How Open Source Machine Learning Software Shapes AI

by

Max Langenkamp

B.S. Computer Science and Engineering, Massachusetts Institute of Technology (2021)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2022

© Max Langenkamp, MMXXII. All rights reserved.

The author hereby grants to MIT permission to reproduce and to distribute publicly paper and electronic copies of this thesis document in whole or in part in any medium now known or hereafter created.

Author . . . .
Department of Electrical Engineering and Computer Science
May 6, 2022

Certified by . . . .
Dylan Hadfield-Menell
Assistant Professor of Artificial Intelligence and Decision Making
Thesis Supervisor

Accepted by . . . .
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

How Open Source Machine Learning Software Shapes AI

by

Max Langenkamp

Submitted to the Department of Electrical Engineering and Computer Science
on May 6, 2022, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

If we want a future where AI serves a plurality of interests, then we should pay attention to the factors that drive its success. While others have studied the importance of data, hardware, and models in directing the trajectory of AI, I argue that open source software is a neglected factor shaping AI as a discipline. I start with the observation that almost all AI research and applications are built on machine learning open source software (MLOSS). This thesis presents four contributions. First, it quantifies the outsized impact of MLOSS by using Github contributions data. By contrasting the costs of MLOSS and its economic benefits, I find that the average dollar of MLOSS investment corresponds to at least $100 of global economic value created, corresponding to $30B of economic value created this year. Second, I leverage interviews with AI researchers and developers to develop a causal model of the effect of open sourcing on economic value. I argue that open sourcing creates value through three primary mechanisms: standardization of MLOSS tools, increased experimentation in AI research, and creation of commuities. Third, I analyze the various incentives behind MLOSS by examining three key factors: business strategy, sociotechnical factors, and ideological motivations. In the last section, I explore how MLOSS may help us understand the future of AI and make a number of probabilistic predictions. I intend this thesis to be useful for technologists and academics who want to analyze and critique AI, and policymakers who want to better understand and regulate AI systems.

Thesis Supervisor: Dylan Hadfield-Menell
Title: Assistant Professor of Artificial Intelligence and Decision Making

Acknowledgments

I owe a deep debt of gratitude to many people. Two in particular have stood by me the most in this intellectual voyage. Daniel Yue did an equal amount of the research on the first half of my thesis (which later became a joint paper). For his good cheer, technical persistence, and openness to my unorthodox and occasionally incoherent ideas, I am grateful. I thank Dylan Hadfield-Menell for being able willing to supervise me on during his hectic first two semesters at MIT. Without him, this thesis would have sprawled into rambling irrelevance.

Before I met Dylan or Daniel, conversations with Tim Hwang and Teddy Collins kindled the ember of my thesis idea. I was remotely working for the Center for Security and Emerging Technology and they were generous enough with their time. Although my collaboration with Tim never panned out, it was the excitement of our initial meetings that drove me to apply for the grant funding necessary to pursue my idea. Accordingly, I must thank Open Philanthropy for providing the funding required to pursue this master's thesis. Of the several audiences I made the case for my research to, it was the officers at Open Philanthropy who made it possible for me to pursue this thesis in the first place.

Dozens of one-off conversations with people: Nadia Asparouhova for her commiseration in pursuing independent research, Helen Toner for her sympathetic ear, Eamon Duede for pointing out that my initial research question was far too broad, Professor Michael Cafarella for comments on history of open source software, Cailean Osborne for his suggestion to use a consistent yet specific enough acronym for my topic of study, Adam Salisbury for his enthusiasm at my research topic.

For early conversations about governance, I thank Seth Frey, Charles Schweik, and Madeyln Sanfilippo. Although I pivoted from that angle of inquiry, I learned a lot during those conversations.

For being open to meandering early-stage conversations, I have several friends I'd like to thank: Tuomas Oikarinen, Nikhil Murthyo, Tan Zhi Xuan, Serena Booth. You all were great.

Contents

List of Figures

List of Tables

The fig tree is pollinated only by the insect Blastophaga grossorun. *The larva of the insect lives in the ovary of the fig tree, and there it gets its food. The tree and the insect are thus heavily interdependent: the tree cannot reproduce without the insect; the insect cannot eat without the tree; together they constitute not only a viable but a productive and thriving partnership. This cooperative "living together in intimate association, or even close union, of two dissimilar organisms" is called symbiosis.*

....

Man-computer symbiosis is probably not the ultimate paradigm for complex technological systems. It seems entirely possible that, in due course, electronic or chemical "machines" will outdo the human brain in most of the functions we now consider exclusively within its province... There will nevertheless be a fairly long interim during which the main intellectual advances will be made by men and computers working together in intimate association... [the years] may be 10 or 500, but those years should be intellectually the most creative and exciting in the history of mankind.

Man-Computer Symbiosis, JCR Licklider (1950)

Chapter 1

Introduction

Interest in artificial intelligence (AI) has exploded over the past decade. Now, even casual consumers interact daily with AI systems. This is often attributed to advances in data, compute, and algorithms [18]. These factors are sometimes described as inputs to the so-called ‘AI production function’. In this thesis, I consider a neglected factor: machine learning open source software (MLOSS). MLOSS is ubiquitous in both research and production. However, it has received comparatively little attention in the literature. I will examine how MLOSS shapes AI practice, examine its influences, and finally explore its implications on the near term future. I conclude that MLOSS is a powerful point of intervention for shaping AI research and a phenomenon that merits further examination.

My argument contains four parts:

1. MLOSS tools play an outsized role in the creation of economic value
2. MLOSS drives AI impact through standardization, experimentation, and community creation
3. A combination of business incentives, sociotechnical factors, and ideological motivations shape MLOSS
4. MLOSS will continue to shape AI both by reinforcing deep learning and by facilitating standardization of other AI capabilities

Overall, this thesis deepens understanding of how MLOSS impacts the AI ecosystem. I offer four contributions, corresponding to the four parts of the argument. First, I estimate the economic impact of MLOSS tools. I argue that the large cost-benefit ratio suggests MLOSS is a useful point of intervention for policymakers. Second, using qualitative interview data, I propose that MLOSS shapes AI development through standardization, experimentation, and community creation. Next I explore the question 'what shapes MLOSS?' through the lens of the business strategy that accompanies its creation, the sociotechnical factors that shape the boundaries of what tools can be created, and the ideological motivations associated with freely sharing tools. Finally, I broadly examine the implications of my findings on the future of AI research and development and present a list of predictions.

1.1 Background

To start my discussion, I introduce some key terms and background context that are crucial for developing my argument in the following sections. In particular, I will define machine learning open source software (MLOSS) and provide a brief overview of MLOSS history.

1.2 Defining Machine Learning Open Source Software (MLOSS)

Following prior work, I refer to AI as "the use of digital technology to create systems capable of performing tasks commonly thought to require intelligence" and will follow the common practice of using the terms 'machine learning' (ML) and 'artificial intelligence' (AI) interchangeably[10, 22]. I refer to machine learning open source software as computer software released under an open source license that is designed specifically with machine learning use cases in mind. This includes software ranging from frameworks (e.g. PyTorch and Pyro) to 'all-in-one' packages (e.g. scikit-learn) to model development tools (e.g. TensorBoard). It does not include software such as the interactive computing tool Jupyter Notebook which, although often used by machine learning practitioners, was not specifically built to accommodate machine learning.

1.3 A Brief History of MLOSS

I review the history of MLOSS to highlight that the phenomena is new, ubiquitous, and increasingly supported by industry efforts.

In my perspective, the history of open source machine learning can be grouped into three eras, punctuated by two critical events: the 2012 ImageNet competition and the release of TensorFlow in 2015.

- *Phase 1: Grassroots Efforts (pre 2012).* Prior to 2012, there were few large and well maintained MLOSS projects [54]. Andrew Ng's famous Introduction to Deep Learning course was originally taught in MATLAB, a closed source language. There were some more targeted ML frameworks such as OpenNN and Torch (which later formed the foundation for PyTorch). However, the packages were either very general or difficult to install and use, and lacked features such as GPU support [55].

- *Phase 2: The Rise of Frameworks (2012-2015).* In 2012, a deep convolutional neural network later known as AlexNet handily won the ImageNet competition, attracting significant attention within academic and certain industry communities [40]. Subsequently, a wave of frameworks emerged from various academic research labs, including Chainer, Theano and Caffe. Open source software played an important role in the creation of these frameworks — for instance, the creators of Caffe directly cite the decision to open source AlexNet as inspiration for their framework [60]. Simultaneously, there were a number of efforts within industry to develop private frameworks, such as Google's DistBelief. Frameworks for alternative approaches, such as Stan, appear and start to gain prominence.

- *Phase 3: Industrialization of AI Research (2015-present).* In 2015, Google's decision to open source TensorFlow changed the landscape in a number of ways. First, by deploying over 200 engineers on the project, TensorFlow provided a package that possessed a quality of engineering far above that of other frameworks [56]. This led other companies to release competitor frameworks, such as Amazon's MXNet, Microsoft's CNTK, and (later) Facebook's PyTorch. In this phase, we also witness the increasing prominence of frameworks for alternative AI methods. Gen, a probabilistic programming package within the programming language Julia, is released and begins to be used by researchers.

Now, open source technologies are ubiquitous in modern ML applications. Consider a hypothetical document-processing company. In their stack, they may leverage Detectron2 (an open source object detection model) programmed in PyTorch (an open source framework) developed in Python (an open source language), originally trained on COCO (an open source data set) and coded in a Jupyter notebook (an open source development environment) [51, 43]. This is not the case in many other technical fields, such as animation graphics or sound engineering.

MLOSS is used in the vast majority of ML applications. Most organizations implement machine learning methods through cloud providers like AWS Sagemaker or GCP's AI platform. Within those platforms, the predominant way of implementing models is to build them via existing libraries such as Google's TensorFlow or Facebook/Meta's PyTorch [14]. Open source software is even more central to AI research. Paperswithcode, a community resource for practitioners to follow AI research, shows that the vast majority of publicly available research code is written using open source frameworks [58]. This matches data from interviews with AI researchers in both academia and industry, where every single practitioner acknowledged the core role of open source tools to their research process.

Before discussing related work, I'd like to add a caveat. While this thesis refers to *machine learning* open source software, much of the focus (outside of the chapter on MLOSS and economic value) is on *deep learning* open source software. This is the case for a couple of reasons. First, deep learning tools — especially frameworks such as PyTorch and TensorFlow — are the most popular MLOSS tools ever created. Accordingly, they have played an outsized role in shaping AI research. Second, although the primary examples provided are from deep learning, I have striven to insure that the effects of MLOSS discussed are not unique to deep learning. This is most true of chapter four — on standardization, experimentation, and community creation.

