AI Supply Chains: An Emerging Ecosystem of AI Actors, Products, and Services

Aspen Hopkins[1*†], **Sarah H. Cen**[2*], **Isabella Struckman**[1], **Andrew Ilyas**[3], **Luis Videgaray**[4], **Aleksander Mądry**[1]

[1]Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology
[2]Department of Computer Science and Stanford Law School, Stanford University
[3]Department of Statistics, Stanford University
[4]Sloan School of Management, Massachusetts Institute of Technology

Abstract

The widespread adoption of AI in recent years has led to the emergence of AI supply chains: complex networks of AI actors contributing models, datasets, and more to the development of AI products and services. AI supply chains have many implications yet are poorly understood. In this work, we take a first step toward a formal study of AI supply chains and their implications, providing two illustrative case studies indicating that both AI development and regulation are complicated in the presence of supply chains. We begin by presenting a brief historical perspective on AI supply chains, discussing how their rise reflects a longstanding shift towards specialization and outsourcing that signals the healthy growth of the AI industry. We then model AI supply chains as directed graphs and demonstrate the power of this abstraction by connecting examples of AI issues to graph properties. Finally, we examine two case studies in detail, providing theoretical and empirical results in both. In the first, we show that information passing—specifically, of explanations—along the AI supply chains is imperfect, which can result in misunderstandings that have real-world implications. In the second, we show that upstream design choices (e.g., by base model providers) have downstream consequences (e.g., on AI products fine-tuned on the base model). Together, our findings motivate further study of AI supply chains and their increasingly salient social, economic, regulatory, and technical implications.

Introduction

Most modern AI systems are no longer developed in-house, at least not in their entirety. Instead, multiple organizations contribute to their development by providing resources, models, and datasets. This phenomenon has led to the rise of *AI supply chains*: complex networks of AI components (and the respective actors who provide them) that contribute to the production of AI systems.[1]

To illustrate, consider an AI healthcare app that transcribes and summarizes doctors' visits, as visualized in Figure 1. Today, such a product is typically built via the efforts of several organizations. One organization (`Org A`) may develop a pre-trained large language model (LLM), while another—the healthcare app developer (`Org B`)—may fine-tune it. In fine-tuning, suppose `Org B` then uses datasets curated by two unrelated hospitals (`Org C` and `Org D`). The result is an AI product built from the contributions of *at least* four participants in the AI supply chain, though numerous others may contribute additional resources such as compute, upstream datasets, and services.

The rise of AI supply chains reflects a change in the status quo of AI development that has been accelerated by generative AI and subsequent general-purpose models (Lee, Cooper, and Grimmelmann 2023). Although early AI supply chains date back several decades, general-purpose models sparked an explosion of AI adoption by making it accessible and affordable. Yet AI supply chains' rapid growth, and their ability to facilitate AI adoption, introduces new challenges and risks to AI development, safety, and regulation.

Several works have begun to document AI supply chains and their impacts, including Lee, Cooper, and Grimmelmann (2023) who explore their implications for copyright; and Widder and Nafus (2023) or Cobbe, Veale, and Singh (2023) who investigate their effects on AI accountability. Our work adds to this body of research, focusing on how *machine learning* outcomes change with the emergence of AI supply chains.

Our contributions. We formally introduce AI supply chains and examine the machine learning challenges that emerge when AI development is distributed across these networks. We consider how AI supply chains emerged, and discuss how their growing adoption suggests that the AI industry is maturing. Then, we present a model of AI supply chains as directed graphs, where nodes correspond to AI components (and their corresponding AI actors). We illustrate the usefulness of this model by connecting issues in the AI supply chain to graph properties.

Through two case studies, we demonstrate how AI supply chains affect machine learning outcomes. In both case studies, we provide formal results alongside experiments.

*These authors contributed equally.
†Corresponding authors: dataspen@mit.edu, shcen@mit.edu

[1]Note that we are differentiating "AI *in* supply chains", which refers to the popular use of AI to optimize traditional supply chain processes, from "AI supply chains", through which ML systems and services are developed and distributed.

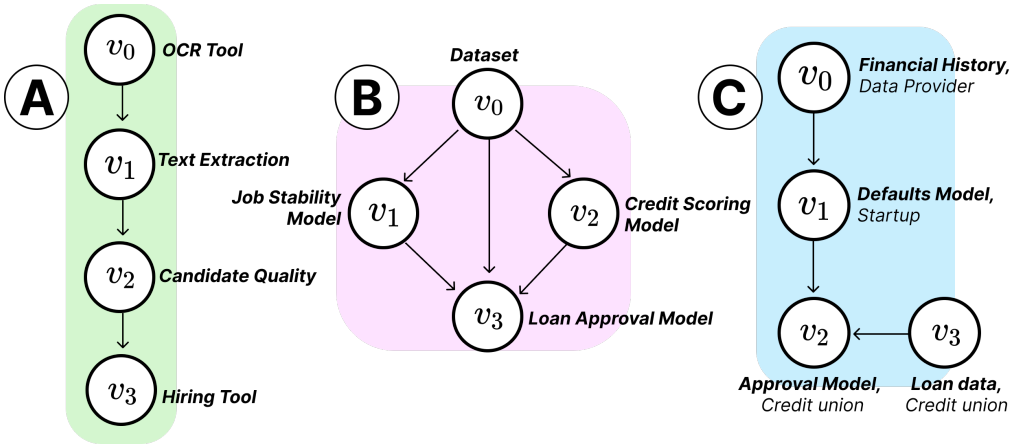


Figure 1: Three examples of AI supply chains. (1) A linear AI supply chain where an OCR tool (v_0) contributes to a text extraction tool (v_1) used when predicting candidate quality (v_2), terminating in a hiring tool (v_3). (2) A generic upstream model (v_1) trained on some dataset (v_0) that is fine-tuned (v_2) using a new dataset (v_3). (3) A healthcare transcription model (v_3) that is composed of one pre-trained LLM (v_0) and patient data from two hospitals (v_1, v_2).

The first case study examines how information propagation along the AI supply chain can be imperfect. Motivated by a setting in which there are explainability requirements (e.g., due to regulation), we show that explanation fidelity degrades with the length and width of the supply chain. In the second case study, we examine how upstream design choices affect downstream outcomes. To do so, we focus on design choices related to algorithmic fairness. We find that when upstream models satisfy certain notions of algorithmic fairness, downstream models experience different performance-fairness trade-offs. [2] Together, our analysis is intended to inform AI development and evaluation as AI supply chains continue to grow.

Background and Related Work

In this section, we discuss the emergence of supply chains in AI development. We then review prior work that examines AI supply chains or explores the setting's broader implications.

Rise of AI Supply Chains

Early machine learning (ML) development took place in silos. Single organizations (and often individual researchers) would build a predictive model end-to-end, from data collection and model selection to training and evaluation. With few exceptions, organizations did not outsource ML development.

The last two decades have seen ML (and subsequently AI) pipelines fragment at a rapid pace. Early examples of this fragmentation occurred in *datawork*. Curating datasets is a costly endeavor that requires collecting, labeling, and cleaning thousands (if not millions) of samples. One major step that the ML community took was the release of open datasets such as WordNet (Miller 1995), MNIST (Deng 2012), and ImageNet (Deng et al. 2009), which enabled ML researchers to train on existing datasets rather than undertake the expensive operation curating them. These datasets (alongside dataset libraries, including UCI Machine Learning Repository (Frank 2010) and Kaggle) accelerated ML development and set the scene for later commercialization of datawork by services, such as Mechanical Turk for data labeling.

Analogous developments for outsourcing *model training* occurred in parallel. As ML models grew, researchers sought ways to achieve high performance with fewer resources (e.g., less compute). One concept that arose was transfer learning (Pan and Yang 2009; Caruana 1994; Bengio 2012): a family of methods for adapting existing models to new tasks that encompasses, e.g., fine-tuning (Yosinski et al. 2014). This approach allowed researchers to achieve state-of-the-art results without training models from scratch. Outsourcing all or part(s) of training became popular through services such as APIs, fine-tuning or post-training services, and AutoML tools (Hutter, Kotthoff, and Vanschoren 2019).

