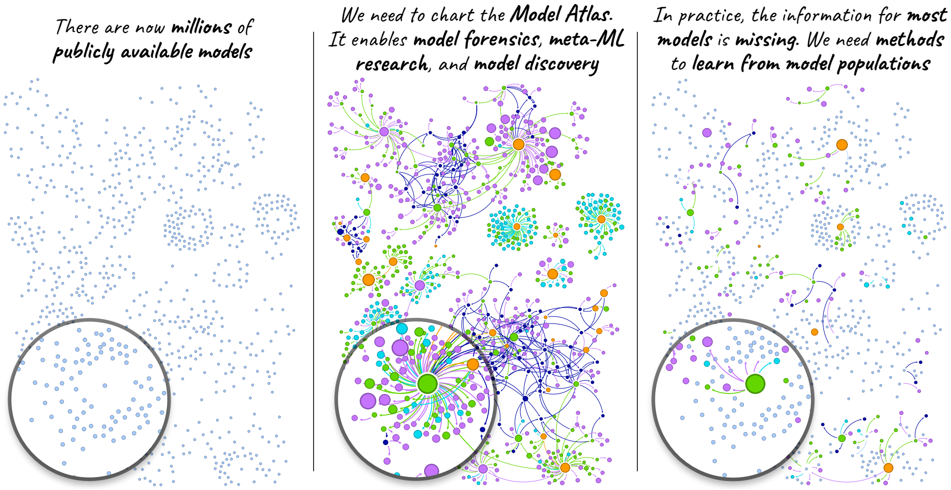
We Should Chart an Atlas of All the World's Models

Eliahu Horwitz **Nitzan Kurer** **Jonathan Kahana** **Liel Amar** **Yedid Hoshen**
School of Computer Science and Engineering
The Hebrew University of Jerusalem, Israel
https://horwitz.ai/model-atlas


There are now **millions** of **publicly available models**

We need to chart the **Model Atlas**. It enables **model forensics**, **meta-ML research**, and **model discovery**

In practice, the information for **most models** is **missing**. We need **methods** to **learn from model populations**

Figure 1: ***Position overview:*** With millions of public models, it becomes important to move beyond individual models and study entire populations (left). The Model Atlas formalizes this shift by representing models as nodes in a graph, with directed edges denoting weight transformations (e.g., fine-tuning). Node size and color, as well as edge color, encode node and edge-level features; light blue indicates missing or unknown information. The atlas enables a range of applications, including model forensics, meta-ML research, and model discovery (center). In practice, most edges and features are unknown. This motivates ML methods that take models as input and infer their properties, thereby completing the missing atlas regions (right). **Zoom in to view edges, best viewed in color.**

Abstract

Public model repositories now contain millions of models, yet most remain undocumented and *effectively lost*: their capabilities, provenance, and constraints cannot be reliably determined. As a result, the field wastes training time and compute, propagates hidden biases, faces intellectual-property risks, and misses opportunities for model reuse and transfer. In this position paper, we advocate charting the world's model population in a unified structure we call the *Model Atlas*: a graph that captures models, their attributes, and the weight transformations connecting them. The Model Atlas enables applications in model forensics, meta-ML research, and model discovery, challenging tasks given today's unstructured model repositories. However, because most models lack documentation, large atlas regions remain uncharted. Addressing this gap motivates new machine learning methods that treat models themselves as data and infer properties such as functionality, performance, and lineage directly from their weights. We argue that a scalable path forward is to bypass the unique parameter symmetries that plague model weights. Charting all the world's models will require a community effort, and we hope its broad utility will rally researchers toward this goal.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Position Paper Track.

1 Introduction

Many scientific breakthroughs occurred when researchers transitioned from observing individual samples to studying entire populations. Consider Darwin's transition from cataloging specimens to understanding evolution through natural selection acting on populations, or modern medicine's move from anecdotal patient outcomes to large-scale clinical trials establishing treatment efficacy. Machine learning (ML) has mirrored this shift for data, replacing hand-crafted features derived from a few observations with representations learned from large sample collections. This paradigm has fueled remarkable progress in many domains like computer vision (CV), natural language processing (NLP), and audio. However, we argue that the ML community has yet to fully apply this population-level perspective to its outputs: the models themselves. While we optimize individual models, we have largely ignored the implicit knowledge embedded across the growing population of trained models. **In this position paper, we advocate for systematically studying entire model populations, and argue that this requires charting them in a unified structure, the *"Model Atlas"*.** We present a schematic overview of this position in Fig. 1.

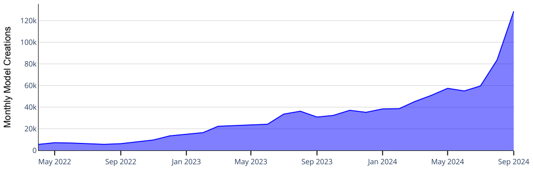


Figure 2: ***Growth in Hugging Face models:*** The number of public models is growing rapidly, but most remain undocumented and effectively lost. We advocate for charting them in a Model Atlas.

This direction is particularly timely: the number of publicly available models is growing at an unprecedented rate. Platforms like Hugging Face (HF) now host over 1.5 million models, with more than 100,000 added each month (see Fig. 2). Yet most models remain poorly documented and *effectively lost*: one cannot determine their capabilities (e.g., evaluations, failure modes), provenance (e.g., parent models, training data), or usage constraints (e.g., license, safety). As a result, the community cannot reliably discover models, attribute credit, reproduce results, or reuse them with confidence. We therefore need to chart the world's models, mapping the machine learning landscape. Concretely, we define the *Model Atlas* as a graph in which nodes represent models and directed edges denote weight transformations between them (e.g., fine-tuning, quantization, merging). Nodes and edges are labeled with attributes such as task performance, training data, and optimization details. The atlas thus transforms model collections from a flat list of models to an interconnected ecosystem, enabling a range of applications, e.g., model forensics, meta-ML research, and model discovery.

Model forensics allows us to trace a model's training trajectory, its origins, the data used, and the transformations it underwent [4, 20, 21, 43, 61, 63, 64]. This has major implications for intellectual property, bias estimation, safety, and reproducibility. The atlas also enables intuitive visualizations of huge model populations, facilitating *"meta-ML research"* to analyze the evolving ML landscape. Concretely, we demonstrate that the atlas can reveal structural patterns, emerging trends, and opportunities to transfer knowledge between communities. For instance, in Fig. 3 we present a preliminary atlas which reveals distinct structural patterns across modalities, Llama-based models [10] have more diverse and complex training dynamics (e.g., quantization and model merging) than Stable Diffusion [42]. Finally, with so many models available, we need tools for accurate model discovery [8, 23, 29, 32, 37, 53, 62, 65] which could reduce training costs, shorten development cycles, and lower environmental impact. Model discovery requires knowing what each model does and how well it performs, information that the atlas captures. While related to prior work (e.g., Pal et al. [39]), our position focuses specifically on the Model Atlas and the methods required to chart it.