1.4 Related Work

Studies on Open Source. Prior approaches have estimated the economic contribution of open source provided some reassurance that a systematic assessment of value was possible. Nadia Asparouhova's book 'Working in Public' was a significant inspiration early on.

Factors Shaping AI. Hitherto, much of the discussion of the factors that shape AI has focused on the role of computation, with some consideration of the role of algorithms and data. For instance, Dafoe (2018) suggests that "[p]lausibly the key inputs to AI are computing power (compute), talent, data, insight, and money."[18] Hwang (2018) examines how the hardware supply chain shapes machine learning development. Rosenfeld (2019) and Hestness (2017) examine the how dataset size relates to model accuracy in AI. Both are part of a growing literature that aims to more explicitly model the relationships between inputs and predictive error in AI. To the best of our knowledge, however, there have been no detailed examinations of how open source software shapes AI development.

Critical Examination of AI. Many scholars have provided insightful examinations about the trajectory and potential pitfalls of AI. The work of Dotan and Milli (2020) [23] as well as that of Lake et al. (2017) [41] particularly helpful for situating deep learning as a paradigm within AI more generally. The commentary in Jo et al. (2020) [38] about the neglected role of data tooling in current AI research has been similarly helpful.

Chapter 2

MLOSS Tools Play an Outsized Role in the Creation of Economic Value

Inspired by the observation that open source software is ubiquitous in AI applications, this chapter will argue that MLOSS tools play an outsized role in the creation of global economic value. Regardless of whether one cares about economic value for its own sake, this suggests that MLOSS significantly shapes AI's impact on society both now and in the future, if not only because economic incentives heavily drive development.

First I'll begin by estimating the cost of MLOSS tools based on Github activity. Next, I'll construct a cost-benefit estimate for AI using prior estimates of AI's economic value. I argue that the benefit-to-cost ratio of MLOSS tools is at least 100-to-1. In other words, for every dollar invested in MLOSS tools, we should expect at least \$100 is created within the AI ecosystem. By this logic, we can conservatively estimate that the global value created by MLOSS will be \$30B, a contribution that we should expect to continue to grow in absolute terms even if no new MLOSS packages are produced.

2.1 The Cost of MLOSS Development

I begin with an estimate of the cost of annual development of the core MLOSS repositories. Whereas this exercise would be impossible for most closed-source technologies, MLOSS development cost can be estimated from contribution data found on the public repositories on Github.

2.1.1 Data Collection

Defining a Comprehensive List of MLOSS Tools. I collected 146 actively maintained MLOSS tools by drawing largely from the list of tools compiled by Chip Huyen, a notable ML researcher and developer [35]. A small majority of the tools were from large technology companies such as Facebook/Meta (PyTorch), and growing startups such as HuggingFace (Transformers). It draws from sources ranging from the Linux Foundation's AI Tools, FirstMark's Data and AI Landscape, and suggestions from the AI community via Twitter. To validate the dataset, I asked a number of practitioners in the community and examined other surveys. The practitioners asked suggested two more tools that were missing, and the Kaggle survey of developers reflected my intuition that the vast majority of developers use the same small number of tools [39]. For this reason, although this list should not be considered authoritative, I believe that the tools not included will have only a negligible affect on the estimates.

Sampling Contribution Data from MLOSS Repositories. I scrape contribution data from each repository's Github contributions page, which pre-aggregates contributions data at the Contributor-Period level, where the 'period of interest' is user-defined (here the period was chosen to be two weeks). I then randomly select 5x two-week periods from the history of each MLOSS repository. For each of those Periods, I record the number of commits and lines modified (added or deleted) for each contributing user in that period.

The result is 3,895 observations of Contributor-Period level data. For example, during the October 16-30th 2017 period, Soumith Chintala (a PyTorch co-founder) contributed 10 commits to the PyTorch repository corresponding to 10 commits and 338 lines modified. I also aggregate to the Repo-Period level, a total of 670 observations. Another example: during the March 16-30th 2018 period, the Scikit-Learn repository had a total of 4 contributors adding 9 commits of 2134 lines modified.

Summarizing the Data. In Figure 1, I present histograms of the number of commits and lines-modified at both the Contributor-Period (Left) and Repo-Period (Right) level. The y-axis shows the number of commits (or lines modified), and the x-axis shows the log-scaled count of observations corresponding to those values. For visualization purposes, I collapse value observations above the 99% quantile to the 99% quantile (aka winsorize at the 99% level).

These resulting distributions seem reasonable. The median Contributor-Period corresponds to 2 commits of 100 lines of code modified. The median active repository receives a commit about 15 times by 3 users in a given two-week period, corresponding to 600 lines of code. However, I also observe the data is right-skewed. Therefore, while these distributions show that my results are not outlier-driven, there is some notable degree of inequality in contributions across MLOSS repositories.

2.1.2 Cost Estimation

I now exploit the dataset to estimate the cost of these MLOSS repositories. Let a "unit" be either a contributor, a commit, or a line modified. I first estimate cost-per-unit, and then estimate the total number of units-per-year over the list of MLOSS repositories. By multiplying these estimates, we can estimate the cost-per-year of MLOSS development.

Cost-Per-Unit. I estimate this using a reference organization with known developer salaries such as PyTorch (Facebook) or TensorFlow (Google), taken from levels.fyi, a salary information website [26, 27]. I assume an average wage of $300,000 a year, corresponding to the senior engineer level (e.g. L4 or E4). This is conservative, as Google and Facebook likely pay more than other MLOSS organizations. For Contributors, this salary is the cost-per-unit. For commits and lines modified, I scale the average contributor-period-level data from the two-week period to the yearly level, and use my known salaries to compute a cost-per-unit estimate, according to the following equation:

$$\frac{\$Cost}{Unit} = \left(User \cdot PeriodAVG \left(\frac{\#Units}{User \cdot Period} \times \frac{26 Periods}{Year} \right) \times \frac{User \cdot Year}{\$Salary} \right)^{-1}$$

Units-Per-Year. I aggregate across the entire set of MLOSS repositories to estimate the number of units per year. For Contributors, I assume that the number of yearly-active contributors is the same as the average number of active contributors in the periods that I observe. This reflects the logic that not all engineers who ever contribute to an open source project are working on it full-time. Therefore, even if the observed periods do not capture all contributors at an organization, I argue that the average number of active contributors is representative of the organizations investment in MLOSS in general. For commits and lines modify, I simply take the average amount of units over the observed periods and scale that to the yearly level.

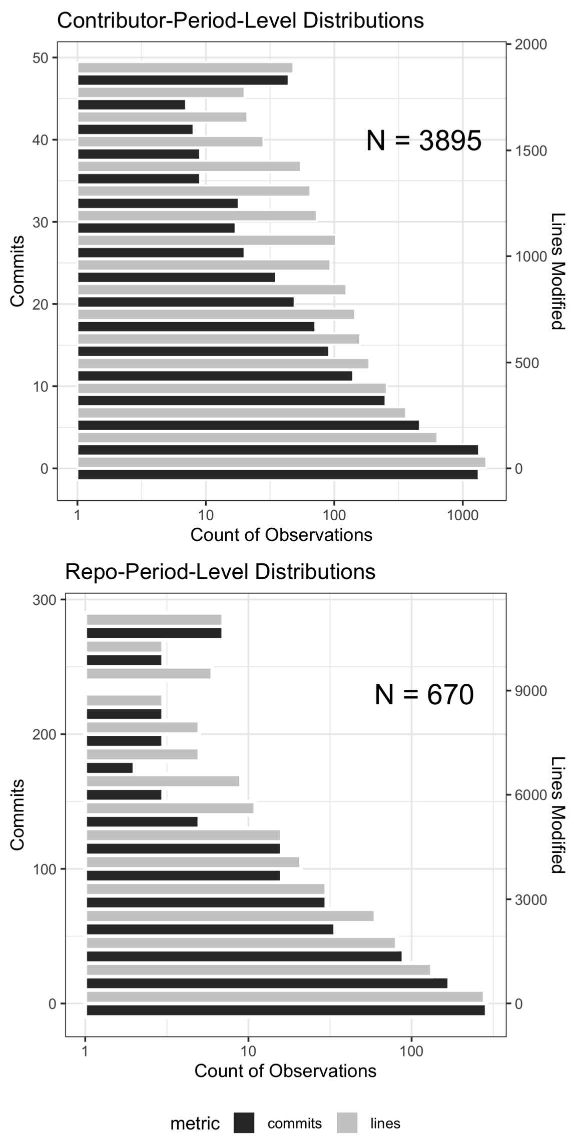


Figure 2-1: Histogram Plots for Commits and Lines-Modified, at Contributor-Period (Top) and Repository-Period (Bottom) levels.

$$\frac{Units}{Year} = \sum_{Repo}\left[PeriodAVG\left(\sum_{User}\frac{\#Units}{User \cdot Period}\right)\right] \times \frac{Period}{2Weeks} \times \frac{52 \cdot Weeks}{Year}$$

Sensitivity Considerations. I estimate total system cost using three different units (Contributors, Commits, and Lines Modified) to ensure that my estimates are not driven by a weak assumption about how contribution practices from large projects extrapolate to the rest of the project. Extrapolating via contributors assumes that different contributors are doing similar work across repositories. By contrast, commits and lines modified present alternative ways of weighting and extrapolating costs to other repos, where I assume that different commits or lines modified are comparable (even if Contributors are not comparable).

Table 2.1.2 presents the cost estimates for each unit. I find that the cost of MLOSS ranges between \$100-\$300MM per year.

Unit	**Ref. Repo**	**Cost/Unit**	**Units/Year**	**Cost/Year**
Commits	pytorch	$2.43K$	$121.83K$	$295.47M$
Contributors	pytorch	$300.00K$	779.00	$233.70M$
Lines	tensorflow	34.09	$5.44M$	$185.56M$
Lines	pytorch	34.08	$5.44M$	$185.53M$
Commits	tensorflow	986.04	$121.83K$	$120.13M$

Table 2.1: Estimated Annual Cost of MLOSS Tools

2.1.3 Limitations to Cost Estimation

While I attempted to make the list as comprehensive as possible, it's possible that I am missing significant MLOSS projects that contribute significantly to economic value creation and costs in ways that I am missing. Furthermore, given my limited sampling, the estimates may also be high in variance. Despite these concerns, I still believe these estimates capture the correct first-order approximation of costs. First, as argued above, there is significant inequality in MLOSS usage, and the expert-compiled list covers the most important tools with respect to usage. Second, the estimates of this methodology can be easily extended via further sampling to mitigate concerns around sampling noise.