This shift toward outsourcing AI development accelerated after 2022 with the rise of generative AI models such as OpenAI's GPT, Anthropic's Claude, and Google's Gemini. These "base" or foundation models lowered barriers to entry by making powerful AI accessible without training from scratch, encouraging organizations to accept upstream dependencies. Many modern AI supply chains now follow a common pattern: a large dataset as raw material, a base model trained on it, and a domain-specific tool fine-tuned for specialized tasks (e.g., a medical summarization app). Around these models, firms like Microsoft, Amazon, Google, Scale AI, and Databricks provide supporting products and infrastructure, while data centers scale to meet compute demands. We note that this fragmentation into supply chains is not inherently harmful; rather, it reflects a natu-

[2] We note that our findings do not indicate that algorithmic fairness is a restrictive criterion that should not be sought. Rather, our results suggest that all design choices (even the choice to not pursue fairness upstream) have downstream consequences.

ral shift signaling the AI industry's maturation. By dividing and outsourcing tasks, supply chains reduce costs and enable specialization. We anticipate that AI supply chains will continue to expand, thus motivating further study.

Related Work

The rising prevalence of AI supply chains has brought with it a growing academic interest in their effects. For example, Widder and Nafus (2023) and Cobbe, Veale, and Singh (2023) examine the social and ethical considerations of having multiple external contributors to an ML product. Lee, Cooper, and Grimmelmann (2023) discuss the legal implications, particularly with regard to copyright violation, while Attard-Frost and Hayes (2023) consider Canada's efforts in regulating AI through AI supply (or value) chains. Bommasani et al. (2023) documents examples of AI supply chains in the wild via "ecosystem graphs," with an emphasis on the role of large pre-trained foundation models. In considering the impacts of foundation models, Suresh et al. (2024) explore how participation can shape general-purpose AI models by targeting downstream applications. Our work contributes to this growing area of research, with an emphasis on their technical implications.

Although not explicitly studied in the context of AI supply chains, there is a mature body of work in ML on problems that we can view as constituent pieces of AI supply chains. For example, transfer learning (Krizhevsky, Sutskever, and Hinton 2012), and rapidly growing contributions to modular multi-modal models (Xu et al. 2023) show how model components may be adapted to new settings. A separate line of work in algorithmic fairness (Dwork and Ilvento 2018) studies the problem of *fairness under composition*, where the goal is to reason about the behavior of distinct ML models. For data, work in federated learning McMahan et al. (2017), data poisoning (Biggio, Nelson, and Laskov 2012), and backdoor attacks (Khaddaj et al. 2023) explore how "upstream" AI decisions affect (and in particular, break) downstream expectations. Each of these directions (amongst numerous others) contributes to our technical understanding of the challenges faced in fragmented AI development.

AI Supply Chains As Directed Graphs

Despite the relative nascence of the AI industry, AI supply chains are remarkably intricate. They consist of models (pre-trained, fine-tuned, or post-trained), datasets (raw or curated, labeled or unlabeled), predictions, benchmark evaluations, AI services, and more. We refer to the above as "components." Moreover, the actions that AI actors can take on these components are varied and complex. For example, given a model as the "component" of interest, an AI actor can take multiple actions: (i) they can *adapt* a pre-trained model to a specific context (Hu et al. 2021; Zhuang et al. 2020); (ii) they can *aggregate or synthesize* multiple models, as is done in certain ensembling methods like weight-averaging (Ganaie et al. 2022); and (iii) they can *train or predict using the outputs* of models (e.g., AI-generated scores or synthetic data). This is certainly not a comprehensive list (retrieval-augmented generation, or RAG, (Lewis et al. 2020) falls into a new category, as do cases where training or evaluation artifacts, e.g., learning gradients, might be shared between participants), but it illustrates the multitude of common actions that can occur on a single AI component.

Below, we propose directed graphs as a natural way to flexibly model AI supply chains and capture these interactions, where nodes (or vertices) correspond to AI components and links (or edges) indicate the dependencies between these components.

Model

Let an AI supply chain be given by a graph $G = (V, E)$, where each node $v \in V$ denotes an *AI component* (such as a model or dataset), and $(v_j, v_i) \in E \subset V \times V$ if and only if there is a directed edge from node v_j to v_i. A node v_k is a *parent* of v_i if there exists a directed edge $(v_k, v_i) \in E$ from v_k to v_i. Thus, an edge corresponds to an *operation* used to obtain a downstream component from its parent. A node v_k is an *ancestor* of v_i (and v_i is a *descendant* of v_k) if there exists a directed path from v_k to v_i, i.e., there exists a path $(v_k, v_{k'}), (v_{k'}, v_{k''}), \ldots, (v_{k'''}, v_i) \in E$.

Example 1. *Consider the example visualized in Figure 1 (2). Vertices v_0 through v_3 denote the AI components. The AI components v_1 and v_3 are the parents of v_2 while v_0, v_1, and v_3 are ancestors of v_2. Both v_0 and v_3 have no parents or ancestors. The AI component v_1 has one parent, v_0, which is also its only ancestor. The set of ancestors for v_2 is $\{v_0, v_1, v_3\}$.*

In this work, we restrict our attention to components that correspond to models or datasets. We leave analysis of additional components (e.g., compute) to future work. To distinguish between organizations in our figures, we label vertices with their AI component *and* the organization that owns it, e.g. (`GPT-4, OpenAI`). In cases where all the nodes are owned by a single organization, we omit the second label.[3]

Examples

Viewing AI supply chains as directed graphs helps formalize our conceptual understanding of them. To illustrate, we use the remainder of this section to discuss four issues that arise in AI supply chains, instantiating examples using the setup described above. In this section, let h_v denote the mapping from v's parents to v for all $v \in V$, e.g., $v_2 = h_{v_2}(v_1, v_3)$ and $v_1 = h_{v_1}(v_0)$ in Figure 1.1.

AI components do not mix modularly. When components are modular, connecting or disconnecting them does not change their individual attributes—much like connecting Lego bricks does not change their shape or color. This modularity is beneficial in supply chains (Jayaram and Vickery 2018). For example, modularity supports efforts to trace failures or assign liability, or to enforce IP licensing and distribute royalties. Modularity in AI supply chains can therefore be desirable.

[3]Organizations can encompass multiple autonomous or semi-autonomous divisions (Chandler Jr 1969; Williamson 1975), and their distinction and definition entails a large body of work. Our labeling is intended to account for this variation.

However, AI supply chains tend to mix the contributions of AI components in a *non-modular* way, much like a blended soup. Consider the transfer learning example in Figure 1.2, where model v_1 is adapted (or fine-tuned) to produce a new model v_2. Given v_2, one cannot easily discern the AI components that produced it. In contrast to modular supply chain components, it is difficult to disentangle the effects of v_2's ancestors on the final AI component. We might therefore study non-modularity as follows.

Example 2. *One can think of non-modularity as arising when, for a fixed AI supply chain G and node v, there is a non-unique mapping from v's ancestors to v. Formally, let $a_1, a_2, \ldots$ denote v's ancestors and recall the definition of h_v. Then, non-modularity is captured when there exist $(h_v, h_{a_1}, h_{a_2}, \ldots) \neq (h'_v, h'_{a_1}, h'_{a_2}, \ldots)$ such that the behavior of the resulting AI component v is the same under either set of mappings. Intuitively, given the final AI component v, one cannot separate the components that gave rise to it in a modular way.*

AI supply chains exacerbate transparency issues in AI. As AI supply chains grow, there may be "hidden interactions" between components. More precisely, there may be interactions between the ancestors of an AI component v that are difficult to identify from the perspective of v. To make this concrete, observe that v is typically aware of its parents: the AI components that directly contribute to v. In some cases, v may even be aware of its grandparent nodes. However, v generally does not have full knowledge of the graph and may be unaware of interactions between its ancestors.