Although many models exist, most are undocumented and contain little to no usable metadata. Over 60% of models on HF have no documentation at all [19, 21, 26], and for the remainder, much is incomplete. This includes many of the platform's most popular models, which often omit key details such as training data, performance, or lineage. Moreover, public repositories represent only a fraction of the global model landscape; private repositories in companies and research labs contain many more models. Consequently, the Model Atlas remains mostly uncharted, populated by "lost models", whose origins, capabilities, and interrelations are largely unknown. Completing the atlas requires inferring its missing nodes, edges, and their attributes. This motivates a new class of ML problems that treat models themselves as data. However, learning directly from models presents major challenges: weights are high-dimensional, exhibit harmful permutation symmetries, and lack established structural priors. To address this, the emerging field of weight-space learning [11, 39, 44, 45, 47] develops

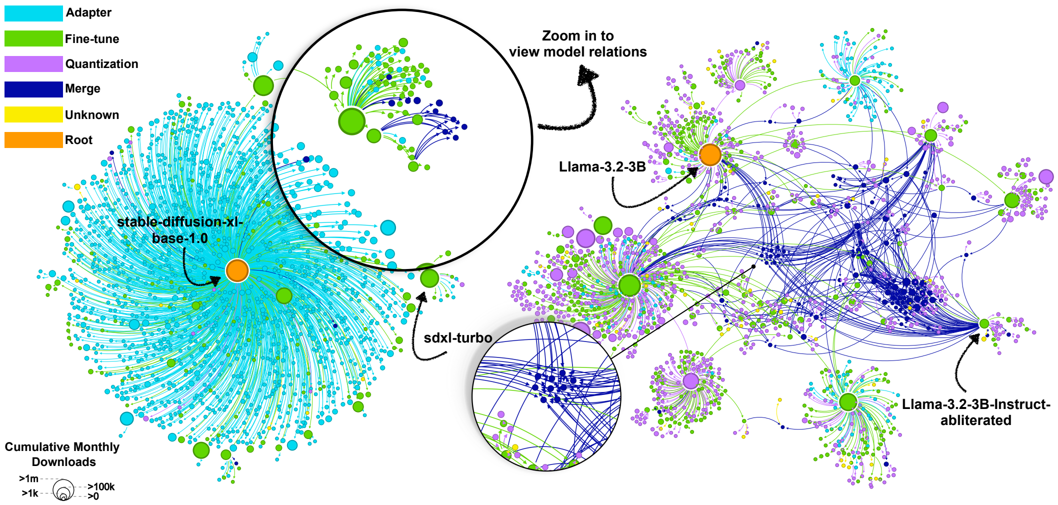


Figure 3: ***The Model Atlas - Stable Diffusion vs. Llama:*** We visualize the atlas of the top 30% most downloaded models in the Stable Diffusion (SD) and Llama regions. Node size reflects cumulative monthly downloads, and color denotes the transformation type relative to the parent model. The atlas reveals that the Llama region has a more complex structure and a wider diversity of transformation techniques (e.g., quantization, merging) compared to SD. *Zoom in to view edges, best viewed in color.*

specialized neural networks that take other models as input and infer their properties. The dominant approach involves designing *equivariant networks*, which account for the permutation symmetries of model weights. While promising, current methods remain computationally expensive and are mostly limited to small-scale models or simplified settings.

We argue that scaling atlas charting to large, diverse model populations requires rethinking the role of weight symmetries. Instead of tackling permutation invariance directly, we propose a path that sidesteps it altogether. By avoiding symmetry constraints, we can leverage standard architectures and training recipes, enabling more efficient and scalable methods. We highlight concrete scenarios where this is already feasible, such as probing models for their functional responses or learning on weights within symmetry-consistent subsets. While these strategies are still in their early stages, they demonstrate the practicality of symmetry-agnostic approaches. Realizing the full potential of the Model Atlas will require new research into priors, architectures, and learning methods. We believe this is both a timely and worthwhile effort that would benefit broad community participation.

2 The Model Atlas

We aim to represent model collections in a way that captures each model's lineage, creation process, functional capabilities, performance, and metadata. We therefore propose the *Model Atlas*, a directed acyclic graph (DAG) denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each node $v \in \mathcal{V}$ represents a model at a specific point in its training trajectory. Directed edges $(u, v) \in \mathcal{E}$ denote a weight transformation from model u to model v, e.g., fine-tuning or quantization.

Both nodes and edges have associated features. Node features include all the information about a model, e.g., its weights, functional traits, and metadata. Traits capture the functional and behavioral properties of a model, such as accuracy, robustness, and fairness. Unlike traits, which reflect learned behaviors, metadata captures static or descriptive attributes such as creation time and license type. Edge features capture information about the transformation from parent model u to child model v, e.g., optimization parameters, training data, and transformation type (e.g., LoRA [22] or quantization).

This graph contains virtually all information about the model population, including how each model was created, its relation to other models, and its capabilities and metadata (see Fig. 4 for an overview). The atlas provides the basis for a wide range of applications that we describe in Sec. 3. However, as we describe in Sec. 4, the core obstacle is that only a small part of this graph is known; for most nodes, only a very small portion of the details about their edges, node features, or edge features are given. To unlock the rich rewards of the Model Atlas, we believe it is paramount to develop new ML tools for recovering the rest of the graph.

3 Why is the Model Atlas useful?

The Model Atlas is central to our position, as it provides a structured way to represent model collections. Its utility stems from the many applications that it enables. We group these into three main categories: model forensics, meta-ML research, and model discovery. Note, some applications rely on access to the atlas's structure and contents, which are mostly unknown in practice. In Sec. 4 we describe scalable methods for charting the missing regions of the atlas.

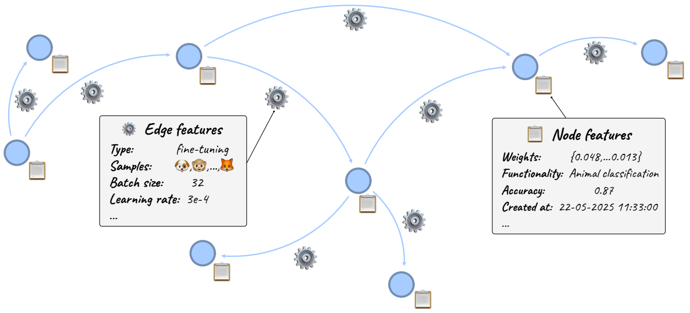


Figure 4: ***Model Atlas illustration:*** Each node in the Model Atlas represents a distinct model state, and each directed edge denotes a weight transformation from one model to another. Edge features encode information about the transformation, node features capture properties of the model itself.

3.1 Model forensics

Neural networks are complex, opaque functions. They do not readily reveal details about their training data or predecessor models. However, this information is critical, particularly for the creators of training datasets or upstream models. By charting models within the atlas, we can infer such provenance relationships. The atlas facilitates this in several ways. First, a directed path between two models indicates which model is descended from the other. Second, if a model descends from others, it implicitly inherits their training data. Since edge features include information about the data used during transformation, the atlas can reveal the full data lineage of a model, an important capability for intellectual property and compliance. The same ideas also extend to bias estimation. If an ancestor exhibits a known bias, we can propagate that knowledge downstream. Finally, the atlas improves reproducibility by recording the exact parameters used to transform one model into another.

3.2 Meta-ML research by visualizing entire model collections

The Model Atlas enables large-scale visualization and analysis of model repositories, a capability we refer to as *"meta-ML research"*. By encoding node and edge features into visual attributes, such as color and size, these visualizations[1] provide an interpretable overview of structure, model attributes, and emerging trends across vast model populations. Note that node position is optimized for clarity [24] and does directly reflect distance between model weights. To demonstrate this, we analyze Hugging Face (HF), the largest public model repository, which hosts over 1.5 million models, with more than 100,000 added each month. While HF encourages documentation via Model Cards [34], over 50% of models lack them entirely. Additionally, fewer than 30% of models specify their parent model. Using these, we construct an *initial* atlas and explore recent trends. In Fig. 5, we visualize a portion of this atlas, comprising over 60,000 models across 28 connected components and more than 65,000 edges. We use this structure to compare development patterns in computer vision (CV) and natural language processing (NLP). See App. A for further visualizations and analysis. We will make the Model Atlas used for these visualizations publicly available through an interactive web interface that allows users to navigate and explore the different models.

Quantization. As shown in Fig. 5, quantization is rare in CV models (fewer than 0.15% of all models in this pool) compared to NLP. This suggests that vision models have not yet reached the scale where inference cost necessitates quantization. However, using the atlas we can spot an emerging change of trend. The presence of a highly downloaded quantized version of Flux [28], one of the largest generative vision models, indicates that image generation models may have just reached the scale where quantization is valuable or that quantization is harder to achieve effectively for CV models.(see zoom-in (A) in Fig. 5). An alternative explanation may be that quantization is harder to achieve effectively for CV models.