2.2 The Benefits of AI and the role of MLOSS in Economic Value Creation

I now contrast the estimated costs of MLOSS with the benefits of AI to global economic value creation in order to argue that MLOSS plays an outsized role.

2.2.1 Economic Benefits of AI

Rather than developing our own estimate, I briefly summarize AI economic benefit estimates from McKinsey and PWC. Each of these organizations estimates that AI will add roughly $3-5 trillion USD to the global economy in 2022 [45, 49]. To arrive at these estimates, these organizations break down the channels through which AI creates and destroys value. AI creates global economic value because it lowers the cost of existing processes and also drives product and service innovation. Furthermore, it does this across a broad range of industries, due to its broad applicability as a general-purpose technology [31]. These organizations delve into various use-cases for AI in order to micro-found their assumptions about value benefits. The results in this thesis do not rely on the specific numbers in the reports, as the methodology used is somewhat opaque. Instead, these estimates provide a rough order of magnitude estimate of the economic contribution of AI. One reassuring factor is that both estimates are within an order of magnitude.

2.2.2 Attributing Economic Benefit to MLOSS

While it's hard to estimate precisely how much value can be attributed to MLOSS, one heuristic that has been proposed is the 70-20-10 rule, which argues that AI value creation comes from investments in Processes, Data/Technologies, and Algorithms, in those proportions [8]. For our purposes, I assume that MLOSS only contributes to the algorithms portion of investment. I further assume that the algorithm value is mostly attributable to MLOSS, based on the observation (described in Section 2) that MLOSS tools are ubiquitous and are the primary means through which models are distributed in practice.

Supposing that the consulting firms over-estimated the global value of AI by a factor of 10, this still attributes 1% of value created to MLOSS tools, or roughly $30B/year value created, corresponding to a benefit-cost ratio of over 100-to-1.

2.2.3 Interpreting the Economic Costs and Benefits of MLOSS

This estimated annual value creation (\$30B/year) and benefit-cost ratio (100:1) are very large – few other classes of technology can match this level of productivity. By comparison, the OECD estimates that open source software in Europe generates value at a 4:1 ratio [24]. Furthermore, because MLOSS is a public good, as the usage of AI scales globally, MLOSS will provide increasing returns to scale – which will only serve to increase the benefit-cost ratio.

Nevertheless, I think these estimates are reasonable and are consistent with other known estimates found in the literature. Even after these conservative estimates, which provide an upper bound on cost and a lower bound on benefits, there is a 100:1 benefit-to-cost ratio. This is consistent with Greenstein and Nagle's estimate of the economic contributions of Apache as between \$2B and \$12B in 2012 [32].

Although the exact value attributable to MLOSS is unclear, this analysis suggests that it plays a outsized role in global economic value creation. Given that MLOSS tooling is ubiquitous and plays a major role in value creation, it is a fruitful arena for fostering AI development. In order to do so, however, we need an understanding of the mechanisms through which MLOSS shapes AI. I'll explore this in the next section.

Chapter 3

MLOSS Drives AI Impact Through Standardization, Experimentation, and Community Creation

In this section, I develop a causal model of how MLOSS creates value and shapes the AI research ecosystem.

Before discussing the methodology and findings, however, I want to preemptively address a question: *how is MLOSS different from more general open source software?*

This question requires more attention than I can give it here. Moreover, this an open area of research that ought to be investigated more. First, many of the effects discussed below (standardization and community creation) apply equally to OSS. However, machine learning has distinct affordances from software engineering more generally. This shapes the effect of the software in ways that are underexplored and difficult to immediately articulate. Here are a some preliminary thoughts:

1. Machine learning open source software is less mature than well-established open source software.

The most studied open source software are projects like Debian and the Apache web server. Not only are these packages over a decade older than today's most popular MLOSS tools, their use cases have been clearly established. Web servers and operating systems were once an active area of research but their conceptual frameworks have been long established and the scope of their tasks is clearly defined. Machine learning is comparatively amorphous and actively being shaped by researchers.

2. Machine learning depends heavily on a wide range of capabilities.

As I will explore in Chapter 4, the current approach to machine learning depends heavily on different resources. Compute and data are the most obvious resources.

ID	Institutional Experience	Role	AI Fields
R1	Big Tech	PyTorch Dev	Frameworks Compilers
R2	Big Tech	TensorFlow Dev	Frameworks
R3	University Startup (Cybersecurity)	PhD Researcher	Audio
R4	University Big Tech	PhD Researcher	Robotics RL
R5	University Big Tech Startup (BioTech)	PhD Researcher	RL
R6	University Big Tech Startup (Translation)	PhD Researcher	NLP
R7	Venture Capital Startup (AV)	Investor AI Engineer	Robotics AV
R8	Startup (BioTech)	AI Engineer	Biophysics

Table 3.1: **Overview of Formal Interview Subjects' Background**. In selecting the interview subjects, I focused on finding people with a variety of difference experiences.

This means that in practice MLOSS subdivides into several submodules that each solve a specific problem for ML. Also, the primacy of large matrices — typically in the form of pretrained weights — require novel forms of software.

3.1 Methodology

3.1.1 Interviewees

I selected AI researchers and developers as subjects largely through convenience and theoretical sampling [11]. During my selection of interview subjects, I focused on ensuring that the interviews covered experiences from both industry and academia over a variety of different ML projects. Overall, I conducted 23 interviews: 8 formal interviews for an average of 50 minutes each, and 15 informal interviews with other AI researchers. The background and experiences of the formal interview subjects is listed in Table 3.1.1. In the next sections, to preserve the privacy of interviewees, I will be referring to individuals with fabricated identifiers (e.g. 'R3') as identified below.

3.1.2 Interview Structure

I followed a general interview structure:

- How did you first get introduced to machine learning?
- What have been your most recent machine learning projects?
- What institutional contexts did you work in, and what tools did you use?
- What are the main technologies that you depend on for your work? How do they fit into your workflow?
- Have you ever used research code from another researcher's project? Why? What was the process for using it like?
- What was the last time you used a new model or technique? What was your process for getting up to speed on it?
- Have you ever shared your own code / tools? Why? What was the process of preparing it like?

Where possible, I asked interviewees to expand on points of interest. All interviews had notes written within 24-hours of the interview. I promised confidentiality and received permission to digitally record the formal interviews, allowing us to transcribe them. In total, this produced 150 pages of interview transcripts and notes.

3.1.3 Archival Materials

To form a historical perspective, I examined a variety of sources. These included materials ranging from the PyTorch five-year review to discussion on the EleutherAI community Discord to materials from Stanford's CS230 [50, 57, 21].

3.1.4 Data Analysis

I iterated between information collection and analysis to generate a theory grounded in data [30]. I use two lenses of analysis:

1. *Thematic analysis*, where the categories are established using interview and archival data;
2. *Theoretical analysis*, where I examine how open sourcing software leads to the effect, and how the effect generates economic value.

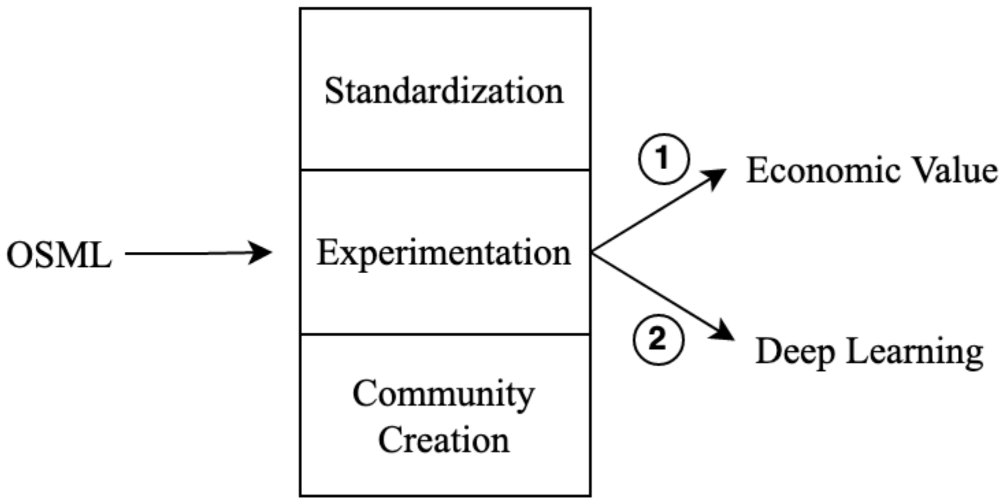


Figure 3-1: (1) I find that MLOSS produces economic value and shapes the AI ecosystem through three primary effects: standardization, experimentation, and community creation. (2) In Chapter 4, I will explore how these same factors reinforce the dominant paradigm of deep learning.

Out of the many effects I encountered over the interviews, I identified three core effects of open-sourcing projects: standardizing interfaces, enabling experimentation, and creating communities.

3.2 Findings

I organize the findings into three distinct effects: Standardization, Experimentation, and Community Creation (show in Figure 3-1).

3.2.1 Effect 1: Standardizing interfaces

One major effect I observed from the interviews and my own experience with open source tools was standardization — the widespread adoption of a single technology or technique among both users and tool-makers. *Thematic Analysis.* I observed standardization at three major levels of analysis.

Programming language and frameworks. Frameworks like PyTorch and TensorFlow provide the core primitives that are used by researchers and developers to construct and train machine learning models. As described in section 2, after a period of divergence in framework development, MLOSS frameworks have significantly consolidated over the past five years.

Our interviewees' experiences substantiate this general trend in framework consolidation; several interviewees started out by working in either older frameworks (R3 worked with Theano) or directly via array-based methods (R5), not a single one of the subjects regularly works with a framework outside of PyTorch, TensorFlow, or JAX today. R3 pointed out that, despite its early prevalence in the ML community, nobody uses MATLAB any more. All researchers that I interviewed emphasized the benefits of using the same framework in terms of their ability to replicate and build on the community's code, as well as sharing their ideas with other researchers.