The longer or more complex an AI supply chain, the more likely it is for hidden interactions to present. For instance, consider Figure 1.2. Suppose that v_2 does not have knowledge of G beyond its parents. Now suppose that there is an edge from v_0 to v_3 that v_2 is not aware of. A number of consequences may ensue from this scenario. For example, the model may exhibit overconfidence on subsets of data included in both v_0 and v_1, leading to downstream misuse. Further, as the interaction was hidden, v_2 may not know to monitor for this possibility, and may similarly lack a mechanism for repair. Hidden interactions can thus introduce vulnerabilities by obscuring characteristics about AI components that may seem innocuous when considered independently, but disproportionately amplify harms when combined. Such interactions are the result of incomplete information in or regarding the AI supply chain and its components. Under our model, we might capture this phenomenon as follows.

Example 3. *Consider an AI component v. Suppose that v has knowledge of its ancestors, but only up to m hops away. That is, v is aware of any ancestors that are within m hops of v and of the edges between these nodes. Then, one can define hidden interactions as the interactions between ancestors further than m hops away from v. There may, for example, be a cycle that includes v but has length $> m$.*

Spreading control over multiple organizations compromises AI resilience. Another consequence of distributing parts of machine learning development across an AI supply chain is that control over AI development is "dispersed" across multiple organizations. Stated differently, a single organization does not, in general, have full control of the pipeline that generates its AI components. Dispersed control has ramifications on the robustness and resilience of AI products and services. If, for instance, an upstream model is updated, it may break downstream AI components or cause them to behave unexpectedly. When all AI components are controlled by the same organization, the teams developing each component can address the problem internally, avoiding communication and information-sharing barriers that exist across organizations. Perhaps more worryingly, actions taken upstream may not be reversible downstream; as such, a downstream organization may not be able to achieve their own design objectives without changing their upstream AI vendors. Consider the following formalization.

Example 4. *Suppose that every AI component $v \in V$ is associated with a different organization. Suppose further that we consider m-dispersed control to exist for an AI component v when changes to ancestors within m hops of v cannot be "reversed" by any changes made to h_v. Formally, recall the definition of h_w. Suppose that $h_w \in \mathcal{H}_w$ for model classes $\{\mathcal{H}_w\}_{w \in V}$. Let $\mathcal{A}_m \subset V \setminus \{v\}$ denote the ancestors of v that are within m hops of v. Then, m-dispersed control occurs at v when there exists $\{h'_a \in \mathcal{H}_a\}_{a \in \mathcal{A}_m}$ such that no h'_v is able to match the behavior v under the original mappings $(h_v, h_{a_1}, h_{a_2}, \ldots)$.*

Cycles in AI supply chains facilitate feedback loops. AI supply chains may contain cycles. For example, an LLM may be used by a downstream organization to generate articles that are then scraped as new training data for an updated version of the LLM. Many of the dangers caused by cycles in AI supply chains are amplified when there are hidden interactions or dispersed control. However, even in their absence, cycles present challenges to AI performance, fairness, and more. Cycles in AI supply chains can be cast as the feedback loops previously discussed in machine learning, including Ensign et al. (2018)'s work on feedback loops in policing, Kleinberg and Raghavan (2021); Bommasani et al. (2022)'s on algorithmic monocultures, and may hold parallels to homogenization mechanisms in recommendation systems (Chaney, Stewart, and Engelhardt 2018). Consider the following example.

Example 5. *Let v_0 be an LLM. Suppose that there are a set of downstream models $\{v_1, v_2, \ldots, v_m\}$ that generate text (such as articles, blogs, and resumes) and are descendants of v_0. Then, if v_0 is continually retrained on textual data obtained from these downstream models, i.e., there is a cycle from v_0 to v_i and back to v_0 for all $i \in [m]$. Such cycles can create a feedback loop and lead to text homogenization. It is possible that the effects of a feedback loop are dampened by, for instance, the existence of many incoming edges to nodes in the cycle that disrupt the feedback loop. One potential method of measuring would be to compare, e.g., the betweenness centrality of nodes in and adjacent to the cycle.*

Case Study 1: Passing Information Along the AI Supply Chain

In an AI supply chain, multiple actors have control over different parts of an AI pipeline. Since no single actor has full control over the pipeline, the actors coordinate by communicating information to one another. In this section, we explore the implications of passing information along the AI supply chain. Specifically, we show that AI supply chains hamper the ability of downstream developers to provide accurate explanations of their predictions. We motivate this problem by considering a setting in which a downstream developer is required to provide an explanation for each of their predictions. However, the developer has limited access to upstream models and cannot probe them in the same way they would an in-house model to provide an explanation. The downstream developer must therefore produce an explanation that is built on explanations that the intermediate model providers produce. Below, we illustrate how downstream explanations built in this way can become increasingly fraught.

Problem Setup

Consider a supply chain corresponding to an AI-powered hiring tool: given a candidate's information, the hiring tool makes a suggestion about whether they should be hired. That is, given an individual represented by a feature vector $\mathbf{x} \in \mathbb{R}^\rho$, the tool makes *a prediction* $y \in \mathbb{R}$ about their success as an employee. Rather than develop the entire model in-house, the developer of this tool leverages a supply chain of third-party services to, for example, process the candidates' resumes and recommendation letters, thus forming an AI supply chain (an example of which can seen in Figure 1 (1)). Formally, one can express the downstream hiring tool $f_v : \mathbb{R}^\rho \to \mathbb{R}$ as a composite function:

$$f_v(\mathbf{x}) = h_v(\mathbf{x}, f_{p_1}(\mathbf{x}), \ldots, f_{p_n}(\mathbf{x})),$$

where $p_1, \ldots, p_n \in V$ are v's parents. Note that f_v and h_v are not the same, as f_v acts on $\mathbf{x}$, whereas h_v acts on inputs that may themselves depend on $\mathbf{x}$.

In this section, we characterize how AI supply chains affect the ability to provide accurate explanations. For simplicity we will focus on *locally linear explanations*, which includes methods such as LIME (Ribeiro, Singh, and Guestrin 2016).

Definition 1 (Local linear explanation)**.** *A δ-local linear explanation at $\mathbf{z}$ for a model g is a matrix $E_\delta(g, \mathbf{z})$ satisfying*

$$E_\delta(g, \mathbf{z}) \in \arg\min_W \mathbb{E}_{\mathbf{u}} ||g(\mathbf{z} + \delta\boldsymbol{u}) - g(\mathbf{z}) - W^\top \mathbf{u}||_2^2,$$

where $\mathbf{u}$ are drawn uniformly at random from the unit ball in $\mathbb{R}^\rho$. That is, $E_\delta(g, \mathbf{z})$ is the best linear *approximation of g around the point $\mathbf{z}$.*

A locally linear explanation is a linear approximation of how g behaves in the neighborhood of $\mathbf{z}$, where the size of the neighborhood is controlled by δ. To see this, note that as $\delta \to 0$, $E_\delta(g, \mathbf{z})$ is equivalent to the Jacobian of f at $\mathbf{z}$ if g is differentiable. More explicitly, W as a linear mapping applied to perturbation $\mathbf{u}$; since $E_\delta(g, \mathbf{z})$ minimizes the right-hand side, it is the choice of W that (in expectation) best approximates the behavior of g between $\mathbf{z}$ and $\mathbf{z} + \delta\mathbf{u}$ for all choices of $\mathbf{u}$ in the δ-ball around $\mathbf{z}$.

Although locally linear explanations are imperfect, they are often used in practice because they provide a first-order approximation of how small changes in $\mathbf{z}$ affect g's output. We adopt this formulation as it includes popular approaches, such as LIME (Ribeiro, Singh, and Guestrin 2016) and SHAP (Lundberg 2017).

Explanation error. In this section, we examine the scenario where the downstream developer is required to report an explanation of its model decision. Because the downstream developer can only access its own model and not upstream ones, it must construct its explanation based on its own model and the information communicated by its parents p_i; namely, its parents' explanations of the parent models.