Fine-tuning vs. adapters. Generative and discriminative models exhibit markedly different adaptation strategies. Most generative models rely on parameter-efficient adapters (e.g., LoRA), whereas

[1] We use "Gephi" by Bastian et al. [3] to visualize the Model Atlas.

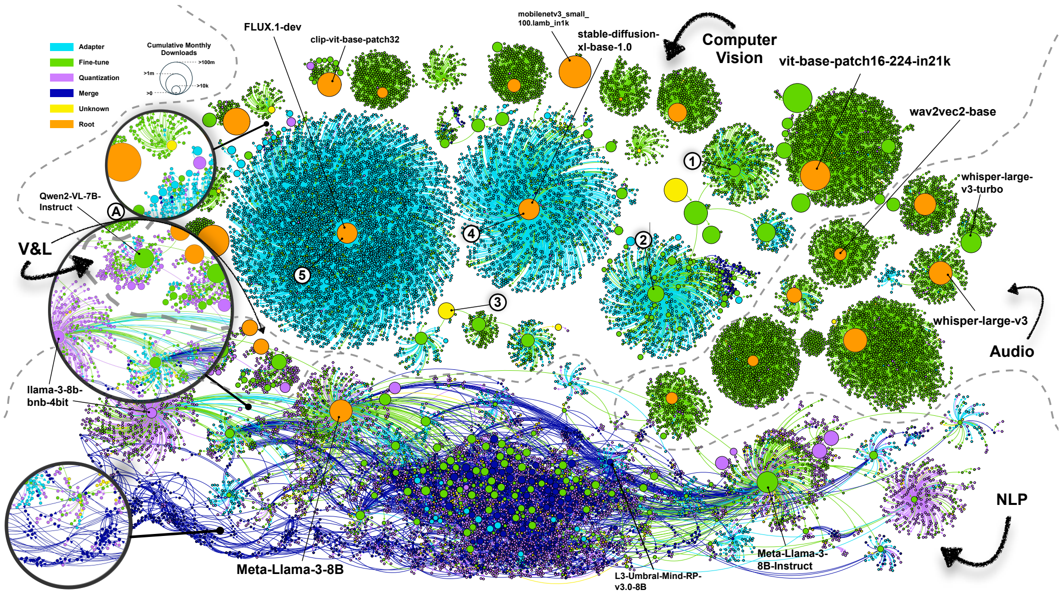


Figure 5: ***The Hugging Face Model Atlas:*** While this is a small subset (63,000 models) of the documented regions of Hugging Face, it already reveals significant trends. *Depth and structure.* The LLM connected component (CC) is deep and complex. It includes almost a third of all models. In contrast, while Flux is also substantial, its structure is much simpler and more uniform. *Quantization.* Zoom-in (A) highlights quantization practices across vision, language, and vision-language (V&L) models. Vision models barely use quantization, despite Flux containing more parameters (12B) than Llama (8B). Conversely, quantization is commonplace in LLMs, constituting a large proportion of models. VLMs demonstrate a balance between these extremes. *Adapter and fine-tuning strategies.* A notable distinction exists between discriminative (top) and generative (bottom) vision models. Discriminative models primarily employ full fine-tuning, while generative models have widely adopted adapters like LoRA. The evolution of adapter adoption over time is evident: Stable-Diffusion 1.4 (SD) (1) mostly used full fine-tuning, while SD 1.5 (2), SD 2 (3), SD XL (4), and Flux (5) progressively use more adapters. Interestingly, the atlas reveals that audio models rarely use adapters, suggesting gaps and opportunities in cross-community knowledge transfer. This inter-community variation is particularly evident in *model merging*. LLMs have embraced model merging, with merged models frequently exceeding the popularity of their parents. This raises interesting questions about the limited role of merging in vision models. For enhanced visualization, we display the top 30% most downloaded models. **The image is high-resolution, zoom in to view individual nodes and edges. Best viewed in color.**

discriminative models almost exclusively use full fine-tuning (see Fig. 5-top). The Model Atlas not only highlights this distinction but also reveals how these trends are shifting over time. In earlier generations, such as SD1.4, only around 50% of models used adapters. In contrast, newer generative models like Flux [28] and Llama 3 [10] overwhelmingly adopt adapter-based methods, indicating a broader move toward more efficient fine-tuning (see pins (1)-(5) in Fig. 5).

Merging. Model merging [50, 52, 56, 58, 59] is about 35 times more common in NLP than in CV. On average, merged NLP models achieve 30% higher influence (measured via descendant downloads; see App. A) than their non-merged siblings. While this does not imply causation, it suggests that merging is an underexplored strategy in vision.

3.3 Model discovery

Training new models is time-consuming, expensive, and energy-intensive. Despite this, practitioners often train or fine-tune a new model for each task. Imagine that instead, one could retrieve a suitable pre-trained model and use it directly. This would reduce costs, shorten development cycles, and lower environmental impact. The atlas provides the infrastructure needed for such model discovery. It enables search based on model capabilities, performance, or transformation history. This goes far beyond search functionality in existing repositories, which mostly rely on textual search of sparse documentation. For example, although tens of thousands of models are trained on ImageNet [7], searching for the class "peacock" among the $< 1.5M$ HF models returns fewer than 100 results. In contrast, an atlas-based search, leveraging traits and lineage, would yield many more relevant results.

If no existing model fits a given task, the Model Atlas can still assist by helping identify models, or sets of models, likely to transfer well. Estimating transferability [8, 23, 29, 32, 37, 53, 62, 65] is computationally expensive, as existing methods typically require evaluating each model on the target dataset. In contrast, a complete atlas enables graph-based search strategies that reduce the number of models that must be tested. For example, a branch-and-bound strategy could prune subgraphs unlikely to yield high transfer performance. Similarly, multi-scale search techniques, akin to binary search, could accelerate retrieval by narrowing the candidate space efficiently. We believe that a complete Model Atlas will enable scalable transferability estimation and more effective model reuse.

4 The atlas is incomplete, we need new methods to chart it

4.1 The atlas is incomplete

Despite the abundance of models, most are undocumented, leaving little usable information. To address this, solutions like Model Cards [34] were introduced to provide standardized descriptions. While an important first step, current documentation practices fall short of handling the scale and complexity of modern model ecosystems. As highlighted earlier, over 60% of models on HF lack *any* model card. Worse still, even documented models typically provide only sparse or inconsistent information. For instance, fewer than 15% of HF models report accuracy, and only 8% include license details (see Fig. 6). Since this data is self-reported, its quality varies widely. Consequently, recent attempts to summarize documentation automatically (e.g., by HF [55]) face fundamental limitations: missing information cannot be recovered through summarization alone. Overall, current practices lack the required coverage, consistency, and adaptability.

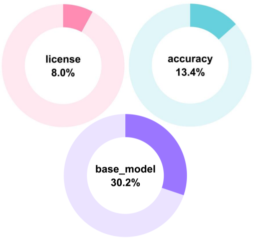


Figure 6: ***Documentation levels in HF:*** Most models on Hugging Face suffer from poor documentation quality.

This documentation failure has created a vast population of "lost models", models whose origins, capabilities, and relationships are largely unknown. This has tangible consequences. Organizations routinely waste resources re-training models that already exist but are undiscoverable. Moreover, even failed models contain valuable insights into architecture, data, and optimization; this implicit knowledge is currently lost, forcing the community to rediscover it. The atlas helps preserve this implicit knowledge and serves as a tool for knowledge retention.