Model Types. I noted that several of the interviewees converged on working with large neural networks. Traditionally, these models would be hard for resource-constrained researchers to leverage, but open source efforts have made many models widely available. R6: "HuggingFace, for example,... made so many things a lot easier and continues to for a lot of people in NLP who work on large models. I don't really know what I would do without HuggingFace.... there's a sense that I'm missing a big chunk of the field if I'm not working on big models at all."

In order to come to consensus on large models, the research community also needs open source datasets to establish benchmarks. For example, the ImageNet challenge was enabled by the public ImageNet data set. This was essential for establishing the importance of deep convolutional neural networks in 2012. In providing benchmarks, public datasets have also facilitated the movement away from less well-suited models such as Markov Chains.

User Experience. The interviews demonstrated convergence in user experience.

A particularly prominent example that emerged from the interviews was the convergence of frameworks on eager execution over graph-based execution. Eager execution lets developers print values while running the model. In contrast, graph-based approaches require users to insert placeholder variables in a computational graph.

Several interviewees noted that TensorFlow's default graph execution was counter-intuitive and made it harder to learn as a beginner, which led to them preferring PyTorch. R6 notes: TensorFlow had such a weird model — you can't print your graph because there's no values, it's just the abstract graph. So I remember I struggled for a long time in the early days. What I've observed is TensorFlow trying to add more of that back into their framework to imitate PyTorch. So, now, you can do things like eager execution.

At this point, there are few differences between the top frameworks. R5 notes "In terms of the specific frameworks themselves... my personal opinion is that there's not a huge difference between all of them – JAX, PyTorch, TensorFlow, etc.”

Beyond user interfaces, openness leads to greater integration between tools. R1 notes that the development of PyTorch XLA, which enables PyTorch (Facebook-based software) compatibility with Tensor-Processing-Units (Google-based hardware), was led by the Google research team [52].

Theoretical Analysis. One explanation for the relationship between open source and standardization is the concept of technology certification, described in Lerner and Tirole 2005 [42]. Open sourcing a technology is much easier than forming consensus initially. However, because these technologies are free to use, high quality projects gain large user bases and complementary products like documentation and integrations with other technology emerge (a cross-side network effect). This attracts new users, who are now able to share and collaborate with other users on the platform (a direct network effect). By contrast, a closed-source technology presents a friction that makes it difficult for all users to adopt and agree on initially, which prevents the accruing of these network effects.

The core economic benefit of standardization is the creation of natural interoperability — where distinct technological systems are able to exchange services and interact in a useful way. Interoperability improves economic outcomes by lowering the costs necessary to train and transfer skills across domains, as well as adding value through the ability for technologies to work with each other.

3.2.2 Effect 2: Enabling experimentation

Open source projects shaped the interviewees’ project preferences, helped them work faster, and gave them new ways of thinking about problems.

Thematic Analysis. I organize the concept of experimentation into three sub-concepts: project choice, development speed-up, and novel conceptualization.

Project Choice. The availability of open source code had a dramatic effect on the projects that the interviewees worked on.

The availability of code was a primary motivation for some research projects. From R4:

> “If PyTorch or an equivalent didn’t exist, and I was looking at these papers that were, ‘hey here’s this neural network that can do this task and we don’t know how it works, but we’re seeing great performance,’ I might

just let that be. Whereas because those tools are there and I can just grab the person's implementation of this, I can very quickly start to probe it to see if the network is actually learning things and test my explainability methods with it. It's just so much easier for me to work on those types of problems, so they become more attractive." R5 stated "My master's thesis was playing around with adversarial examples... looking at neural-tangent-kernel-based models... If 'neural-tangents' (a key library) disappeared, that would be very annoying. I literally don't know how to do neural tangent stuff without 'neural-tangents', and I would literally have to go back into the papers... which would be pretty bad."

When code was not available or did not work, it changed the scope of the project or whether someone would pursue it. R4 said "I was trying to build a system where the first step was to implement these old interactive RL Frameworks. I was having some problems with that, so it definitely was going really slowly and was off putting to me. So I ended up diverting the project. Instead of implementing those things myself, I ended up doing more of a literature review."

Beyond research, open source enables startups that work on AI problems with limited resources to exist. R8 argues that if their startup "had to reimplement a deep-learning framework from scratch, that would not be feasible, because we would need to hire people who really understand compilers and CUDA and things like that. If it was closed source, we would pay for it, maybe... but you definitely need to have ML at a certain stage of maturity to allow [our startup] to do what we do."

Faster Research and Development. Most obviously, open source projects speed up development of new AI applications.

Most subjects felt that, although they conceptually understood open source frameworks such as TensorFlow and PyTorch, these tools vitally lowered the friction and increased the speed of specifying model architectures and training neural networks. R4: "For me, the turnaround could be as short as eight weeks. [Because of existing code] it's very, very quick to get a prototype, and then you're running your experiments. The cycle is super rapid because of the availability of these tools."

MLOSS infrastructure also enables startups to iterate quickly. R7 notes that "these open source tools let you rapidly prototype and iterate, which is important in the early stages of a company, when they're figuring out what their product is"

Novel Conceptualization. The most subtle form of increased experimentation comes from the way that open source projects change how AI developers conceptualize problems.

One way this shows up is in how developers conceive of what new models and applications can be built. R1 recalls a fellow researcher's comment: "there's no way I would have thought of these ideas if it wasn't for using PyTorch". R3 also notes, "In PyTorch you can have 'if' statements and all sorts of weird things that are not normally part of neural networks, and it back-propagates through them easily. So you have more freedom to experiment with novel ideas and structures because of that." Because of advances in Torch in particular, new kinds of network architectures such as Tree-LSTMs are possible [62].

Theoretical Analysis. Open-sourcing a project enables experimentation because it significantly lowers both the economic and knowledge barriers between project creators and consumers. Especially in the case of MLOSS, given available code, the barriers to reproducing a paper are very low. For example, R3 notes that "if you read a biology paper, there's no way you're going to, in an afternoon, reproduce the results... But in machine learning, that's pretty doable." By lowering the barriers to entry, open source encourages researchers to enter fields based on the quality of their ideas rather than their prior knowledge-base or institutional circumstance. This model is similar to the one presented in Murray et al. (2016), which finds that openness enables researchers to join new fields quickly and opportunistically work on relevant problems in the context of biology research[47].

Enabling experimentation creates economic value because it leads to the discovery of a variety of machine learning models that enable AI to solve a broad range of problems. This enables AI to solve a diverse breadth of use-cases across a variety of problem domains. Furthermore, it allows for the most effective techniques of different AI subfields to be transferred over rapidly to new subfields – for example the recent transfer of Transformer architectures from NLP problems over to computer-vision problems.

3.2.3 Effect 3: Community Creation

Perhaps the most under-discussed mechanism that interviewees referenced is the effect of open source on community creation. By community, I mean a space for both technology contributors and users to interact – with common digital spaces today being Github, Reddit, and Discord.

Thematic Analysis. Open-sourcing a project leads creators to be more in-touch with users, encourages users to contribute tools themselves, and inspires the creation of related educational materials that make it easier for others to get involved.

Increased Feedback. Open sourcing projects enables greater feedback on the project, which improves its design. Soumith Chintala, a co-creator of PyTorch, emphasized the role of the openness of the community in helping to direct the prioritization of PyTorch and making it a great user experience. "[Soumith] read the entire volume of information that [his] community produced – github issues, forum posts, slack messages, twitter posts, reddit and hackernews comments. It was an incredibly useful signal..." [13].

This effect extends beyond the focal project – R2 noted that, because of the open-nature of the feedback, PyTorch had an advantage as "second mover". PyTorch learned from the mistakes of the previous TensorFlow framework.

For researchers, open-sourcing code enables their ideas to be more closely validated. R4 notes "If I make mistakes, I want somebody else to publish a paper saying, 'hey, you got this wrong...' I want this pursuit of truth and openness is the best way to get there."

Users Become Contributors. Open source machine learning software encourages and makes possible broad involvement. From R4:

> "If [research] required you to build your own system... we would see many fewer people participating in this field." R5 observes a cultural aspect associated with open source projects not shared by closed source projects: "Open source incentivizes people to play around with the frameworks. I don't see people say, 'Here's some cool thing I did in Matlab, come check it out.' But people will say 'PyTorch is a cool framework, and here's something I made while messing around in PyTorch.' And they'll share it in the blog post..."

Furthermore, MLOSS encourages unlikely participants to contribute to projects. Consider the EleutherAI community, an open source community that grows and coordinates primarily through their Discord Server. One undergrad who contributes to the project writes

> "One day during the pandemic summer of 2020, I found myself in this strange dream-like place, a community of international Machine Learning flaneurs who somehow became convinced that they could actually make history. At first, I thought it would just be a fun place to discuss new

> AI developments. But I soon discovered that yeah, these people are serious about their ambitions, and more thrillingly they actually would like to have me on board! As it turns out, the fact that Machine Learning engineers despise JavaScript (while still needing it) become [sic] my entry ticket to some of the coolest projects I ever worked on."

Improved Educational Materials and Settings. Open source tools inspire the community to develop associated educational material to extend the reach of the user base. All of the interviewees entered the field through openly available education materials on AI – ranging from Nielsen's online book on Neural Networks to Andrew Ng's CS231 course at Stanford. R3 notes that

> "I studied [the Nielsen textbook] on my own time and got very interested because I actually realized that this whole thing is not as complicated as I thought it would be. I could actually run the example and eventually started building some of my own things."

Open source communities incentivize the creation of high-impact educational settings. R2 comments

> "we [Google] escalated from (just) teaching university students in the US to going to these road shows, because we also obviously wanted to teach people in all sorts of different emerging markets. TensorFlow is an international platform and is adopted by people everywhere, so... we teach them colab, introducing them to colab, helping them connect to the TPU or GPU accelerator so that they can run a model in their browser now they don't have to worry about actually installing it."

Theoretical Analysis. The formation of open source communities has been studied extensively in the literature, with prior explanations focusing on desires to reciprocate in response to someone else's effort, to have impact, or to gain a reputation in a way that is useful [24]. I think all of these mechanisms are likely at play in the open source machine-learning setting. However, I note one final mechanism for community involvement – participating in these communities makes the products themselves better, in a way that benefits the user.

Community creation creates economic value because it lowers the cost of using these tools and increases the number of available applications. By encouraging community members to become well-versed in the available tools and models, open-sourcing also makes it easier for firms to find the necessary labor needed to implement machine learning models that meet their organizational needs.

In summary, open sourcing creates economic value through three distinct intermediate mechanisms: standardization, experimentation, and community creation. These concepts are represented in Figure 3-1. In the next section, I will explore the factors that shape MLOSS itself.