We then show that the error in the communicated explanations from a node's parents can propagate the error in upstream explanations, leading to compounding error as the supply chain grows. To gain intuition for why error can compound, note that it is generally infeasible to compute $E_\delta(g, \mathbf{z})$ exactly for $\delta > 0$, so organizations typically compute empirical approximations $\hat{E}_\delta(g, \mathbf{z})$ instead (e.g., via linear regression). Fitting a locally linear explanation can take hundreds or even thousands of queries to each model of interest, and the explanations that developers provide thus always contain some approximation error.

Computing explanations. In this section, we will repeatedly make use of the "chain rule". That is, when a developer (i.e., node) v receives an explanation $\hat{E}_\delta(f_p, \mathbf{x})$ from each parent p, they combine their parents' explanations with the explanation of their own model $\hat{E}_\delta(h_v, \mathbf{z})$ to produce the *full* explanation $f_v(\mathbf{x})$ using the chain rule. (Here, $\mathbf{z}$ is the input that node v receives, i.e., $\mathbf{z} = (\mathbf{x}, f_{p_1}(\mathbf{x}), \ldots, f_{p_n}(\mathbf{x}))$.)

Formally, given a node i and its parents $\text{pa}(i)$, then the explanation that i computes given the explanation of its parents is given by:

$$\hat{E}_\delta(f_i, \mathbf{x}) = \sum_{j \in \text{pa}(i)} \hat{E}_\delta(h_i, \mathbf{z}_i)^\top \hat{E}_\delta(f_j, \mathbf{x})$$

The chain rule is the appropriate way to combine locally linear explanations because $E_\delta(g, \mathbf{z})$ approaches the Jacobian of g at $\mathbf{z}$ (if g is differentiable and locally smooth) as $\delta \to 0$.

Explainability Across Supply Chains is Unreliable

Suppose that an employer must explain why an application $\mathbf{x}$ receives a decision y, but $\mathbf{x}$ is first pre-processed by upstream services f_{p_i}. How faithful is the employer's explanation to the actual AI pipeline's behavior? In this section, we formalize the phenomenon that communicating across an AI supply chain can lead to downstream information being misleading. In particular, we show that, even for a linear supply chain, the "error" of a locally linear explanation is compounded by a constant that grows exponentially in the depth of the supply chain. For convenience, we call the "end-to-end" explanation the explanation that is computed when a single actor has access to the entire AI system and can compute the explanation directly, and we call the "supply-chain"

explanation the explanation that is computed by combining the explanations provided by actors in the supply chain.

Theorem 1. *Consider the setup in . Consider a node* 1 *that is the descendant of a linear supply chain of depth* d*, where* $j+1$ *is the parent of node* j*. Let* $\hat{E}_\delta(h_j, \mathbf{z}_j) = E_\delta(h_j, \mathbf{z}_j) + \Delta_j$ *for all* j*, where we assume each* Δ_j *is a random matrix with mean-zero entries drawn i.i.d. from a distribution with variance* σ^2*. We assume that: (i)* $\|E_\delta(h_i, \mathbf{z}_i)\|_2^2 \leq C_1$ *and* $\|E_\delta(f_{i+1}, \mathbf{x})\|_F^2 \leq C_2$ *for universal constants* C_1 *and* C_2*; (ii)* f_j *and* h_j *are locally smooth and differentiable at* $\mathbf{x}$ *and* $\mathbf{z}_j$*, respectively, for all* j *and all* $\mathbf{x}$ *of interest; (iii)* Δ_j *is independent of any quantities upstream of it; (iv)* $\mathbb{E}[\mathbf{z}_i] = [0]$ *for all* i*; and (v)* $dim_2(\Delta_i) = dim_2(\Delta_j)$ *for all* $i, j \in [d]$*. Note that (v) is not necessary but simplifies the expression. Then,*

$$\mathbb{E}\left[\|\hat{E}_\delta(f_1, \mathbf{x}) - E_\delta(f_1, \mathbf{x})\|_F^2\right] \leq$$

$$C_3^{d-1}\mathbb{E}\left[\|\hat{E}_\delta(f_d, \mathbf{x}) - E_\delta(f_d, \mathbf{x})\|_F^2\right] + C_4 \sum_{\tau=1}^{d-1} C_3^{\tau-1}, \quad (1)$$

as $\delta \to 0$*, where* C_3 *and* C_4 *are constants that depend on* $\dim_2(\Delta)$ *and* σ^2 *as well as* C_1 *and* C_2*, respectively. In other words, the distance between the supply-chain explanation and the end-to-end explanation at the downstream node amplifies the upstream error by an expression that can be upper bounded by an exponential of the supply chain depth* d*. Moreover, we show that this bound is tight.*

The proof is given in the Supplementary Materials. In this result, one can think of $\mathbb{E}[\|\hat{E}_\delta(f_d, \mathbf{x}) - E_\delta(f_d, \mathbf{x})\|_F^2]$ as the error in an end-to-end explanation and $\mathbb{E}[\|\hat{E}_\delta(f_1, \mathbf{x}) - E_\delta(f_1, \mathbf{x})\|_F^2]$ as the error in the supply-chain explanation after information is passed along a d-length linear supply chain. Therefore, this result says that the supply-chain explanation **can amplify the error of an end-to-end explanation by a factor that is exponential in** d**, plus an additive term.** The root issue is that each organization does not have direct access to upstream models and, in the absence of such access, relies on explanations provided by upstream organizations.[4]

Another way of viewing this result is that the empirically estimated explanations $\hat{E}_\delta(h, \mathbf{z})$ must be sufficiently close to the *true* value $E_\delta(h, \mathbf{z})$ for explanations in AI supply chains to be meaningful. This result has many implications. For instance, from a policy perspective, if model explanations are legally mandated but a model is situated in an AI supply chain, policymakers must ensure that either (1) the model's ancestor tree is not too deep (or wide), or (2) that all upstream organizations provide explanations that are accurate within a tolerance that increases in strictness as the ancestor tree grows, which places a strict requirement on upstream organizations.

Although our case study is centered around explanations, this result implies that information passing of any kind along the AI supply chain should be considered carefully.

Empirical Evaluation

In the previous section, we show that AI supply chains complicate the process of generating accurate explanations. We now study this phenomenon experimentally, extending the theoretical result from a linear supply chain to one in which a node's ancestors are an inverted m-ary tree of depth d.

Setup. As before, we refer to "end-to-end" explanations as LIME explanations computed when given access to the entire AI supply chain, and "supply-chain" explanations as those computed by combining the explanations of upstream models. We use the chain rule to compute the supply-chain explanations (see). Our goal is to compare the end-to-end and supply-chain explanations; if there is information loss in the supply chain, the two explanations will differ.