4.2 Charting the Model Atlas

Although we advocate the Model Atlas as a foundational structure, much of it remains missing. Completing the atlas requires inferring its missing nodes, edges, and their attributes. This motivates a new class of ML problems where models (or sets of models) serve as input, and the goal is to predict properties of the atlas. While ML for images, text, and audio is well-established, learning directly from model weights is only starting. This presents two key challenges. First, model weights are high-dimensional, making them computationally difficult to process. Second, neural network weights exhibit parameter space symmetries [17]. For instance, permuting neurons in hidden layers of an MLP does not affect the model's function, but does change the raw weight representation. This property complicates the development of techniques that generalize across diverse model populations, as most ML methods do not naturally account for these symmetries.

The emerging field of studying models based on their weights is called Weight-Space Learning [11, 44, 45, 47, 54]. Research in this area generally falls into 3 categories: i) *Learning weight representations*, recovering model functionality and performance [18, 19, 25–27, 30, 35, 36, 41, 43, 68]. ii) *Developing generative models for weights*, synthesizing of new model parameters [2, 9, 12, 16, 40, 48, 59]. And iii) *Recovering weight trajectories*, tracing the evolution of model parameters over time [4, 5, 20, 21, 57, 60, 61, 63, 64]. Despite promising progress, most work in this space is limited to synthetic or simplified settings. Current approaches cannot yet handle many of the subtasks and scale involved in charting real-world model populations.

Avoiding, not embracing equivariance. The dominant approach in weight-space learning is to design equivariant networks [27, 30, 31, 35, 36, 41, 51, 66–68] that respect the permutation symmetries of neural weights. This is an elegant but computationally costly endeavor, requiring specialized adaptations for different layer types (e.g., convolutional, self-attention, or state-space). Unfortunately, standard deep learning architectures are not equivariant by default, and current solutions remain too expensive for web-scale model repositories. Rather than redeveloping architecture classes from scratch, we argue for an alternative path: reformulating tasks to avoid permutation symmetry altogether. This allows us to use standard architectures and training tricks. While not all weight-space learning tasks admit such reformulations, many graph-based tasks do.

Table 1: ***Atlas-based documentation imputation:*** Using atlas structure improves prediction of model accuracy and other attributes, compared to naively using the majority label. In (b), we report the prediction accuracy.

(a) Metric Imputation Results

	Method	MSE	MAE	Corr.
TruthfulQA (0-shot, mc2)	Baseline	100.217	8.541	-
	Ours 1-NN	32.830	3.247	0.856
	Ours 2-NN	28.720	3.235	0.864
	Ours 3-NN	25.544	3.147	0.877
	Ours 5-NN	23.512	3.093	0.885
Helleswag (0-shot)	Baseline	95.000	7.500	-
	Ours 1-NN	30.000	3.000	0.860
	Ours 2-NN	27.000	2.900	0.870
	Ours 3-NN	24.000	2.800	0.880
	Ours 5-NN	22.000	2.700	0.890

(b) Attribute Prediction Accuracy

Attribute	Graph Avg.	Hub Avg.
`pipeline_tag`	0.60	0.79 (+0.19)
`library_name`	0.81	0.84 (+0.02)
`model_type`	0.66	0.81 (+0.15)
`license`	0.49	0.85 (+0.35)
`relation_type`	0.61	0.80 (+0.19)

Graph kNN. One simple approach is to predict properties of unlabeled models based on their neighbors in the partially charted atlas. For example, we used the Mistral-7B connected component (17.5k models, with only 300 labeled on the *TruthfulQA (0 shot) (mc2)* metric; see App. E for more details). We estimated performance for each unlabeled node by averaging the scores of its k nearest neighbors, where distance is defined by path length in the undirected atlas. Tab. 1a shows that this simple method can predict model accuracy surprisingly well. Beyond accuracy, the atlas can also help infer missing metadata. We use the concept of *model hubs*, defined as sets of sibling leaf nodes (79% of models belong to some hub). Assuming models in the same hub are similar, we impute missing attributes using majority voting across labeled nodes. We tested hub-based predictions on five attributes that are often missing on HF, including: license, model and inheritance types (see Fig. 6). Compared to global majority baselines, hub-based prediction significantly improves accuracy, by 35% for license prediction and 19% for both inheritance type and pipeline tag (Tab. 1b). These methods are simple and scalable, though not always sufficiently precise and limited by the completeness of the known atlas.

Learning on functional features. Permutation symmetry affects the parameter space, not the function space. One natural workaround is to represent a model via its responses to predefined inputs. Probing-based methods [18, 25, 26] select a set of m probes (inputs) $x_1, \ldots, x_m$ and represent the model as the concatenation of its responses:

$$\phi(f) = (f(x_1), f(x_2), \ldots, f(x_m))$$

As long as the responses exhibit structure, we can learn downstream tasks using standard architectures (see Fig. 7-left). Similarly to a good examiner, the key is to know which questions to ask. While determining the optimal set of probes is an exciting, outstanding challenge, Kahana et al. [25] showed that even quite native probes performed better than many state-of-the-art methods. The main limitation of probing methods is their runtime, as computing their representations requires m forward passes per model, which can become costly at scale.

Learning directly on weights within a connected component. A core motivation of equivariant architectures is that weights suffer from permutation symmetries. But it turns out that this is not always the case. A child model keeps mostly the same neuron ordering as its parent model, as finetuning only changes weights very slightly. Recursively, all models fine-tuned from a foundation model will typically have the same permutation. More formally, all models within the same connected component in the Model Atlas have consistent neuron ordering. In such cases, we can apply standard architectures directly to weights without equivariance (see Fig. 7-right). The outstanding challenge is to develop architectures able to deal with the extremely high dimension of network weights (millions to billions). While current architectures [19] use factorized classifiers, taking in a single model layer, we believe better architectures, that span multiple layers, will be better.

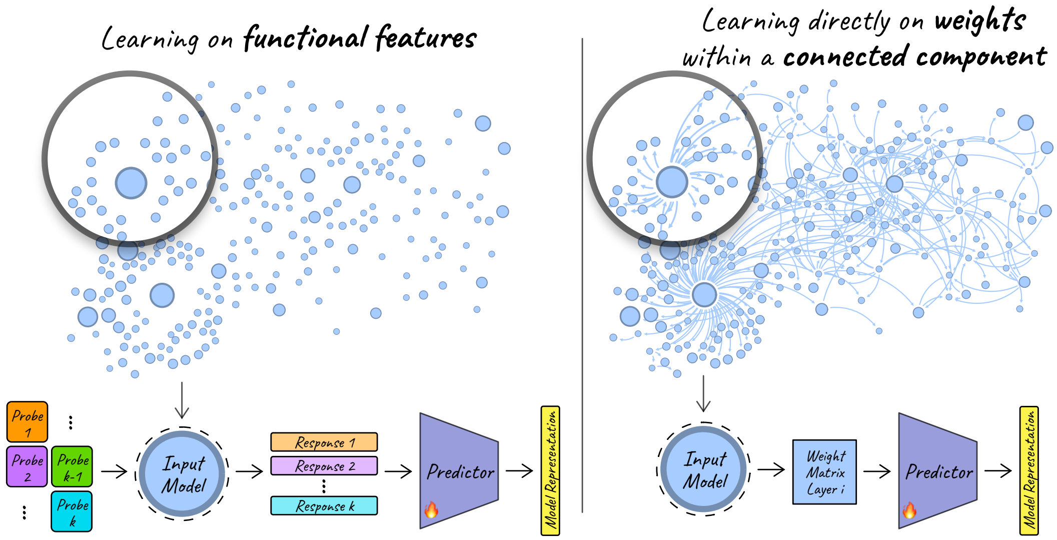


Figure 7: ***Equivariance-bypassing strategies:*** When working with a general collection of models, one can probe each model and train a model on its responses, thereby sidestepping the challenge of weight permutations (left). In contrast, within a connected component of the atlas, neuron permutations are mostly fixed, allowing us to train standard architectures directly on the weight matrices of individual layers (right). This significantly reduces computational cost. In both cases, these strategies support scalable charting of large, in-the-wild populations. The fire symbol indicates the learned component.