Chapter 4

What shapes MLOSS tools?

So far we've considered the value of MLOSS and a simple account of how it shapes the ecosystem. Yet the reader might naturally be drawn to another question: what are the factors that shape MLOSS? In this chapter, I address this question by focusing on three most relevant types of factors: business incentives, sociotechnical forces, and ideological motivations.

Before discussing the factors themselves, it's helpful to distinguish between three types of MLOSS producers. The first are large technology companies like Microsoft, Amazon, and Google. They invest the most in providing MLOSS, and many of the most popular open source tools are supported by them. The second are smaller technology companies ('startups') who tend to provide tools more targeted towards the deployment of machine learning models. One prominent example in this category is Hugging Face, the startup provides pretrained NLP models and associated infrastructure. The final producers are nonprofits and academia. Nonprofits like NumFOCUS sponsored projects like Scikit-learn and NumPy, while academic groups such as the MIT probabilistic computing group produce tools like the probabilistic framework Gen.

These boundaries are often fuzzy in practice. The Linux Foundation, for instance, is a nonprofit, but many of its projects like the Open Neural Network Exchange were sponsored by several large technology companies. Similarly, smaller technology companies may contribute to projects primarily housed at a large technology company. Nevertheless, these distinctions are useful because in practice most work on a given project comes from one of these three categories. The interested reader can examine the associated collected open source dataset to evaluate this for themselves.

In the first section, I'll address how business incentives motivate MLOSS for large and smaller technology companies. In the next section, I'll examine the sociotechnical factors shaping all MLOSS. In the third section, I'll briefly explore the distinct ideological motivations behind MLOSS.

4.1 Business incentives for MLOSS

In general, there are many different reasons a given company might fund developers to work on an open source software project. It is particularly helpful to separate the incentives facing the largest technology companies from those facing smaller startups.

I will discuss three effects that create incentives for large technology companies (e.g. Facebook, Amazon, Google, Baidu) to produce MLOSS: standard shaping, generating goodwill, and enhancing applications.

Prior to discussing each of these reasons, it's worth noting that the barriers for big technology companies to provide MLOSS are very low. Firstly, the current production cost of even the most popular open source tools is insignificant in comparison to the research and development budget at these companies. For instance, Facebook spent \$1.9 billion on research and development in 2017, whereas we estimate they spent no more than \$10 million, or 0.5% of their RD budget, supporting PyTorch. Furthermore, this vastly overstates the cost of providing the software. Large companies that heavily use machine learning tools like PyTorch need centralized code for building models, whether or not they intend to release such code publicly. Providing public support and documentation requires considerable time and effort, but is a fraction of the effort that is required to build and maintain the tools in the first place.

Having considered the context, let's examine the business incentives that motivate investment in MLOSS.

4.1.1 Providing MLOSS leads to easier talent acquisition and collaborations

Over the course of many conversations with dozens of ML practitioners, it was obvious that open source tool providers are held in high regard in the community. Researchers and practitioners, when prompted, are quick to express gratitude especially for the well-engineered tools. This goodwill translates directly to prestige for the tool developers. One developer working on a popular framework told me that their primary motivation for working in a large technology company was to work on their open source framework.

4.1.2 MLOSS enables standard setting, research adoption, and other forms of indirect control over AI development

It may be initially puzzling to note that many of the companies that spend the most providing MLOSS (Google, Amazon, Facebook) also aggressively patent their research. Google, for instance, spends tens of millions of dollars a year maintaining patents. Yet it seems counter-intuitive that a company that wants software to be freely available also purchase expensive patents.

These strategies are in fact complementary. The patents prevent other companies from suing the patent holders and allow the patent holders a degree of control over their use of the technology. Open-sourcing a project increases adoption rates, increasing the likelihood that other developers will use tools from within the project sponsor's ecosystem.

Additionally, MLOSS lowers the friction for academic researchers to use the tools. In a field where a large fraction of research is still within academic institutions, open sourcing software reduces the friction between academia and industry. This means that more talented academic researchers will go to work for industry research labs, but also that the research labs are able to rapidly adopt (and spread) new machine learning methods.

The power over standards that sponsoring an open source ML project gives is subtle and should not be overstated. Norms within the open source community are strong and highly critical of any attempt to restrict usage. Facebook, for instance, notably failed to add a restrictive licensing stipulation to its popular package React because of pressure from developers[53]. Similarly, the core decisions within most open source frameworks come from developers themselves, rather than a mandate stemming from the company's business strategy. This can be seen by looking at the history of decision making on the open source forums.

Nonetheless, it is clear that sponsoring a critical tool gives the sponsor significant indirect power within the ecosystem. Decisions like which hardware to prioritize compatibility with, or which applications to provide support for, can favor the interest of the project sponsor. At the very least, providing a critical component of infrastructure ensures that key pieces of future research will be compatible with their existing system.

4.1.3 Providing MLOSS enhances the value of existing capabilities by incorporating community feedback

MLOSS increases the value of existing AI applications within companies. By soliciting improvements from tens of thousands of practitioners, TensorFlow became less buggy, more usable and general. This in turn, could improve products that use TensorFlow like Dialogflow, Google's platform for building chatbots, as well as services like Youtube's auto-transcribe and Google translate.

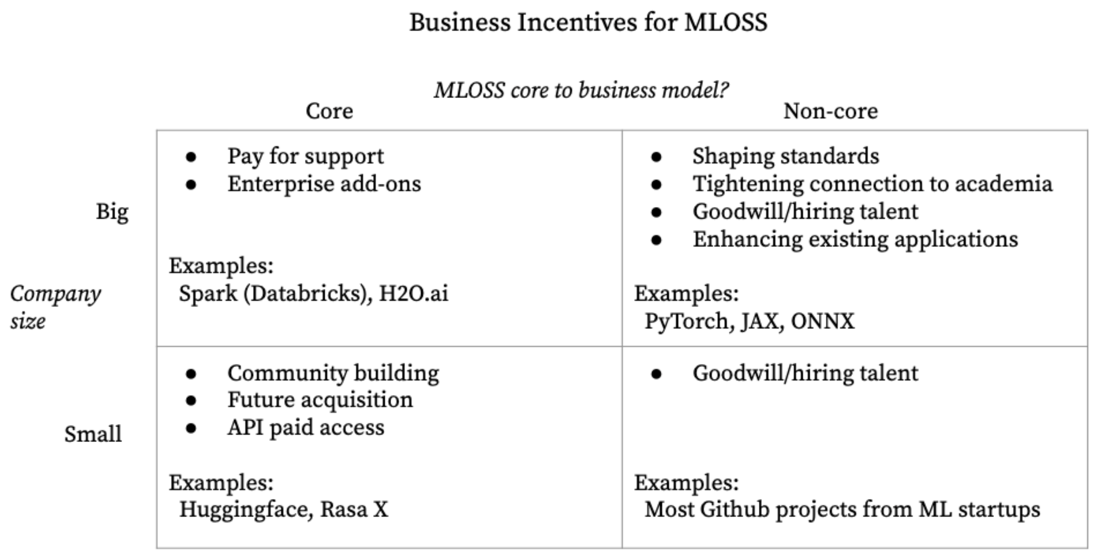
More abstractly, however, providing MLOSS increases the value of the sponsor's AI capabilities. If we conceive of the value of AI as a bundle of capabilities (e.g. data, tooling, compute, algorithms), then commoditizing the tooling can increase the value of the bundle by increasing demand for AI as a whole. Concretely, Amazon's compute platform becomes much more valuable when high quality frameworks lead to twice the number of ML practitioners.

4.1.4 Startups provide MLOSS for community, talent, and complementary services

Moving on from large technology companies, smaller technology companies and startups appear to be motivated primarily by goodwill and increased demand for complementary services. Companies like the natural language processing company Hugging Face, which provides the most popular package to import and use language models, find it much easier to hire talented people because there is much goodwill and prestige associated with the providers of heavily used MLOSS. Similarly, Hugging Face's free and popular language models have majorly increased demand for both their ML infrastructure and consulting services. Similarly, Rasa, a startup providing an open source AI chatbot, has a monetized platform for enterprise developers to customize their chatbot. In most cases, MLOSS startups are seeking to grow their community, hire talent, and sell complementary services rather than attempt to shape the future of AI or strengthen ties with academia. Figure 4-1 provides a more specific breakdown of incentives as they differ across organizations.

4.2 Sociotechnical forces shaping MLOSS

There are many forces that shape MLOSS that are by no means distinct to ML or even OSS. Three general forces are helpful for understanding the evolution of MLOSS:



Business Incentives for MLOSS

MLOSS core to business model?

		Core	Non-core
Company size	Big	• Pay for support • Enterprise add-ons Examples: Spark (Databricks), H2O.ai	• Shaping standards • Tightening connection to academia • Goodwill/hiring talent • Enhancing existing applications Examples: PyTorch, JAX, ONNX
	Small	• Community building • Future acquisition • API paid access Examples: Huggingface, Rasa X	• Goodwill/hiring talent Examples: Most Github projects from ML startups

Figure 4-1: A 2x2 of the business incentives for MLOSS

standardization, usability, research trends.

4.2.1 Some code wants to be standardized

Some programs lend themselves more easily to standard interfaces than others. This simple fact has major bearing on what software becomes MLOSS. Consider the two following tasks: matrix differentiation and dataset manipulation.

Matrix differentiation is a process that is notoriously tedious to implement yet absolutely crucial for any artificial neural network to data. These two factors — that it was time consuming to implement from scratch yet necessary for any work with deep learning — meant it was one of the first procedures in MLOSS to become standardized. Automatic differentiation was one of the first and largest contributions of frameworks such as Theano, Torch, and TensorFlow.

Dataset manipulation, in contrast, is not very difficult and highly idiosyncratic. Despite the fact that everyone has to do some data manipulation while training, there is no standard tool for examining dataset quality across a variety of dataset types.

A priori, one should expect the programs most easy to standardize to be the first to be open sourced.

4.2.2 The most usable software tends to win

Another significant technological factor that shapes the growth of MLOSS is usability. The usability of a tool effectively acts as a selection pressure, favoring the tools with the most intuitive user interface. Notably, PyTorch's imperative-style specification of neural networks was eventually adopted by TensorFlow. This is despite the initial lack of support for imperative approaches to production systems. The competition between TensorFlow's graph-based execution and PyTorch's eager execution recalls the famous Lisp vs C, or 'the right thing' vs 'worse is better'[28]. Just as the simpler, less complete, 'worse' C programming language prevailed in adoption over the more complicated, complete, 'right' Lisp language, so too did PyTorch's simpler approach prevail over TensorFlow's more efficient graph-based model.