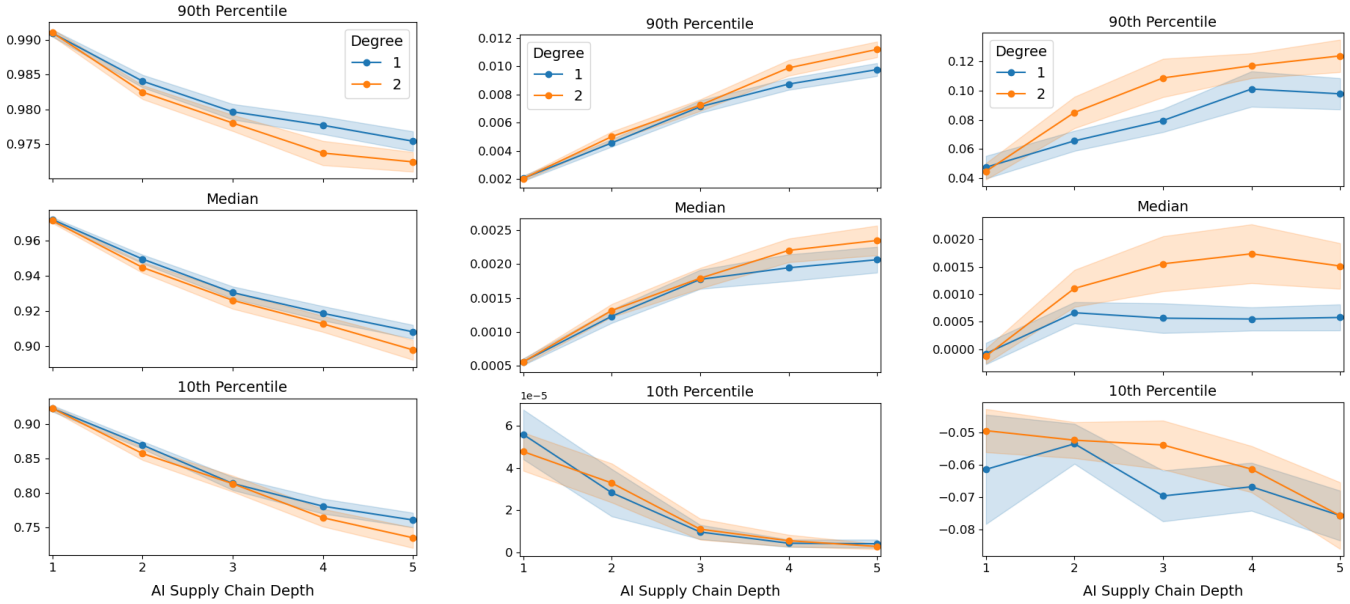
For the AI supply chain, we consider the explanations for a downstream node whose ancestor graph is an m-ary inverted tree of depth d; doing so allows us to study different AI supply chains by varying m and d. At each level of the tree, a model passes the query of interest $\mathbf{x}$ to its parents, then concatenates the output of its parents to $\mathbf{x}$ and passes this to its own model. Each model (i.e., node) in the ancestor graph is a simple three-layer neural network with ReLU activations and BatchNorm after each hidden layer, ending with a linear layer and sigmoid activation that produces a scalar output. The hidden dimensions double with each hidden layer. In order to mimic real-world model interactions, models further upstream are larger; specifically, the first hidden layer of a node's model is 16 multiplied by the "level" of node in the ancestor tree. Similarly, the larger models are trained for more epochs ($15\sqrt{\text{node level}}$) and on larger datasets ($1000\sqrt{\text{node level}}$). All models are trained with a learning rate of 0.001 and a batch size of 32.[5]

We generate samples $\mathbf{x} \in \mathbb{R}^{\text{featDim}}$ by sampling a mixture of two isotropic Gaussians with variance 1, where the cluster centers are sampled uniformly at random within the unit hypercube. We generate binary training labels using a second-order polynomial model with Gaussian noise, where coefficients are sampled uniformly in $[-1, 1]$, and the output is then passed through a sigmoid function and thresholded at 0.5. For each trial, we compare the end-to-end and supply-chain explanations for 100 different values of $\mathbf{x}$. We run 50 trials and set featDim to 12. We set number of samples LIME uses to compute each explanation to 50 or 100 and the LIME radius to 0.1 or 0.2, with further results included in the Supplementary Materials.

Results. We measure three quantities of interest: The first is the cosine distance between end-to-end and supply-chain explanations. The second is the mean-squared error (MSE)

[4] Alternatively, downstream organizations could query upstream models many times (e.g., typically hundreds or thousands of times for LIME estimates), but doing so can increase API costs by multiple orders of magnitude, so we assume that downstream organizations would not do this unless absolutely necessary.

[5] Note that the training conditions are not necessarily realistic, but as we are only interested in the fidelity of explanations and not the performance of the models, the predictive abilities of the models (even what they predict) do not affect our results. In fact, one could run our experiment on randomly initialized models.



(a) Cosine similarity (b) MSE (c) Recourse error

Figure 2: Simulations for Case Study 1 for 50 LIME samples and a LIME radius of 0.2. From left to right, we plot (a) the cosine similarity between end-to-end and supply-chain explanations, (b) the mean-squared error (MSE) between the two explanations, and (c) the recourse error between the two explanations. In all three figures (a)-(c), the x-axis is the depth of the ancestor tree and the legend gives the "width" or degree of the ancestor tree. The three subplots show the means and 95% confidence intervals of the 90th percentile, 50th percentile (median), and 10th percentile of the respective metric.

between the two explanations (supply-chain minus end-to-end). The third is what we refer to as the *recourse error*: the distance one would travel along the direction indicated by the supply-chain explanation in order to change one's predictive outcome minus the analogous quantity for the end-to-end explanation. Intuitively, while explanations are sometimes of standalone interest, they often serve as mechanisms to give individuals a path to *recourse*. That is, individuals use explanations to improve their outcomes (e.g., if an applicant is rejected, they may use the explanation to improve their qualifications and therefore future applications). Our definition of recourse error is inspired by the literature on counterfactual explanations, which examine how far one would have to move along the direction indicated by the explanation to flip one's prediction from positive to negative, or vice versa (Wachter, Mittelstadt, and Russell 2017). To avoid instabilities in our analysis, we set the max value of the recourse error to 1000. Our analysis therefore sheds light on whether explanations produced through an AI supply chain can mislead or place undue burden on individuals.

Our results are summarized in Figure 2. For each measure, there are three subplots for the 90th, 50th, and 10th percentiles of the respective metric. More explicitly, recall that we run the simulation 50 times to average over the randomness in the model training and data generation. Moreover, recall that, for each simulation we computed explanations for 100 different values of $\mathbf{x}$. The percentiles are taken over the 100 values of $\mathbf{x}$, and the means and 95% confidence intervals are taken over the 50 trials. We choose to provide these three percentiles to illustrate the "best", "median", and "worst" case outcomes across values of $\mathbf{x}$, as means and medians may fail to capture the overall behavior.

Corroborating our theoretical analysis, the cosine similiarity between end-to-end and supply-chain explanations degrades as the supply chain length increases, as can be seen by all the decreasing trends in Figure 2a. Additionally, as one may expect, the cosine similarity also degrades with the ancestor tree's degree. The MSE in Figure 2b indicates a similar trend, as both the 90th percentile and median values increase with depth and degree. Although the 10th percentile MSE seems to improve, note that the scale of this plot's y-axis is very small. Finally, we also examine the recourse error in Figure 2c. It shows that the error increases with the depth and degree of the supply chain, which extends beyond our theoretical findings (note that the 90th percentile recourse error scale is larger than the other two percentiles). Moreover, if we study the sign of the recourse error (which is the supply-chain recourse distance minus the end-to-end recourse distance), we find that the recourse distance of the supply-chain explanation is consistently **larger** than that of the end-to-end explanation, across all three percentiles.

Taken with the cosine similarity result, Figure 2a and 2c indicate that if a decision subject seeks to change their outcome under an AI-driven decision, the supply-chain explanation points them in a **different direction** than an end-to-end explanation would, causing decision subjects to have to exert more effort than they would under the end-to-end explanation.

Case Study 2: Upstream Choices Affect Downstream Outcomes

In the previous section, we examine how communication along the AI supply chain can lead to misleading information downstream. In this section, we turn our attention to how decisions made upstream can fundamentally restrict downstream actors. We study a setting in which an upstream model is designed to meet some notion of algorithmic fairness. We then show that **fine-tuning on this model restricts downstream models in their ability to achieve their own performance-fairness objectives**. Our result goes beyond showing that downstream models inherit the upstream notion of fairness. In fact, we show **the downstream model can *undo* upstream fairness, but at the cost of other desiderata**. Thus, training upstream models to satisfy an attribute does not guarantee that downstream models will inherit that attribute but can restrict the downstream developer in other unexpected ways. Although we study algorithmic fairness, this result does not indicate that implementing fairness is restrictive; rather, we use fairness as a concrete example but believe many design choices (even the lack of an explicit choice as a choice in and of itself) have downstream effects.[6]

Problem Setup

Consider an upstream base model f_p and a downstream model f_v, where f_v is fine-tuned on f_p. Suppose both f_v and f_p take in inputs $\mathbf{x} \in \mathbb{R}^\rho$, which are realizations of the random variable X, and output $y \in \mathbb{R}$.[7] For instance, $\mathbf{x}$ could be a vector of features representing an individual that the downstream model seeks to evaluate, and instead of using the off-the-shelf base model f_p, the downstream actors fine tunes it on specialized data to produce f_v.