Working within the same connected component allows for simpler tools, as [21] demonstrated, computing a node's nearest neighbors provides a clue to what nodes are connected to it by an edge, allowing link prediction. Clustering models by their weights also reveals other shared attributes [15, 21]. We extend this insight by introducing a link prediction process that incorporates real-world structural priors extracted with the help of meta-ML research using visualization of HF. By analyzing patterns such as quantization, duplication, temporal upload dynamics, and checkpoint structures, we develop simple yet effective decision rules that guide edge prediction. These priors allow us to move beyond generic tree-based heuristics and chart realistic DAGs while significantly improving the accuracy at a lower computational cost. See App. C for full implementation and results.

5 Open challenges in Model Atlas charting

Charting the full atlas will require new research into priors, architectures, and learning methods. We now outline key open challenges; see App. B for further discussion.

New visualization methods. In this paper, we visualized the Model Atlas using off-the-shelf graph layout algorithms [3, 24]. While these are already useful for meta-ML research, they are not designed for the unique structural and semantic characteristics of Model Atlases. Future visualization methods could exploit specific structural priors, such as large hubs and near tree-like hierarchies. Additionally, incorporating node and edge features (e.g., model metadata, weight distance metrics, or performance estimates) could help position semantically related models closer together. Moreover, most existing graph layouts also assume static graphs, whereas Model Atlases evolve continuously. Supporting dynamic insertions of nodes and edges will require new online layout algorithms.

Charting priors and weight-space learning architectures. The approaches highlighted in Sec. 3 exploit specific patterns and priors that allow them to bypass the challenge of weight permutation. However, given the scale of modern model repositories and the size of individual models, extending charting efforts to the web scale will likely require new priors and architectures. In particular, future work may explore downsampling strategies that retain essential semantic information, enabling efficient processing of large model populations. Most current methods also assume access to model weights. Extending charting to black-box settings, such as proprietary APIs like ChatGPT or Gemini [14, 38], will require new techniques based on outputs, activations, or metadata.

Recovering training trajectories. Most current weight-space learning methods focus on node-level properties such as function or performance. However, recovering edge-level attributes, e.g., optimizer, data, or learning dynamics, is equally important. Existing methods often assume single-parent transitions and do not account for more complex scenarios such as model merging or distillation. Extending charting algorithms to account for multi-parent edges, distillation relationships, and non-weight-based transformations remains a largely unexplored challenge. More broadly, current atlases can be viewed as sparse samples of a larger underlying atlas capturing the continuous training trajectory. In most cases, edges in the atlas represent transitions between model states that may have occurred through multiple intermediate optimization steps. Recovering this full, fine-grained Model Atlas, including the intermediate state, presents a compelling and open challenge for future research.

Probe selection methods. Probing-based approaches provide a powerful way to represent models functionally, but they involve a tradeoff: using too many probes increases runtime, while too few increases sensitivity to probe choice. To overcome this, we need new methods for selecting a minimal yet expressive set of probes. This could involve real input probes (e.g., curated data points), requiring principled selection strategies, or synthetic probes, which call for generative or optimization-based techniques. Designing robust, task-adaptive probing schemes is an open problem.

6 Alternative view

While we strongly advocate for charting model collections through the Model Atlas, it is important to acknowledge valid alternative perspectives.

Foundation models may render model collections obsolete. One view is that the rise of increasingly capable foundation models, such as ChatGPT and Gemini [14, 38], may reduce the relevance of smaller, task-specific models. If this consolidation continues, some might argue that model repositories will shrink or become unnecessary, thereby reducing the importance of charting them. While reasonable, this perspective does not align with current trends. In practice, the number of public models continues to grow. Moreover, even a small number of dominant models would produce numerous variants, checkpoints, and fine-tuned descendants. These models would still benefit from organization, provenance tracking, and comparison, tasks the atlas is ideal for.

Equivariant methods may eventually scale. Our position is that equivariant methods will struggle to scale to full repositories. A valid alternative view is that given sufficient research effort, the field will find the right architecture, training tricks, and hyperparameters to make them efficient alternatives. While this is a different technical approach from the one we advocate, this would not change the formulation and applications of the atlas, which is the essential part of our position. Furthermore, this is unlikely to help within connected components where there are no permutation symmetries.

Charting may not require machine learning. A valid argument is that charting the Model Atlas could be addressed by simply tightening documentation requirements, eliminating the need for new machine learning methods. Under this approach, model creators would be required to upload all relevant information alongside the model or embed it directly within the weights file. While such protocols may eventually be adopted, they are unlikely to capture all the information we might want about a model. They would also have no effect on the millions of existing models that remain undocumented. Finally, we draw some parallels with code documentation, which is good practice and encouraged by all, but is rarely done. We believe forcing model creators to document is doomed to fail.

7 Conclusion

The rapid growth in the number of trained models presents an opportunity to study them collectively rather than in isolation. The Model Atlas formalizes this perspective by representing models, their attributes, and the transformations that link them in a unified structure. We have outlined its potential applications in model forensics, meta-ML research, and model discovery. The central challenge is that most edges in the graph, as well as the majority of node and edge features, are missing. Addressing this gap calls for a new class of machine learning methods that treat neural networks as data and infer meaningful properties from them. We argue that the most promising approaches are those that sidestep the symmetries inherent in weight space, enabling scalable solutions across diverse model populations. We hope that the Model Atlas will inspire broader research into understanding and organizing the global population of models.

Acknowledgements

This work was supported in part by the "Israel Science Foundation" (ISF), the "Council for Higher Education" (Vatat), the "Center for Interdisciplinary Data Science Research" (CIDR), the "Israeli Cyber Authority", and "KLA".

References

[1] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002. 19

[2] Maor Ashkenazi, Zohar Rimon, Ron Vainshtein, Shir Levi, Elad Richardson, Pinchas Mintz, and Eran Treister. Nern–learning neural representations for neural networks. *arXiv preprint arXiv:2212.13554*, 2022. 6

[3] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: an open source software for exploring and manipulating networks. In *Third international AAAI conference on weblogs and social media*, 2009. 4, 8, 20

[4] Nicholas Carlini, Daniel Paleka, Krishnamurthy Dj Dvijotham, Thomas Steinke, Jonathan Hayase, A Feder Cooper, Katherine Lee, Matthew Jagielski, Milad Nasr, Arthur Conmy, et al. Stealing part of a production language model. *arXiv preprint arXiv:2403.06634*, 2024. 2, 6

[5] Jialuo Chen, Jingyi Wang, Tinglan Peng, Youcheng Sun, Peng Cheng, Shouling Ji, Xingjun Ma, Bo Li, and Dawn Song. Copy, right? a testing framework for copyright protection of deep learning models. In *2022 IEEE symposium on security and privacy (SP)*, pages 824–841. IEEE, 2022. 6

[6] Derek J. de Solla Price. Networks of scientific papers. *Science*, 149(3683):510–515, 1965. 19, 20

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6

[8] Nan Ding, Xi Chen, Tomer Levinboim, Soravit Changpinyo, and Radu Soricut. Pactran: Pac-bayesian metrics for estimating the transferability of pretrained models to classification tasks. In *European Conference on Computer Vision*, pages 252–268. Springer, 2022. 2, 6

[9] Amil Dravid, Yossi Gandelsman, Kuan-Chieh Wang, Rameen Abdal, Gordon Wetzstein, Alexei A Efros, and Kfir Aberman. Interpreting the weight space of customized diffusion models. *arXiv preprint arXiv:2406.09413*, 2024. 6

[10] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. 2, 5

[11] Gabriel Eilertsen, Daniel Jönsson, Timo Ropinski, Jonas Unger, and Anders Ynnerman. Classifying the classifier: dissecting the weight space of neural networks. *arXiv:2002.05688*, 2020. 2, 6