4.2.3 Focus in MLOSS mirrors the attention of the community

MLOSS is also significantly shaped by the prevailing paradigm within machine learning. That the predominant model type for machine learning is currently deep learning means that more MLOSS projects will naturally be created to address problems in deep learning, rather than an alternate paradigm such as probabilistic programming or automated planning.

Of course, the prevailing paradigm within machine learning is itself a product of a large number of different factors. The state of hardware, the most prestigious benchmarks, the commercial applicability, are all important in shaping the dominant approach within machine learning. For a thoughtful consideration of this topic, I point the reader to Dotan and Milli (2019)[23].

4.3 Ideological motivations shaping MLOSS

A final, critical, aspect of MLOSS development is the role of ideology. Most of the key figures in MLOSS are motivated by particular visions of the world. These visions may be religious in nature or the consequence of strongly held values. These values tend to either be about helping improve the experience of other fellow developers, or else furthering the state of AI.

The OSS researcher Nadia Asparouhova talks about the surprising presence of religious belief in OSS [1]. SQLite, the most popular open source database engine on the internet, famously has a Code of Ethics page that includes several biblically inspired commandments[3]. Travis Oliphant, the creator of NumPy, one of the most commonly used libraries in Python, tells the story of creating NumPy as an act of public service against the wishes of his advisors and peers at Brigham Young University[4].

Other developers believe in the pure good of furthering AI. In a podcast interview, when asked for the reason Facebook sponsors PyTorch, Soumith Chintala explains that "we have a single point agenda at [Facebook AI Research], which is to solve AI" which involves empowering others to work on the problem [2]. The implication seems to be that 'solving AI' would lead to enormous upside for society, whether by allowing new drugs to be discovered or by proving new theorems. Similarly, H2O.ai and Hugging Face both refer to 'democratizing AI' as a central motivation for open sourcing their products.

4.4 Evaluating the theory

One question that may appear to the reader is "why did MLOSS seem to gain significant popularity when it did?" Only four years prior, there had been a paper by several machine learning researchers decrying the lack of open source code in machine learning[54].

While there are many factors that can explain the rise of MLOSS, two sociotechnical forces provide the most helpful lens: technological capacity and attention.

The first sociotechnical factor I'd point to is technological capacity. In order for deep learning open source software to have gained popularity, researchers needed both hardware and standardized interfaces for deep learning. In the last couple of decades, the commercial growth of video games led to the creation of GPUs, which researchers, in turn, realized could be utilized to calculate model gradients. At the same time, early attempts to build neural networks, such as Collobert's Torch library [16] provided a standard interface for ML models. Thus, the building blocks of automatic differentiation and templates for specifying neural network models existed. This was a crucial resource for subsequent projects to draw from.

The second sociotechnical factor is the captured attention of the community. The winning entry in the 2012 ImageNet competition ignited the imagination of academia and industry alike. Academics like Yangqing Jia, creator of Caffe, were inspired by Alex Krizhevsky's open source cuda-convnet to build their own frameworks. At a similar point, people like Google's Jeff Dean realized that machine learning would provide enormous commercial value in the near future. Google Brain was formed and built an initially private neural network framework called Distbelief [20]. Distbelief eventually came to inform TensorFlow, which then sparked the ‘industrialization’ of open source machine learning software.

Chapter 5

Discussion: MLOSS and the Future of AI

This final section discusses the implications of MLOSS on the near term trajectory of AI. I will begin by examining MLOSS in the context of different machine learning paradigms, then present a model of how MLOSS shapes deep learning production. After arguing for the emergence of two trends — a shift away from frameworks and the rising importance of data tools — I will briefly consider the risks on a 10 year time horizon associated with emerging MLOSS and present a list of probabilistic predictions.

5.1 MLOSS reinforces the deep learning paradigm

I'll begin with a quick overview of different paradigms in AI, and then present two examples of how current MLOSS tools reinforce deep learning. My aim in this section is to illustrate how MLOSS tools may inadvertently contribute to an 'AI research monoculture', whereby there are strong incentives for the majority of funders and AI researchers to work within the domain of large neural networks.

5.1.1 On Different AI Paradigms

Beyond deep learning, other paradigms in AI research include probabilistic machine learning [29], rule-based expert systems [19], and automated planning [25].

These different approaches are good for different tasks. For a concrete example, consider the problem of verifying the code for flight software. The stakes of buggy flight software are very high and require a large degree of certainty. In such domains, engineers opt to use theorem provers, which are closely linked to automated planning, to formally verify that the software is free of bugs. This is a task that deep learning algorithms are ill-suited for because they cannot provide formal guarantees.

If we want to build AI systems that augment, rather than compete with humans, it is important to consider the tasks and interactions that other paradigms lend themselves to. It is not easy to assess the relative merits of each paradigm today, let alone predict which will succeed. However, several of the researchers expressed a sense that the 'hype' around deep learning is inducing a myopia and paradoxically degrading the standards of scholarship, even as empirical results incrementally improve. Lipton and Steinhardt (2018) point to something similar with their exploration on the lack of explanation within much recent deep learning work [44]. I find it likely that broadening focus from deep learning would engender a greater array of AI systems, increasing the diversity of needs that a particular AI system can meet. I will discuss the associated risks in the final section.

5.1.2 Better support for open source deep-learning tools reinforces deep learning

The two most popular open source tools in deep learning and in automated planning are, respectively, PyTorch and FastDownward [33]. As a tool developed largely by the Facebook AI Research term, PyTorch is incredibly well supported. Installation only requires a single line of code in the terminal, and completes in a few minutes. PyTorch runs on many types of hardware (with training on GPUs and CPUs) and operating systems (including iOS and Android). A user can typically resolve technical issues with a single search engine query, which parses tens of thousands of posts and active users. There are dozens of helpful snippets of code detailing how to, for instance, debug tensors with mismatched dimensions.

In contrast, consider FastDownward. Installation is non-trivial and requires basic knowledge of operating systems to handle downloading a compressed bundle of files and managing their installation manually. The project supports Linux, macOS, and Windows, but does not appear to have support for GPUs or more exotic operating systems. Furthermore, it's difficult to get immediate support if a user runs into technical issues: In most of the bug queries I tried, a straightforward search via a search engine did not yield answers, and I had to turn to their custom forum. I do not mean to disparage FastDownward, which appears to be a well maintained project with clear documentation. My point here is that it is very difficult for a much smaller community and project to match the support for one that is so much better resourced in terms of engineers.

The quality of available tools affects how researchers choose the paradigm they work in. Given the friction of using a tool like FastDownward, which is harder to install, find example code for, and debug, a researcher may be naturally inclined to pursue deep learning tasks. Because of the ease of experimenting with deep learning frameworks and the documentation produced by its communities, she is more likely to work on problems in deep learning. These may be tasks like investigating the outputs of large language models, which involve large amounts of data, rather than, for instance 2D scene navigation, where problem formulation and algorithmic construction is more important [17].

5.1.3 MLOSS tools, in driving industry adoption, have shifted researchers' economic incentives

Economic incentives may also predispose a prospective researcher towards deep learning – a downstream effect of the larger community associated with that paradigm. Because deep learning tools are so easy to use, reliable, and supported by a large user community, many companies can use deep learning. Companies have en masse begun to apply deep learning via tools like TensorFlow, which has resulted in many more jobs in industry for experts in deep learning than in other paradigms [6]. There is also much higher demand in industry for jobs that involve deep learning. On Indeed.com, there were over 15k jobs including the description 'deep learning'. For 'probabilistic programming' and 'automated planning', there were 40 and 8 jobs, respectively [36, 37].

Having examined the role of MLOSS in favoring certain paradigms over others, I'd like to examine how it shapes deep learning.

5.2 MLOSS and the deep-learning production function

In order to examine MLOSS' effect on deep learning, it will help to contextualize the role of MLOSS alongside the other factors in the 'AI production function'. The production function, as first introduced by Cobb and Douglas [15], conceives of abstract variables, such as capital and raw materials, that parameterize an abstract function mapping the respective inputs to outputs. Within the literature on AI governance, Dafoe introduced the notion of the 'AI production function'[18], suggesting the "inputs of compute, talent, data, investment, time, and indicators such as prior progress and achievements". Dafoe's discussion of 'AI progress', along with discussions found in similar work, attempts to be agnostic to the particular paradigm of AI. In practice, this equates 'deep learning' and 'AI' — a move that limits the precision of the conversation. In this section, I have instead opted to more explicitly refer to 'deep learning'.

While a production function can help to explicitly separate the factors that influence deep-learning development, it also has its limits. The production function doesn't account for the possibility of shared dependencies of the factors of production, and hides crucial contextual information about each factor. To understand these factors and their relationships, it is helpful to visualize an ordered dependency graph of capabilities that address different user needs. This has the benefit of accounting for shared capabilities within each factor (e.g. intermediate model representations depend on both compute infrastructure and on the MLOSS framework) and also of providing context in the relative maturity of each factor.

Figure 5-1 presents factors as capabilities required for a machine learning developer to build a predictive model. This technique is known as Wardley mapping, a business strategy tool invented by the researcher Simon Wardley in 2005[61].

The purpose of this figure is twofold. First, Wardley mapping is a helpful tool to understand the factors that shape the development of deep learning, and subsequently, AI. Second, the map conveys two emerging themes in deep-learning research and development: the commodification of frameworks and the rising importance of data.

The map centers around the basic need of an ML developer — 'building a machine-learning model' — and makes explicit the primary capabilities required. In this case, the primary capabilities are compute, data, and frameworks. Each of these capabilities have their own capabilities and are ordered on the y-axis based on how many other capabilities a given capability depends on. Note that, for simplicity, I have deliberately excluded the algorithms themselves as well as human capabilities.

The capabilities are then organized according to the their relative maturity. On the x-axis, the less developed, newer capabilities are put on the left while the the more standardized, well-defined capabilities are put on the right. As technologies develop, their capabilities become better defined and move to the right.

In organizing capabilities in this way, two intuitions are made explicit: the commodification of deep learning frameworks and the rising importance of data tooling. Let's explore these in more detail.