Upstream design choices. Model developers inevitably make design choices when training their models. These choices are not necessarily optimal for downstream actors, as developers cannot tailor their base models to suit all possible downstream use cases (unless they are providing custom models). In our analysis, we use the abstraction that the upstream design choice can be represented as a *conditional independence desideratum*. That is, we suppose f_p is trained such that $f_p(X)$ is (approximately) conditionally independent of X_1 given random variable $Z \in \mathcal{Z}$, i.e.,

$$f_p(X) \perp X_1 \,|\, Z,$$

where $X = (X_1, X_2, \ldots)$, and the randomness is given by the training data distribution For example, Z could be $(X_2, \ldots, X_d)$ or Z could be the empty set.

We adopt this setup because conditional independence encompasses a family of requirements that appear not only in algorithmic fairness but other types of "structured" settings, such as causal inference on structural causal models. It is thus a natural class of requirements that an upstream developer may consider. For instance, statistical parity and equalized false positive rates are both notions of algorithmic fairness that can be expressed as conditional independence requirements (Barocas, Hardt, and Narayanan 2017). Statistical parity for sensitive attribute X_1 is equivalent to setting $Z = \emptyset$, and equalizing odds across sensitive groups determined by X_1 is equivalent to setting $Z = Y$, where Y is the true outcome (e.g., how well a job applicant would perform if they were hired). Intuitively, $X_1 \perp f_p(X) \mid Z$ asks that, given information Z, knowing the value of a sensitive attribute (e.g., race or sex) X_1 provides no additional information about $f_p(X)$.

We examine the implications of fine-tuning f_v on f_p when f_p satisfies this conditional independence. In the next two sections, we provide a theoretical characterization, followed by a more general empirical evaluation.

Conditional independence affects embeddings. We now show that upstream design decisions can influence downstream models in ways that are not immediately obvious. Naively, we might expect that imposing a constraint on the upstream model f_p will lead to the downstream model f_v inheriting the same constraint. However, we show—consistent with prior empirical findings (Salman et al. 2022; Steed et al. 2022; Yuan et al. 2024; Hu et al. 2025)—that a downstream developer can deliberately override or "undo" the upstream constraint. Contrasting prior work, however, we show that undoing the upstream constraint has implications for the developer's ability to meet their other downstream requirements.

We begin with a theoretical result. We model f_p as a linear combination of basis functions:

$$f_p(\mathbf{x}) = \sum_{i=1}^{N} \phi_i(\mathbf{x}) w_i, \tag{2}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_\rho) \in \mathbb{R}^\rho$, $\phi_i : \mathbb{R}^\rho \to \mathbb{R}$, $w_i \in \mathbb{R}$, $\mathbf{w} = (w_1, \ldots, w_N) \in \mathbb{R}^N$, $\mathbf{\Phi} = (\phi_1, \ldots, \phi_N) \in \mathbb{R}^{N \times \rho}$. We assume that ϕ_i's (and thus f_p) are deterministic. Similar models appear in kernel methods (Schölkopf 2002), function approximation (Murphy 2012; Hornik, Stinchcombe, and White 1989; Cybenko 1989), and deep learning theory (Goodfellow, Bengio, and Courville 2016). Such models are popular because it is a universal function approximator as $N \to \infty$ under sufficiently rich basis functions. Intuitively, one can think of the basis functions as embeddings and N as a proxy for model size.

We model fine-tuning a downstream model f_v on an upstream model f_p as learning a linear function on the embeddings of the upstream model f_p. As the basis functions $\boldsymbol{\phi}_i$ represent the embeddings of f_p, the fine-tuned model is given by $f_v(\mathbf{x}) = \sum_{i=1}^{N} \boldsymbol{\phi}_i(\mathbf{x}) r_i$, where the coefficients $r_i \in \mathbb{R}$ are learned from fine-tuning.

Theorem 2. *Suppose the base model is trained to achieve approximate independence of $f_p(X)$ and X_1 given Z such*

[6] Although we focus on fairness, we believe our findings signal more broadly that upstream decisions have unexpected downstream repercussions. Notably, our results do not show that upstream decisions are simply *reflected* in downstream models (i.e., upstream fairness appears downstream); rather, upstream decisions can be "undone" but at the cost of other considerations.

[7] We adopt the convention that capital letters denote random variables and lowercase letters denote realizations.

that there exists $\varepsilon_p > 0$ for which

$$\left| \frac{\partial}{\partial x_1} \mathbb{E}_{D_p}[f_p(X) \mid X_1 = x_1, Z = z] \right| \leq \varepsilon_p,$$

for all $x_1 \in \mathbb{R}$ and $z \in \mathcal{Z}$. Let ε_v denote the analogous value achieved by f_v (i.e., how close $f_v(X)$ is to conditional independence from X_1 given Z). Note that this is an appropriate condition for approximate conditional independence, since f_p is a binary classifier.[8] *Similarly, suppose the downstream model is trained to achieve approximate independence of $f_v(X)$ and X_2 given Z such that*

$$\left| \frac{\partial}{\partial x_2} \mathbb{E}_{D_v}[f_p(X) \mid X_2 = x_2, Z = z] \right| \leq \eta_v,$$

for all $x_2 \in \mathbb{R}$ and $z \in \mathcal{Z}$. Then, as long as $\varepsilon_p, \eta_v > 0$ and the bounds above are tight for some value of (x_1, z) and (x_2, z), then there exist functions g, S, and S' such that

$$\frac{\varepsilon_v}{\varepsilon_p} \cdot C_5(\mathbf{w}, \mathbf{\Phi}) \leq \|\mathbf{r}\| \leq \eta_v \cdot C_6(\mathbf{r}, \mathbf{\Phi}), \tag{3}$$

for constants C_5 and C_6 that depend on $\mathbf{w}$ and $\mathbf{r}$, respectively, and both constants can depend on $\mathbf{\Phi}$ and the types of upstream/downstream constraints.

The proof is provided in the Supplementary Materials. This result shows that a design choice made on the upstream model f_p (as captured by ε_p) affects the downstream, fine-tuned model f_v (as captured by $\|\mathbf{r}\|$, ε_v, and η_v). Concretely, we can make three observations from the result (and corresponding proof). First, the downstream model f_v does not necessarily inherit the upstream conditional independence. In fact, f_v can "undo" it, which would be captured by ε_v being large compared to ε_p. This leads to our second observation, which is that the upstream conditional independence (with respect to X_1) affects f_v's ability to achieve its conditional independence with respect to X_2. For example, even if f_v may wish to "undo" the upstream conditional independence with respect to X_1, doing so implies increasing ε_v, which raises the lower bound on $\|\mathbf{r}\|$. However, the right-hand side of 3 shows that $\|\mathbf{r}\|$ is also bounded above by η_v multiplied by a constant that (as shown in the proof) does not depend on $\|\mathbf{r}\|$, but this may not be compatible with f_v's desire to achieve their own downstream design desideratum by keeping η_v small. (Although one may be able to satisfy both conditional independences by letting η_v and ε_v *both* be small, this is often impossible in practice because η_v and ε_v are often inversely related. For example, it is well known that various notions of algorithmic fairness are impossible to achieve simultaneously (Chouldechova 2017)). Third, the downstream model is restricted in its choice of model parameters $\mathbf{r}$ if it wishes to fine-tune *and* satisfy its own desideratum η_v. Because the downstream developer cannot change the upstream constraint ε, it *must trade off between what it can control: its own desideratum and f_v's performance*.

[8] For regression, one would require instead that there exists an $\varepsilon' > 0$ such that $\left| \frac{\partial}{\partial x_1} P[f_p(X) = y \mid X_1 = x_1, Z = z] \right| \leq \varepsilon_p$ for all $x_1 \in \mathbb{R}$, $z \in \mathcal{Z}$, and $y \in \mathbb{R}$.