[12] Ziya Erkoç, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. Hyperdiffusion: Generating implicit neural fields with weight-space diffusion. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 14300–14310, 2023. 6

[13] Damian Falk, Léo Meynent, Florence Pfammatter, Konstantin Schürholt, and Damian Borth. A model zoo of vision transformers. *arXiv preprint arXiv:2504.10231*, 2025. 16

[14] Google DeepMind. Gemini. `https://gemini.google.com`, 2023. Accessed: 2025-05-22. 8, 9

[15] Almog Gueta, Elad Venezian, Colin Raffel, Noam Slonim, Yoav Katz, and Leshem Choshen. Knowledge is a region in weight space for fine-tuned language models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. 8, 17

[16] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016. 6

[17] Robert Hecht-Nielsen. On the algebraic structure of feedforward network weight spaces. In *Advanced Neural Computers*, pages 129–135. Elsevier, 1990. 6

[18] Vincent Herrmann, Francesco Faccio, and Jürgen Schmidhuber. Learning useful representations of recurrent neural network weight matrices. In *Forty-first International Conference on Machine Learning*, 2024. 6, 7

[19] Eliahu Horwitz, Bar Cavia, Jonathan Kahana, and Yedid Hoshen. Representing model weights with language using tree experts. *arXiv preprint arXiv:2410.13569*, 2024. 2, 6, 7, 16, 17

[20] Eliahu Horwitz, Jonathan Kahana, and Yedid Hoshen. Recovering the pre-fine-tuning weights of generative models. In *Forty-first International Conference on Machine Learning*, 2024. 2, 6, 14

[21] Eliahu Horwitz, Asaf Shul, and Yedid Hoshen. Unsupervised model tree heritage recovery. In *The Thirteenth International Conference on Learning Representations*, 2025. 2, 6, 8, 16, 17, 18, 19, 20

[22] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 3

[23] Long-Kai Huang, Junzhou Huang, Yu Rong, Qiang Yang, and Ying Wei. Frustratingly easy transferability estimation. In *International conference on machine learning*, pages 9201–9225. PMLR, 2022. 2, 6

[24] Mathieu Jacomy, Tommaso Venturini, Sebastien Heymann, and Mathieu Bastian. Forceatlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software. *PloS one*, 9(6): e98679, 2014. 4, 8

[25] Jonathan Kahana, Eliahu Horwitz, Imri Shuval, and Yedid Hoshen. Deep linear probe generators for weight space learning. In *The Thirteenth International Conference on Learning Representations*, 2025. 6, 7

[26] Jonathan Kahana, Or Nathan, Eliahu Horwitz, and Yedid Hoshen. Can this model also recognize dogs? zero-shot model search from weights. *arXiv preprint arXiv:2502.09619*, 2025. 2, 7, 16

[27] Miltiadis Kofinas, Boris Knyazev, Yan Zhang, Yunlu Chen, Gertjan J Burghouts, Efstratios Gavves, Cees GM Snoek, and David W Zhang. Graph neural networks for learning equivariant representations of neural networks. *arXiv preprint arXiv:2403.12143*, 2024. 6, 7

[28] Black Forest Labs. Flux. `https://github.com/black-forest-labs/flux`, 2024. 4, 5

[29] Yandong Li, Xuhui Jia, Ruoxin Sang, Yukun Zhu, Bradley Green, Liqiang Wang, and Boqing Gong. Ranking neural checkpoints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2663–2673, 2021. 2, 6

[30] Derek Lim, Haggai Maron, Marc T Law, Jonathan Lorraine, and James Lucas. Graph metanetworks for processing diverse neural architectures. *arXiv preprint arXiv:2312.04501*, 2023. 6, 7

[31] Derek Lim, Moe Putterman, Robin Walters, Haggai Maron, and Stefanie Jegelka. The empirical impact of neural parameter symmetries, or lack thereof. *arXiv preprint arXiv:2405.20231*, 2024. 7

[32] Daohan Lu, Sheng-Yu Wang, Nupur Kumari, Rohan Agarwal, Mia Tang, David Bau, and Jun-Yan Zhu. Content-based search for deep generative models. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–12, 2023. 2, 6, 16

[33] Michael Luo, Justin Wong, Brandon Trabucco, Yanping Huang, Joseph E Gonzalez, Ruslan Salakhutdinov, Ion Stoica, et al. Stylus: Automatic adapter selection for diffusion models. *Advances in Neural Information Processing Systems*, 37:32888–32915, 2024. 16

[34] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 220–229, 2019. 4, 6

[35] Aviv Navon, Aviv Shamsian, Idan Achituve, Ethan Fetaya, Gal Chechik, and Haggai Maron. Equivariant architectures for learning in deep weight spaces. In *International Conference on Machine Learning*, pages 25790–25816. PMLR, 2023. 6, 7

[36] Aviv Navon, Aviv Shamsian, Ethan Fetaya, Gal Chechik, Nadav Dym, and Haggai Maron. Equivariant deep weight space alignment. *arXiv preprint arXiv:2310.13397*, 2023. 6, 7

[37] Cuong Nguyen, Tal Hassner, Matthias Seeger, and Cedric Archambeau. Leep: A new measure to evaluate transferability of learned representations. In *International Conference on Machine Learning*, pages 7294–7305. PMLR, 2020. 2, 6

[38] OpenAI. Chatgpt. https://chatgpt.com, 2022. Accessed: 2025-05-22. 8, 9

[39] Koyena Pal, David Bau, and Renée J Miller. Model lakes. *arXiv preprint arXiv:2403.02327*, 2024. 2

[40] William Peebles, Ilija Radosavovic, Tim Brooks, Alexei A Efros, and Jitendra Malik. Learning to learn with generative models of neural network checkpoints. *arXiv preprint arXiv:2209.12892*, 2022. 6

[41] Theo Putterman, Derek Lim, Yoav Gelberg, Stefanie Jegelka, and Haggai Maron. Learning on loras: Gl-equivariant processing of low-rank weight spaces for large finetuned models. *arXiv preprint arXiv:2410.04207*, 2024. 6, 7, 16

[42] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 2

[43] Mohammad Salama, Jonathan Kahana, Eliahu Horwitz, and Yedid Hoshen. Dataset size recovery from lora weights. *arXiv preprint arXiv:2406.19395*, 2024. 2, 6

[44] Konstantin Schürholt, Dimche Kostadinov, and Damian Borth. Self-supervised representation learning on neural network weights for model characteristic prediction. *Advances in Neural Information Processing Systems*, 34:16481–16493, 2021. 2, 6

[45] Konstantin Schürholt, Boris Knyazev, Xavier Giró-i Nieto, and Damian Borth. Hyper-representations as generative models: Sampling unseen neural network weights. *Advances in Neural Information Processing Systems*, 35:27906–27920, 2022. 2, 6, 16

[46] Konstantin Schürholt, Diyar Taskiran, Boris Knyazev, Xavier Giró-i Nieto, and Damian Borth. Model zoos: A dataset of diverse populations of neural network models. *Advances in Neural Information Processing Systems*, 35:38134–38148, 2022. 16

[47] Konstantin Schürholt, Giorgos Bouritsas, Eliahu Horwitz, Derek Lim, Yoav Gelberg, Bo Zhao, Allan Zhou, Damian Borth, and Stefanie Jegelka. Neural network weights as a new data modality. In *ICLR 2025 Workshop Proposals*, 2024. 2, 6

[48] Konstantin Schürholt, Michael W. Mahoney, and Damian Borth. Towards scalable and versatile weight space learning. In *Forty-first International Conference on Machine Learning*, 2024. 6

[49] Konstantin Schürholt, Léo Meynent, Yefan Zhou, Haiquan Lu, Yaoqing Yang, and Damian Borth. A model zoo on phase transitions in neural networks. *arXiv preprint arXiv:2504.18072*, 2025. 16