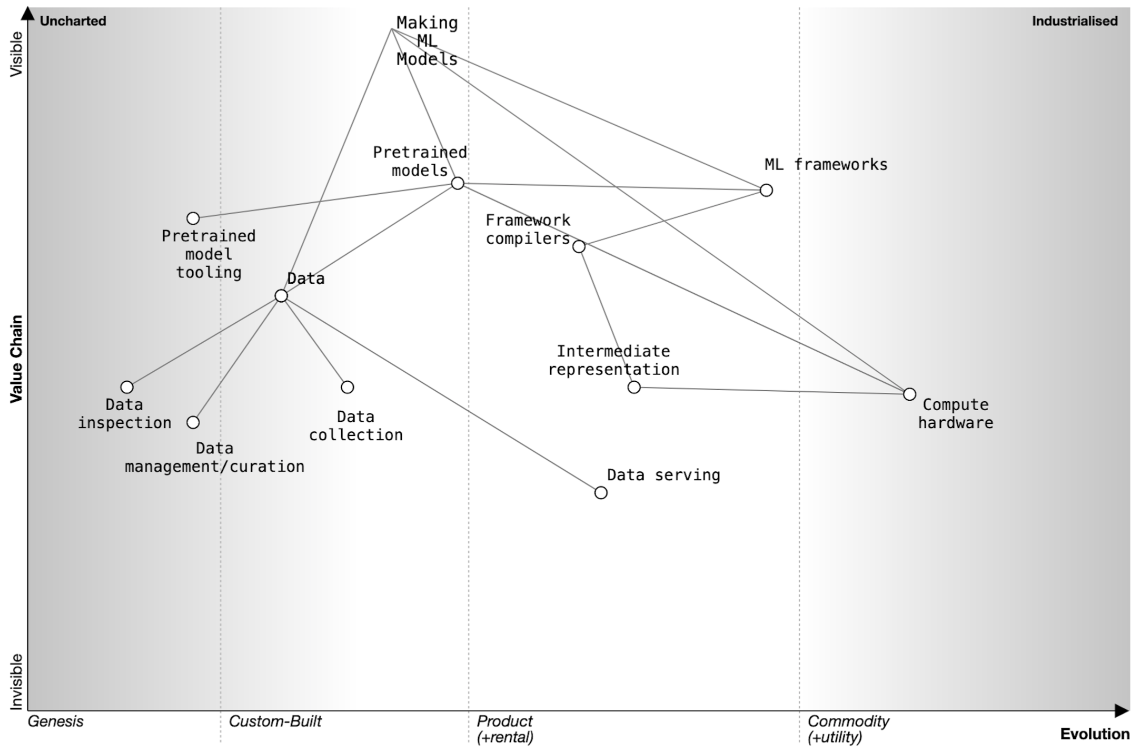


Figure 5-1: Wardley Map of Machine Learning

5.3 Attention shifts away from deep learning frameworks

Competition among deep learning frameworks has been long and fierce. Before PyTorch and TensorFlow, there was Chainer, Theano, Torch, and Caffe (among others). JAX, one of the most popular deep learning frameworks, has gained adoption among researchers over TensorFlow very recently. Deepmind, for instance, announced at the end of 2020 that it was adopting JAX as its primary framework [5]. Similarly, Huawei released their own deep learning framework MindSpore in March of 2020 and has gained significant adoption in China [46]. Given the continued activity within the framework space, one might wonder if we should expect development effort to continue to focus on deep learning.

I contend that competition among deep learning frameworks is largely over. Of all the capabilities in a deep learning system, none have received as much attention as the hardware for computation or the software for frameworks. In the early days of TensorFlow, it was difficult to express certain architectures (e.g. Tree RNNs) using the framework. Now, the practitioners I've talked to say that expressing models using frameworks is easy and no longer a bottleneck. Soumith Chintala, one of the creators of PyTorch, provides a suggestive quote: "With PyTorch and TensorFlow, you've seen the frameworks sort of converge... the next war is compilers for the frameworks — XLA, TVM, PyTorch has Glow, a lot of innovation is waiting to happen."

Chintala points to framework compilers as the next domain of competition, but where else might we expect attention? One area that I expect to gain significant traction is tooling for pretrained models. Many recent notable projects have iterated on large pretrained models. Github's Copilot , the large model designed to assist with Python code-writing, was a finetuned version of GPT-3[12]. At the most basic level, this would involve infrastructure for serving pretrained models. The current large language models are too large to run on a single computer, and we already see services like OpenAI's GPT-3 API and Hugging Face's serving infrastructure meeting this need. Later tools could address allow for managing different model versions, meta-frameworks for composing large distinct pretrained models, and tools for incorporating different modalities (e.g. vision, sound, text) into pretrained models.

In addition to framework compilers and tools for pretrained models, data tooling will become increasingly important. I expand on this in the following section.

5.4 Trends in data tooling

A natural consequence of the commodification of frameworks is that complementary capabilities receive more attention. Among these capabilities, data tooling is the perhaps the most underdeveloped. First, I'll point out the lack of standardization for data manipulation, especially in academic research. Next, I'll explore some research perspectives on the rising importance of data. Finally, I'll consider whether we might expect data tooling to be open sourced and what that implies about the medium-term future.

Before discussing the specific trends within data tooling, let's consider the different types of data tools.

Data tool	**Function**	**Example**
Production	Acquisition and creation of data	MTurk, ScraPy
Inspection	Examining and visualizing dataset	Matplotlib
Curation/management	Keeping track of data versions	DVC
Database/serving	Providing pipeline for deployed models	Presto, mlflow

Table 5.1: Different types of data tooling.

5.4.1 Despite its central importance to model-building, working with data is ad-hoc in practice

One gets drawn into machine learning for the excitement of model building, but finds in reality that a huge amount of manual effort is required to prepare and verify the data. For most data-intensive projects, a majority of the time is spent preparing the data [37]. This is especially felt among academic ML researchers, who typically create their own data pipelines. In industry, large companies also seem to build their own data pipelines, but rather than a single PhD establishing a custom pipeline for their experiment, companies like Tesla will have dozens of ML engineers working on a highly efficient patented pipeline [59] that their researchers can easily build on.

Although there are several startups attempting to solve this problem, ranging from Snorkel AI to Octopai, there has yet to be standardization or a consolidation on one particular tool. Especially among researchers, there is far from a 'PyTorch for data'. While one might question whether in principle a single tool for manipulating data is possible, it is clear that data production, inspection, and curation/management is a major bottleneck within research and deployment of deep learning systems.

5.4.2 Researchers are paying more attention to data

The early deep learning researcher Andrew Ng, for instance, argues for 'data-centric AI': "[h]old the code fixed and iteratively improve the data" [48]. He argues that data has been heavily neglected (1% of research on data improvement vs 99% of research on model improvement) and helped start the 2021 NeurIPS Data-Centric AI workshop [37]. If Ng is correct, then tools for data inspection and high quality data production will become especially important.

Moreover, the recent results with large language models such as Google's PaLM, LaMDA, or OpenAI's GPT-3 [9] have been pointed to as the merit of naively adding more data in addition to compute. However, this might be equally seen as pointing to increased gains from larger amounts of data. While GPT-3 had ten times as many parameters as GPT-2, it was also trained on over an order of magnitude more data (570 GB compressed plaintext vs 40GB). Recent papers, such as Hoffman et al (2022) [34] provide further evidence that existing models can benefit from significantly increased training data.

If we expect models to continue to benefit from larger scaling, then we should also expect the growing significance of data production tools.

5.4.3 Will future data tooling be open source?

Right now it seems highly uncertain whether we would expect data tools to be open source. The companies with the largest presence in the data labeling space like Amazon (Mechanical Turk) and Scale AI are proprietary platforms. However, there are notable exceptions like Databricks, a $38B enterprise data company that builds on top of the open source framework MLFlow.

Whether data tools are open sourced depends in part on the scale of the task. Systems that require petabytes of data, such as Tesla's self-driving car pipeline, are far less likely to be open sourced than the gigabyte-sized experiments at a university or small startup. This can be best understood with the contexts of the business incentives described in Chapter 5: since academic researchers are unlikely to use large-scale data tooling, open sourcing such a tool wouldn't attract talent nor tighten the relationship with academia in the same way that open sourcing a framework could. Furthermore, the data pipeline is often the source of competitive differentiation among large companies that use AI. Nevertheless, as tools like H2O.AI indicate, we may see businesses pursuing an 'open core' model and offering accompanying services or enterprise add-ons.

One plausible scenario is a bifurcation between large and small scale tools. The large scale tools are provided by proprietary platforms like Scale whereas some fraction of the small scale data tools are open source and freely available to researchers.

Insofar as one believes that open source tools are important for research quality or fairness among research subjects, as Jo argues [38], providing support for open source data tools could be a helpful lever to foster. This need not be through the creation of an entirely new project, but rather support for existing open source projects in the domains of less developed data capabilities (e.g. data inspection). However, if we worry about the ability of organizations to misuse new capabilities (see, for instance, the increasing surveillance-relevant applications of deep learning [7]), naive open sourcing of data tools could strengthen misuse. Furthermore, although the practice of open sourcing software can lead to reduced risk from public vulnerabilities in software, this is far from a given. In many cases, without the right infrastructure for safe disclosure and bug-patching, providing the code to ML platforms can enable adversaries to abuse existing systems. Any effort to open source tooling should be done with caution and careful attention to their dual-use capabilities and the accompanying information security risks.

What do trends in data tooling imply for medium term risks from machine learning? It seems that the trends within data tooling, similar to the trends in computation, point towards increasing concentration of capabilities into a small number of firms. This may allow for easier regulation; governments have in the past successfully demonstrated ability to regulate monopolies emerging from general purpose technologies like electricity. However, it may be much more concerning that these capabilities are developing far more rapidly than our wisdom of how to control our technologies. I have no antidote, but suggest that more granular examination of the factors shaping deep learning production can provide the prerequisite knowledge we need to govern this strange (relatively) new technology.

5.5 Predictions

In the spirit of building a helpful theory, I have attempted to make my findings falsifiable by presenting a number of concrete predictions about the near term future. Each claim is accompanied by a probability estimate.

1. PyTorch, and JAX will be two of top three frameworks for deep learning according to Paperswithcode (outside of China) as of January 2027. 0.75
2. Python will be the most popular language for machine learning in 2027. 0.90
3. ONNX will become accepted as the dominant intermediate representation framework 0.7
4. Between 2023-2027, none of the top publicly disclosed 5 largest language models will be open sourced. 0.95
5. As of 2027, the three most popular platforms that provide data tooling are largely proprietary/do not open source a crucial part of their stack. 0.7

5.6 Questions for further investigation

1. What properties are shared/different between MLOSS and OSS within other domains (say compilers like LLVM)?
2. How do we expect the evolution of data tooling to be different from framework tooling?
3. How different are the capabilities required for scaling up probabilistic programming versus deep learning (specifically compute)?

