, SI 2020 IT and Digitization, National Bureau of Economic Research, July 2020.

The Shining, dir. Stanley Kubrick, Warner Bros., 1980.

Shivakumar, Sujai, Charles Wessner, and Thomas Howell, *Balancing the Ledger: Export Controls on U.S. Chip Technology to China*, Center for Strategic and International Studies, February 21, 2024.

Shnitzer, Tal, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin, "Large Language Model Routing with Benchmark Datasets," arXiv, version 1, September 27, 2023.

Sorscher, Ben, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos, "Beyond Neural Scaling Laws: Beating Power Law Scaling via Data Pruning," *Advances in Neural Information Processing Systems*, proceedings of the 36th International Conference on Neural Information Processing Systems, 2022.

Stiennon, Nisan, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano, "Learning to Summarize from Human Feedback," arXiv, version 3, February 15, 2022.

Stoer, J. and R. Bulirsch, *Introduction to Numerical Analysis*, 3rd ed., Springer, 2002.

Stryker, Cole, and Eda Kavlakoglu, "What Is Artificial Intelligence (AI)?" IBM, August 9, 2024. As of March 17, 2025: https://www.ibm.com/think/topics/artificial-intelligence

Trinh, Trieu H., Yuhuai Wu, Quoc V. Le, He He, and Thang Luong, "Solving Olympiad Geometry Without Human Demonstrations," *Nature*, Vol. 625, January 18, 2024.

Villalobos, Pablo, Anson Ho, Jaime Sevilla, Tamay Besiroglu, Lennard Heim, and Marius Hobbhahn, "Will We Run Out of Data? Limits of LLM Scaling Based on Human-Generated Data," arXiv, version 2, June 4, 2024.

Wang, Zhilin, Yi Dong, Olivier Delalleau, Jiaqi Zeng, Gerald Shen, Daniel Egert, Jimmy J. Zhang, Makesh Narsimhan Sreedhar, and Oleksii Kuchaiev. "HelpSteer2: Open-Source Dataset for Training Top-Performing Reward Models." arXiv, version 1, June 12, 2024.

Wolfe, Cameron R., "Understanding and Using Supervised Fine-Tuning (SFT) for Language Models," *Deep (Learning) Focus* blog, September 11, 2023. As of March 3, 2025: https://cameronrwolfe.substack.com/p/understanding-and-using-supervised

Zeng, Zhiyuan, Qinyuan Cheng, Zhangyue Yin, Bo Wang, Shimin Li, Yunhua Zhou, Qipeng Guo, Xuanjing Huang, and Xipeng Qiu, "Scaling of Search and Learning: A Roadmap to Reproduce o1 from Reinforcement Learning Perspective," arXiv, version 1, December 18, 2024.