[50] Viraj Shah, Nataniel Ruiz, Forrester Cole, Erika Lu, Svetlana Lazebnik, Yuanzhen Li, and Varun Jampani. Ziplora: Any subject in any style by effectively merging loras. *arXiv preprint arXiv:2311.13600*, 2023. 5

[51] Aviv Shamsian, Aviv Navon, David W Zhang, Yan Zhang, Ethan Fetaya, Gal Chechik, and Haggai Maron. Improved generalization of weight space networks via augmentations. *arXiv preprint arXiv:2402.04081*, 2024. 7

[52] George Stoica, Pratik Ramesh, Boglarka Ecsedi, Leshem Choshen, and Judy Hoffman. Model merging with svd to tie the knots. *arXiv preprint arXiv:2410.19735*, 2024. 5

[53] Anh T Tran, Cuong V Nguyen, and Tal Hassner. Transferability and hardness of supervised classification tasks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1395–1405, 2019. 2, 6

[54] Thomas Unterthiner, Daniel Keysers, Sylvain Gelly, Olivier Bousquet, and Ilya Tolstikhin. Predicting neural network accuracy from weights. *arXiv preprint arXiv:2002.11448*, 2020. 6

[55] David Van Strien. Smol-hub-tldr. https://huggingface.co/davanstrien/Smol-Hub-tldr, 2024. Accessed: 2025-05-22. 6

[56] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR, 2022. 5

[57] Jiashu Xu, Fei Wang, Mingyu Derek Ma, Pang Wei Koh, Chaowei Xiao, and Muhao Chen. Instructional fingerprinting of large language models. *arXiv preprint arXiv:2401.12255*, 2024. 6

[58] Prateek Yadav, Colin Raffel, Mohammed Muqeeth, Lucas Caccia, Haokun Liu, Tianlong Chen, Mohit Bansal, Leshem Choshen, and Alessandro Sordoni. A survey on model moerging: Recycling and routing among specialized experts for collaborative learning. *arXiv preprint arXiv:2408.07057*, 2024. 5

[59] Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36, 2024. 5, 6

[60] Zhiguang Yang and Hanzhou Wu. A fingerprint for large language models. *arXiv preprint arXiv:2407.01235*, 2024. 6

[61] Nicolas Yax, Pierre-Yves Oudeyer, and Stefano Palminteri. PhyloLM: Inferring the phylogeny of large language models and predicting their performances in benchmarks. In *The Thirteenth International Conference on Learning Representations*, 2025. 2, 6, 16

[62] Kaichao You, Yong Liu, Jianmin Wang, and Mingsheng Long. Logme: Practical assessment of pre-trained models for transfer learning. In *International Conference on Machine Learning*, pages 12133–12143. PMLR, 2021. 2, 6

[63] Runpeng Yu and Xinchao Wang. Neural lineage. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4797–4807, 2024. 2, 6, 16

[64] Runpeng Yu and Xinchao Wang. Neural phylogeny: Fine-tuning relationship detection among neural networks. In *The Thirteenth International Conference on Learning Representations*, 2025. 2, 6, 16

[65] Yi-Kai Zhang, Ting-Ji Huang, Yao-Xiang Ding, De-Chuan Zhan, and Han-Jia Ye. Model spider: Learning to rank pre-trained models efficiently. *Advances in Neural Information Processing Systems*, 36:13692–13719, 2023. 2, 6

[66] Allan Zhou, Chelsea Finn, and James Harrison. Universal neural functionals. *arXiv preprint arXiv:2402.05232*, 2024. 7

[67] Allan Zhou, Kaien Yang, Kaylee Burns, Adriano Cardace, Yiding Jiang, Samuel Sokota, J Zico Kolter, and Chelsea Finn. Permutation equivariant neural functionals. *Advances in neural information processing systems*, 36, 2024.

[68] Allan Zhou, Kaien Yang, Yiding Jiang, Kaylee Burns, Winnie Xu, Samuel Sokota, J Zico Kolter, and Chelsea Finn. Neural functional transformers. *Advances in neural information processing systems*, 36, 2024. 6, 7

A Additional meta-ML research insights and visualizations

Depth. Fig. 9 shows that NLP models have a wide range of DAG depths, while in CV, nearly all models have a direct edge to the root foundation model. This suggests that the CV community puts more focus on new foundation models, while the NLP community often embraces iterative refinement. As an example, in Fig. 5 the LLM connected component (CC) exhibits significant depth and complexity, representing almost a third of the models. In contrast, while Flux is also substantial, its structure is much simpler and more uniform.

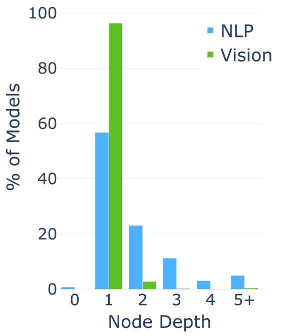
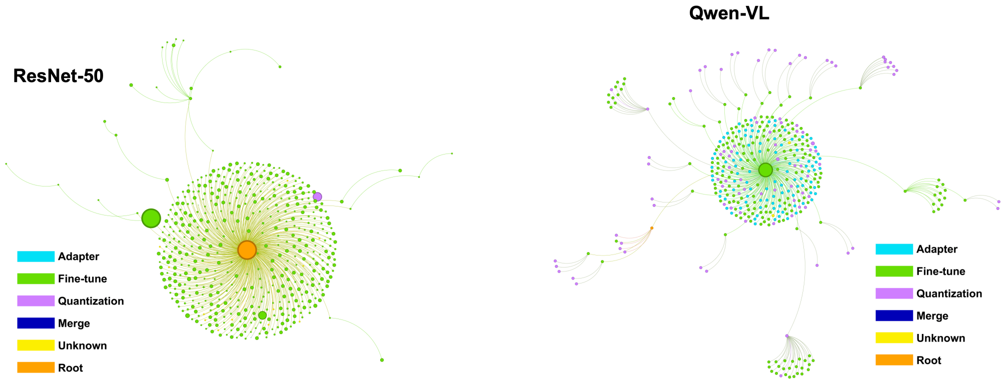


Figure 9: ***Vision vs. NLP Node depth:*** By analyzing over 314k models, we found that over 96% of CV models are situated one node away from the root, while only 55% of NLP models have this shallow depth. Over 5% of NLP models have depth of at least five nodes. This shows that NLP models are much deeper than CV models, suggesting the NLP community embraces iterative refinement over moving to the latest foundation models.

Better model impact measurement. There are several ways of measuring the impact of a model. For example, HF uses likes, trends (based on an undisclosed proprietary algorithm), and downloads. These metrics are somewhat myopic, as they measure the direct popularity of models but not the popularity of their descendant models. In fact, our graph analysis reveals that for 50% of non-leaf nodes, the total downloads of their descendants exceed their own individual downloads. This is partially due to the popularity of quantized child models and to incremental improvement of child models, e.g., finetuning or merging. Our analysis further showed that for non-leaf models, the sum of descendant nodes downloads exceeds those of the model itself in most cases (often by large margins). This suggests that simple model download counts underestimate the influence of the parent model.

We can therefore use the atlas to introduce a new model impact metric: `sub_tree_downloads`. We calculate this metric by summing the downloads of the model node and those of all its descendants. This number describes how many downloads this model causally affected. It has important applications to intellectual property rights, as the models and data used to train the target models affected all of its downstream downloads. It also quantifies the social impact of the biases of this model.

Restoring removed models. There are legal and commercial reasons for removing models from repos. Some models have been removed from the repository over time, affecting the integrity of the model DAGs. Out of 33,870 identified source models, 1,612 are missing due to deletion. A notable case is *runwayml/stable-diffusion-v1-5*, which has 3,038 broken references. To preserve the integrity of the dependency structure, the missing node was reconstructed and its connections manually restored. Using methods such as Horwitz et al. [20] also allows restoration of the weights.