4. How does MLOSS in China differ from those in the U.S? What does this imply about the diffusion of knowledge about AI research?

5. How are the incentives for open sourcing data tools different than the incentives for other MLOSS (especially frameworks)?

Bibliography

[1] Nadia Asparouhova, Public Faith. https://nadia.xyz/public-faith.

[2] Soumith Chintala interview - pytorch: Fast differentiable dynamic graphs in python. https://www.youtube.com/watch?v=am895oU6mmY,.

[3] Sqlite code of ethics: https://sqlite.org/codeofethics.html.

[4] Travis oliphant: NumPy, SciPy, anaconda, python & scientific programming | lex fridman podcast #224 - YouTube https://www.youtube.com/watch?v=gFEE3w7F0ww&t=3089s.

[5] Using JAX to accelerate our research https://www.deepmind.com/blog/using-jax-to-accelerate-our-research.

[6] Google AI. Case Studies and Mentions - TensorFlow https://web.archive.org/web/20220228134129/https://www.tensorflow.org/about/case-studies, 2022.

[7] James Dunham Ashwin Acharya, Max Langenkamp. Trends in AI research for the visual surveillance of populations. center for security and emerging technology. https://cset.georgetown.edu/publication/trends-in-ai-research-for-the-visual-surveillance-of-populations/.

[8] BCG. Artificial Intelligence and AI at Scale/ https://www.bcg.com/en-us/capabilities/digital-technology-data/artificial-intelligence. Blogpost, 2020.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[10] Miles Brundage, Shahar Avin, Jack Clark, Helen Toner, Peter Eckersley, Ben Garfinkel, Allan Dafoe, Paul Scharre, Thomas Zeitzoff, Bobby Filar, Hyrum Anderson, Heather Roff, Gregory C. Allen, Jacob Steinhardt, Carrick Flynn, Seán Ó hÉigeartaigh, Simon Beard, Haydn Belfield, Sebastian Farquhar, Clare Lyle, Rebecca Crootof, Owain Evans, Michael Page, Joanna Bryson, Roman Yampolskiy, and Dario Amodei. The Malicious Use of Artificial Intelligence:

Forecasting, Prevention, and Mitigation. *arXiv:1802.07228 [cs]*, February 2018. arXiv: 1802.07228.

[11] Kathy Charmaz. *Constructing grounded theory.* Sage Publications, London ; Thousand Oaks, Calif, 2006.

[12] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code.

[13] Soumith Chintala. Growing open-source. `https://soumith.ch/posts/2021/02/growing-opensource/`. Blogpost, August 2021.

[14] Google Cloud. Introduction to built-in algorithms > ai platform training. `https://cloud.google.com/ai-platform/training/docs/algorithms`. Website, 2022.

[15] Charles W Cobb and Paul H Douglas. A theory of production. *The American economic review*, 18(1):139–165, 1928.

[16] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS workshop*, number CONF, 2011.

[17] International Planning Conference. Booklet & Papers. Website, 2022.

[18] Allan Dafoe. AI Governance: A Research Agenda. University of Oxford, August 2018.

[19] Randall Davis and Jonathan J King. The origin of rule-based systems in AI. *Rule-based expert systems : The MYCIN experiments of the Stanford heuristic programming project / Edited by Bruce G. Buchanan and Edward H. Shortliffe*, 1984. Place: S.l. Publisher: s.n. OCLC: 848324685.

[20] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, and Andrew Y Ng. Large scale distributed deep networks. page 9.

[21] Stanford CS Department. CS230 Deep Learning. `https://cs230.stanford.edu/`. Website, 2022.

[22] Joseph Donia and Jay Shaw. Co-design and Ethical Artificial Intelligence for Health: Myths and Misconceptions. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, page 77. Association for Computing Machinery, New York, NY, USA, July 2021.

[23] Ravit Dotan and Smitha Milli. Value-laden Disciplinary Shifts in Machine Learning. *arXiv:1912.01172 [cs, stat]*, December 2019. arXiv: 1912.01172.

[24] European Commission. Directorate General for Communications Networks, Content and Technology. *The impact of open source software and hardware on technological independence, competitiveness and innovation in the EU economy: final study report.* Publications Office, LU, 2021.

[25] Richard E. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3):189–208, December 1971.

[26] Levels FYI. Facebook e4 software engineer compensation. `https://www.levels.fyi/company/Facebook/salaries/Software-Engineer/E4/`. Website, 2022.

[27] Levels FYI. Google l4 software engineer compensation. `https://www.levels.fyi/company/Google/salaries/Software-Engineer/L4/`. Website, 2022.

[28] Richard Gabriel. The rise of "worse is better". *Lisp: Good News, Bad News, How to Win Big*, 2(5), 1991.

[29] Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459, May 2015. Number: 7553 Publisher: Nature Publishing Group.

[30] Barney Glaser and Anselm Strauss. *The Discovery of Grounded Theory.* Weidenfeld and Nicolson, London, 1967.

[31] Avi Goldfarb, Bledi Taska, and Florenta Teodoridis. Could machine learning be a general purpose technology? A comparison of emerging technologies using data from online job postings. SSRN Scholarly Paper ID 3468822, Social Science Research Network, Rochester, NY, May 2021.

[32] Shane Greenstein and Frank Nagle. Digital dark matter and the economic contribution of Apache. *Research Policy*, 43(4):623–631, May 2014.

[33] M. Helmert. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26:191–246, July 2006.

[34] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models.

[35] Chip Huyen. Machine Learning Tools Landscape v2 (+84 new tools). `https://huyenchip.com/2020/12/30/mlops-v2.html`. Blogpost, December 2020.

[36] Indeed.com. Deep Learning job query. `https://www.indeed.com/jobs?q=Deep\%20Learning&l&vjk=87d0802a025023b4`. Website, 2022.

[37] Indeed.com. Probabilistic programming job query. `https://www.indeed.com/jobs?q=Probabilistic\%20Programming&l&vjk=4ac08ba6112c10e2`. Website, 2022.

[38] Eun Seo Jo and Timnit Gebru. Lessons from archives: strategies for collecting sociocultural data in machine learning. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, pages 306–316, New York, NY, USA, January 2020. Association for Computing Machinery.

[39] Kaggle. State of Data Science and Machine Learning 2021. `https://www.kaggle.com/kaggle-survey-2021`. Website, 2021.

[40] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[41] Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, 2017. Publisher: Cambridge University Press.

[42] Josh Lerner and Jean Tirole. The Economics of Technology Sharing: Open Source and Beyond. *Journal of Economic Perspectives*, 19(2):99–120, June 2005.

[43] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context. *arXiv:1405.0312 [cs]*, February 2015. arXiv: 1405.0312 version: 3.

[44] Zachary C Lipton and Jacob Steinhardt. Troubling trends in machine learning scholarship. *arXiv preprint arXiv:1807.03341*, 2018.

[45] McKinsey. Sizing the potential value of AI and advanced analytics | McKinsey. Blogpost, 2018.

[46] MindSpore. `https://github.com/mindspore-ai/mindspore`.

[47] Fiona Murray, Philippe Aghion, Mathias Dewatripont, Julian Kolev, and Scott Stern. Of Mice and Academics: Examining the Effect of Openness on Innovation. *American Economic Journal: Economic Policy*, 8(1):212–252, February 2016.

[48] Andrew Ng. MLOps: From model-centric to data-centric AI. page 29.

[49] PWC. PwC's Global Artificial Intelligence Study: Sizing the prize. Blogpost, 2018.

[50] PyTorch. PyTorch Turns 5! `https://www.youtube.com/watch?v=r7qB7mKJOFk`, January 2022.

[51] Facebook AI Research. Detectron2 A pytorch-based modular object detection library. `https://ai.facebook.com/blog/-detectron2-a-pytorch-based-modular-object-detection-library-/`. Blogpost, 2021.

[52] Facebook AI Research. PyTorch - XLA. `https://github.com/pytorch/xla`. Github Repository, March 2022. original-date: 2018-11-05T22:42:04Z.

[53] White Source Software. Facebook react finally relicensed under MIT open source license. `https://www.whitesourcesoftware.com/resources/blog/facebook-react-relicensed-under-mit-open-source-license/`.

[54] Sören Sonnenburg, Mikio L. Braun, Cheng Soon Ong, Samy Bengio, Leon Bottou, Geoffrey Holmes, Yann LeCun, Klaus-Robert Müller, Fernando Pereira, Carl Edward Rasmussen, Gunnar R&#228, tsch, Bernhard Schölkopf, Alexander Smola, Pascal Vincent, Jason Weston, and Robert Williamson. The Need for Open Source Software in Machine Learning. *Journal of Machine Learning Research*, 8(81):2443–2466, 2007.

[55] Synced Review. A Brief History of Deep Learning Frameworks. `https://syncedreview.com/2020/12/14/a-brief-history-of-deep-learning-frameworks/`. Blogpost, December 2020.

[56] The TWIML AI Podcast with Sam Charrington. Soumith Chintala Interview - Pytorch: Fast Differentiable Dynamic Graphs in Python. `https://www.youtube.com/watch?v=am895oU6mmY`. Video, November 2017.

[57] Eleuther AI `https://discord.com/invite/zBGx3azzUn`. Join the EleutherAI Discord Server! Website, 2022.

[58] Facebook AI Research. `https://paperswithcode.com/trends`. Papers with code trends. Website, 2022.

[59] Timofey Uvarov, Brijesh Tripathi, and Evgene Fainstain. Data pipeline and deep learning system for autonomous driving. `https://patentscope.wipo.int/search/en/detail.jsf?docId=WO2019245618`.

[60] Edge AI + Vision. The Caffe Deep Learning Framework: An Interview with the Core Developers. `https://www.edge-ai-vision.com/2016/01/the-caffe-deep-learning-framework-an-interview-with-the-core-developers/`. Blogpost, January 2016.

[61] Simon Wardley and D Moschella. The future is more predictable than you think– a workbook for value chain mapping. In *Leading Edge Forum, CSC*, 2013.

[62] Yao Zhou, Cong Liu, and Yan Pan. Modelling Sentence Pairs with Tree-structured Attentive Encoder. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2912–2922, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.