In other words, this result shows that **(i) the downstream model does *not* necessarily inherit the upstream constraint and can in fact reverse it, but (ii) even so, the upstream constraint restricts the downstream model and imposes a trade-off between the downstream's performance and own desideratum** (or, more accurately, imposes a trade-off that is stronger than what would naturally exist). Fact (i) is consistent with the literature showing that upstream unlearning or fairness can be undone with fine-tuning that is often interpreted to mean that upstream constraints do not have downstream impact. However, fact (ii) implies that, if we analyze multiple properties of f_v simultaneously instead of separately, we will observe the "footprint" of the upstream constraint.

In the following section, we perform this analysis empirically and examine how downstream models fine-tuned on upstream models with different objectives, i.e., fairness, experience trade-offs between performance and fairness.

Empirical Evaluation

Our previous result indicates that one can surface the upstream "footprint" by examining the trade-offs that downstream models must make between their own design objectives. In this section, we instantiate constraints as fairness criteria and study the effect of upstream design choices on downstream models, using performance-fairness Pareto frontiers to examine the trade-offs made downstream.

Setup. Mirroring our theoretical analysis, we have a base model f_p trained on dataset D_p and a downstream model f_v fine-tuned on f_p using dataset D_v. Both f_p and f_v are binary classifiers, and they are trained to minimize the binary cross-entropy loss plus a fairness regularization term, moderated by constant α, such that the base model seeks to minimize

$$\mathcal{L}_{\text{BASE}}(f_p, D_p) = \text{BCE}(f_p, D_p) + \alpha_p \cdot \mathcal{R}_{\text{fairness}}(f_p, D_p),$$

and the downstream model seeks to minimize

$$\mathcal{L}_{\text{FT}}(f_v, D_v) = \text{BCE}(f_v, D_v) + \alpha_v \cdot \mathcal{R}_{\text{fairness}}(f_v, D_v).$$

Both the base and fine-tuned models use ResNet18 as their architectures and are trained and tested on images from the Waterbirds dataset (Koh et al. 2021). Waterbirds is a dataset of images that contain either "land" or "water" birds on "land" or "water" backgrounds. Thus, previous works (Koh et al. 2021; Liu et al. 2021; Sagawa* et al. 2020; Shah et al. 2023) have used Waterbirds to study algorithmic fairness since the bird label should not be sensitive to the background label, but the labels are correlated (as water birds are more likely to be on water backgrounds, and vice versa). In our experiment, both the base and fine-tuned models are trained to predict the bird label from the image, with the background as the sensitive attribute.

To analyze fairness, we use three different notions of algorithmic fairness: demographic parity, equalized false positive rate (FPR), and equalized odds. Their formal definitions can be found in (Barocas, Hardt, and Narayanan 2017); they correspond to asking that the positive and negative classification rates for groups differing only by sensitive attribute are the same; the false positive rates across sensitive groups

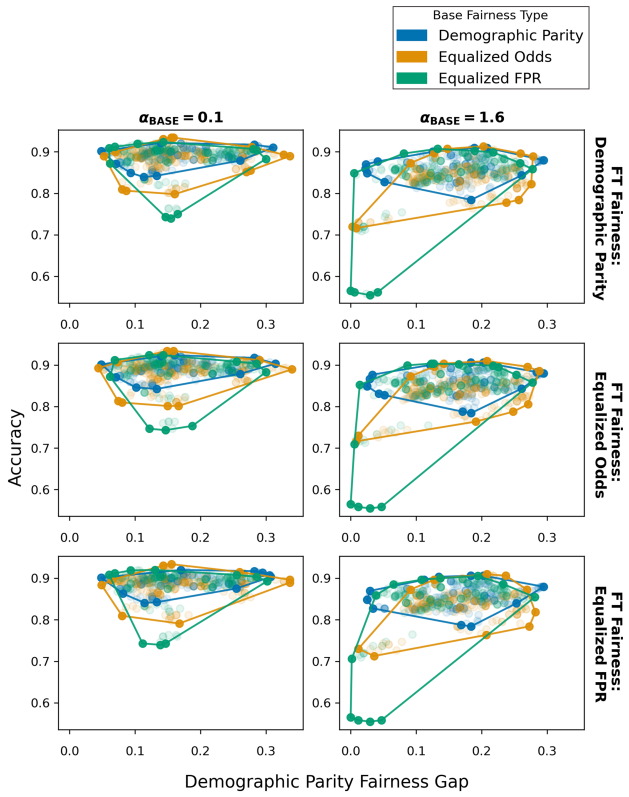


Figure 3: Empirical results for Case Study 2. Every point corresponds to a downstream (fine-tuned) model. On the x-axis, we plot the model's demographic parity fairness gap (the absolute difference between the positive classification rates of the two sensitive groups), and on the y-axis, we plot the model's accuracy. Each column corresponds to a different base model regularization constant (i.e., the strength of the fairness regularization used to train the base model). Each row corresponds to a different type of downstream fairness criterion. The color of each point corresponds to the type of upstream fairness regularizer (used to train the base model). We show the convex hull of the points (that correspond to the same color, or base regularizer) where the top-left portion of the convex hull corresponds to the empirical performance-fairness Pareto frontier. To sweep out the Pareto frontier and elicit downstream models with different performance-fairness characteristics, we vary the fine-tuning regularization constant α_v.

are the same; and finally that both the false and true positive rates are the same; respectively. These results are obtained by training base models with a fairness regularizer before subsequent fine-tuning, adopting an additional fairness constraint. We train each base model under a distinct values of α_p ranging from 0 to 25.6. We then fine-tune each base model on D_v for values of α_p ranging from 0 to 25.6. In total, we trained over 10,000 models. Additional details of implementation can be found in Supplementary Materials.

Results. Figure 3 presents a sampling of our findings, in this case for demographic parity. We plot the accuracy on the y-axis and the fairness gap, where larger values denote greater unfairness, on the x-axis. Each point in the plot corresponds to a fine-tuned model f_v. To obtain a range of fine-tuned models with different performance-fairness characteristics, we vary the fine-tuning regularization constant α_v. The fairness regularizer used to train the base model is denoted by each point's color. Each column corresponds to a different base model regularization constant (i.e., the strength of the fairness regularizer used to train the base model). Each row corresponds to a different type of downstream (fine-tuning) fairness criteria/regularizer.

The plot shows the range of achievable downstream models under different upstream constraints (i.e., the base model's fairness notion). The columns correspond to the strength of the upstream constraint (higher values indicate stronger constraints). The rows correspond to the downstream desideratum (which may not match the upstream one). Although we choose to visualize all points (i.e., all fine-tuned models), the top-left curve of each plot corresponds to the empirical performance-fairness Pareto frontier. Additional plots presenting the results of other fairness interventions, i.e., equalized FPR, can be found in the Supplementary Materials, along with further results for different α_v values.

There are three takeaways from the figure. First, the stronger the upstream constraint, the greater the performance-fairness trade-off on the downstream model, as can be observed by pareto frontier moving down and to the right as we move from left to right within each row. Second, within each column, there is not much variation across rows compared variation across colors within each plot, indicating that the downstream constraint (row) has a limited effect on the Pareto frontier compared to the upstream constraint (color). Third, there is some though limited variation across rows, indicating that the downstream constraint does have some effect on the achievable performance and fairness of the downstream model. These trends are consistent across fairness desiderata.

Conclusion

In this paper, we call attention to the emergence of AI supply chains and offer a model through which to interrogate their implications on learning. Through our case studies, we show that developing machine learning systems across AI supply chains raises challenges that have technical and legal ramifications. Our findings underscore the need for new frameworks and methodologies that account for the emergence of AI supply chains, including improving evaluation metrics, auditing mechanisms, and regulatory oversight. [9]

[9] Additional proofs, extended results, and implementation details are provided in a separate Supplementary Material document at https://osf.io/6b59p/?view_only=11a7ff7d1667478da2ed43bc10bc77eb.

Acknowledgments

We would like to thank Percy Liang, Daniel E. Ho, Helen Nissenbaum, Hongseok Namkoong, Frederike Zufall, Jat Singh, Max Simchowitz, Susan Silbey, and others for fruitful conversations on AI supply chains. The authors gratefully acknowledge funding from the Open Philanthropy Project, National Science Foundation (NSF) grant CNS-1955997 and the Air Force Research Laboratory (AFOSR) grant FA9550-23-1-0301.

References

Attard-Frost, B.; and Hayes, H. A. 2023. Valuing Value Chains: On Canadian AI Regulation, Co-Governance, and the Scope of AI Value Chains. *Regulating Digital (Forthcoming: University of Toronto Press, Eds. Helen A. Hayes & Nicole Goodman)*.

Barocas, S.; Hardt, M.; and Narayanan, A. 2017. Fairness in machine learning. *Nips tutorial*, 1: 2017.

Bengio, Y. 2012. Deep learning of representations for unsupervised and transfer learning. 17–36.

Biggio, B.; Nelson, B.; and Laskov, P. 2012. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*.

Bommasani, R.; Creel, K. A.; Kumar, A.; Jurafsky, D.; and Liang, P. S. 2022. Picking on the same person: Does algorithmic monoculture lead to outcome homogenization? *Advances in Neural Information Processing Systems*, 35: 3663–3678.

Bommasani, R.; Soylu, D.; Liao, T. I.; Creel, K. A.; and Liang, P. 2023. Ecosystem graphs: The social footprint of foundation models. *arXiv preprint arXiv:2303.15772*.

Caruana, R. 1994. Learning many related tasks at the same time with backpropagation. *Advances in neural information processing systems*, 7.

Chandler Jr, A. D. 1969. *Strategy and structure: Chapters in the history of the American industrial enterprise*, volume 120. MIT press.

Chaney, A. J.; Stewart, B. M.; and Engelhardt, B. E. 2018. How algorithmic confounding in recommendation systems increases homogeneity and decreases utility. 224–232.

Chouldechova, A. 2017. Fair Prediction with Disparate Impact: A Study of Bias in Recidivism Prediction Instruments. *Big Data*, 5(2): 153–163.

Cobbe, J.; Veale, M.; and Singh, J. 2023. Understanding accountability in algorithmic supply chains. 1186–1197.

Cybenko, G. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4): 303–314.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A large-scale hierarchical image database. 248–255.

Deng, L. 2012. The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. *IEEE Signal Processing Magazine*, 29(6): 141–142.

Dwork, C.; and Ilvento, C. 2018. Fairness under composition. *arXiv preprint arXiv:1806.06122*.

Ensign, D.; Friedler, S. A.; Neville, S.; Scheidegger, C.; and Venkatasubramanian, S. 2018. Runaway feedback loops in predictive policing. 160–171.

Frank, A. 2010. UCI machine learning repository. *http://archive. ics. uci. edu/ml*.

Ganaie, M. A.; Hu, M.; Malik, A.; Tanveer, M.; and Suganthan, P. 2022. Ensemble deep learning: A review. *Engineering Applications of Artificial Intelligence*, 115: 105151.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep learning*, volume 1. MIT press Cambridge, MA, USA.

Hornik, K.; Stinchcombe, M.; and White, H. 1989. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5): 359–366.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Hu, S.; Fu, Y.; Wu, S.; and Smith, V. 2025. Unlearning or Obfuscating? Jogging the Memory of Unlearned LLMs via Benign Relearning. In *The Thirteenth International Conference on Learning Representations*.

Hutter, F.; Kotthoff, L.; and Vanschoren, J. 2019. *Automated machine learning: methods, systems, challenges*. Springer Nature.

Jayaram, J.; and Vickery, S. 2018. The role of modularity in the supply chain context: Current trends and future research directions.

Khaddaj, A.; Leclerc, G.; Makelov, A.; Georgiev, K.; Salman, H.; Ilyas, A.; and Madry, A. 2023. Rethinking backdoor attacks. 16216–16236.

Kleinberg, J.; and Raghavan, M. 2021. Algorithmic monoculture and social welfare. *Proceedings of the National Academy of Sciences*, 118(22): e2018340118.

Koh, P. W.; Sagawa, S.; Marklund, H.; Xie, S. M.; Zhang, M.; Balsubramani, A.; Hu, W.; Yasunaga, M.; Phillips, R. L.; Gao, I.; et al. 2021. Wilds: A benchmark of in-the-wild distribution shifts. 5637–5664.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

Lee, K.; Cooper, A. F.; and Grimmelmann, J. 2023. Talkin'Bout AI Generation: Copyright and the Generative-AI Supply Chain.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-t.; Rocktäschel, T.; et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33: 9459–9474.

Liu, E. Z.; Haghgoo, B.; Chen, A. S.; Raghunathan, A.; Koh, P. W.; Sagawa, S.; Liang, P.; and Finn, C. 2021. Just train twice: Improving group robustness without training group information. In *International Conference on Machine Learning*.

Lundberg, S. 2017. A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-efficient learning of deep networks from decentralized data. *Artificial intelligence and statistics*, 1273–1282.

Miller, G. A. 1995. WordNet: a lexical database for English. *Commun. ACM*, 38(11): 39–41.

Murphy, K. P. 2012. *Machine learning: a probabilistic perspective*. MIT press.

Pan, S. J.; and Yang, Q. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10): 1345–1359.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. " Why should i trust you?" Explaining the predictions of any classifier. 1135–1144.

Sagawa*, S.; Koh*, P. W.; Hashimoto, T. B.; and Liang, P. 2020. Distributionally Robust Neural Networks. In *International Conference on Learning Representations*.

Salman, H.; Jain, S.; Ilyas, A.; Engstrom, L.; Wong, E.; and Madry, A. 2022. When does Bias Transfer in Transfer Learning? arXiv:2207.02842.

Schölkopf, B. 2002. Learning with kernels: support vector machines, regularization, optimization, and beyond.

Shah, H.; Park, S. M.; Ilyas, A.; and Madry, A. 2023. Modeldiff: A framework for comparing learning algorithms. In *International Conference on Machine Learning*.

Steed, R.; Panda, S.; Kobren, A.; and Wick, M. 2022. Upstream mitigation is not all you need: Testing the bias transfer hypothesis in pre-trained language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Suresh, H.; Tseng, E.; Young, M.; Gray, M.; Pierson, E.; and Levy, K. 2024. Participation in the age of foundation models. 1609–1621.

Wachter, S.; Mittelstadt, B.; and Russell, C. 2017. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harv. JL & Tech.*, 31: 841.

Widder, D. G.; and Nafus, D. 2023. Dislocated accountabilities in the "AI supply chain": Modularity and developers' notions of responsibility. *Big Data & Society*, 10(1): 20539517231177620.

Williamson, O. E. 1975. Markets and hierarchies: analysis and antitrust implications: a study in the economics of internal organization. *University of Illinois at Urbana-Champaign's Academy for Entrepreneurial Leadership Historical Research Reference in Entrepreneurship*.

Xu, H.; Ye, Q.; Yan, M.; Shi, Y.; Ye, J.; Xu, Y.; Li, C.; Bi, B.; Qian, Q.; Wang, W.; et al. 2023. mplug-2: A modularized multi-modal foundation model across text, image and video. In *International Conference on Machine Learning*, 38728–38748. PMLR.

Yosinski, J.; Clune, J.; Bengio, Y.; and Lipson, H. 2014. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27.

Yuan, X.; Pang, T.; Du, C.; Chen, K.; Zhang, W.; and Lin, M. 2024. A Closer Look at Machine Unlearning for Large Language Models. *arXiv preprint arXiv:2410.08109*.

Zhuang, F.; Qi, Z.; Duan, K.; Xi, D.; Zhu, Y.; Zhu, H.; Xiong, H.; and He, Q. 2020. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1): 43–76.