(a) ***Individual connected components - 1/6***

(b) ***Individual connected components - 2/6***

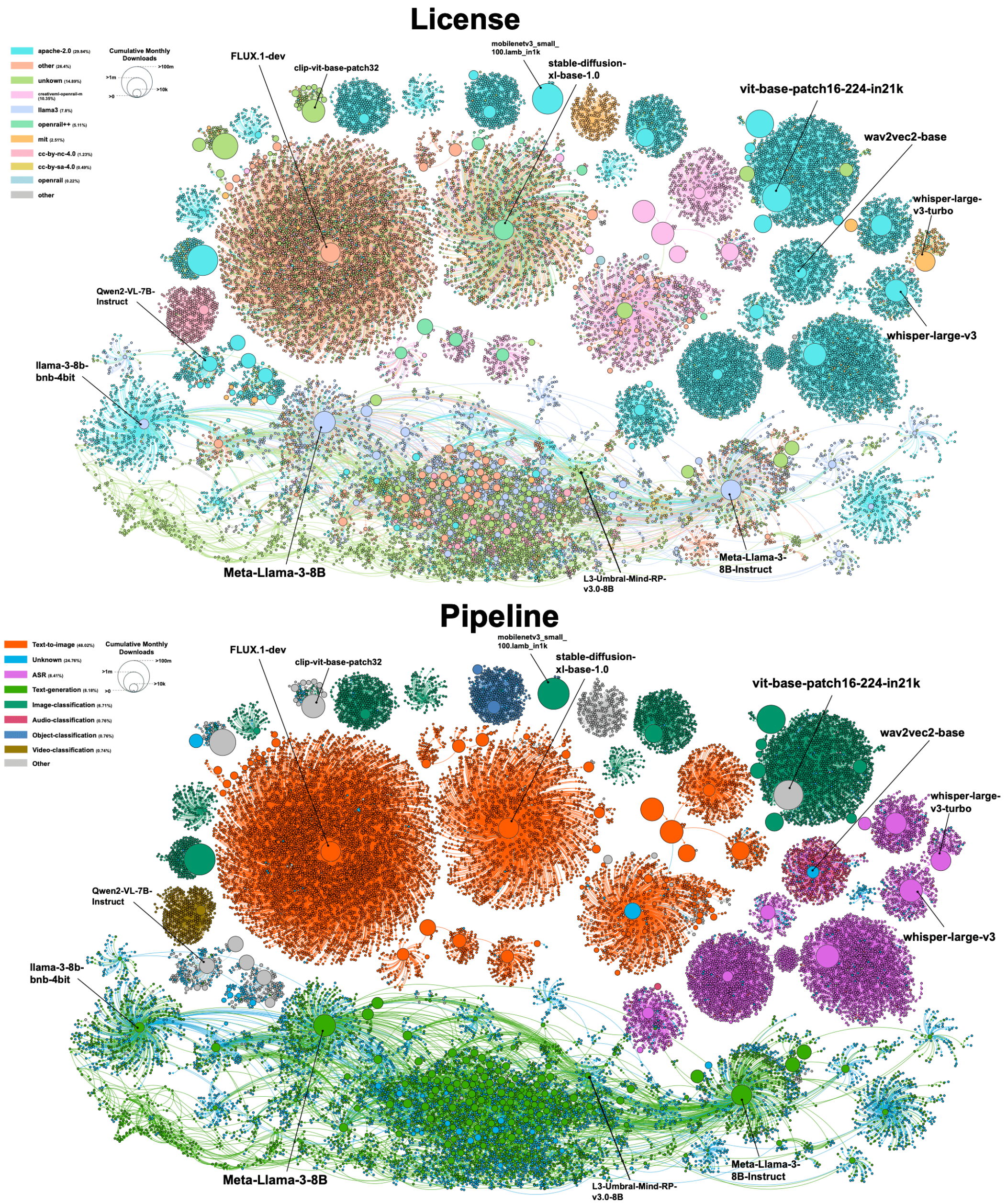
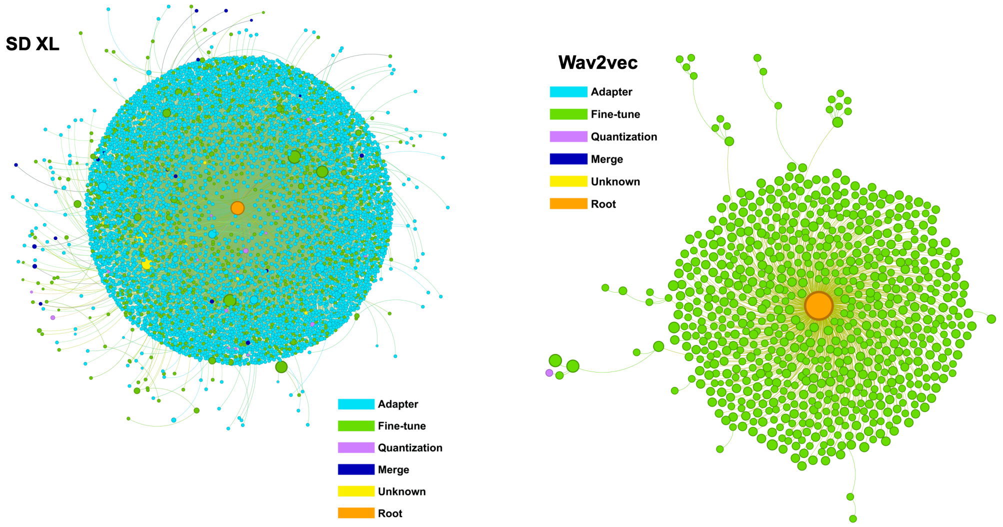


Figure 10: ***Visualizing other node features***

Additional visualizations In Fig. 10, we visualize the same atlas regions shown in Fig. 5, but colored according to different node features, specifically, license type and pipeline tag. In Figs. 8a, 8b, 11a, 11b, 16 and 17, we provide detailed visualizations of individual connected components. Figures have been compressed to reduce file size.

B Additional open challenges in Model Atlas charting

Meta-ML research. This paper presents a simple example of meta-ML research through structural and visual analysis of the Model Atlas. However, extracting deeper insights will require more sophisticated methods. One direction involves visual and interactive exploration, comparing regions of the atlas by architecture family, task domain, training patterns, or performance metrics. Another involves training models on the atlas structure itself, for example, using graph neural networks



(a) ***Individual connected components - 3/6***

(b) ***Individual connected components - 4/6***

(GNNs) to predict missing attributes. Beyond analysis, the Model Atlas can be extended by the broader community through the addition of new layers of node and edge features. Researchers in interpretability, for instance, might annotate models with salient neurons or circuits, while robustness researchers could contribute metrics such as adversarial susceptibility or failure modes. These enriched features would support downstream meta-ML tasks, enabling comparisons of interpretability across architectures or correlations between robustness and lineage.

Model charting datasets. As our goal is to scale model charting to real-world scenarios, it is essential to use datasets drawn directly from real-world repositories. However, most existing dataset papers [13, 46, 49] and methods that introduce datasets [19–21, 41, 45] rely on training their own model collections. Curating datasets and benchmarks from models hosted on Hugging Face offers a convenient and scalable alternative. In fact, using the already-charted Model Atlas, one can assemble large datasets of in-the-wild models directly from Hugging Face. We include two curated datasets with this paper for graph-level model attribute prediction and charting tasks (see App. E for details).

C Charting the atlas structure

While the preceding section highlighted the importance of the Model Atlas, in practice, over 60% of it is unknown. This is primarily because model uploaders frequently do not provide parent model information, as illustrated in Fig. 12. Atlas charting aims to recover the missing edges. Recent methods [21, 61, 63, 64] tackled some aspects of this task, but were not applied to fully in-the-wild settings. As such, some of their key assumptions (e.g., tree-like structures) do not hold in real-world repos. Additionally, some methods [61] require running each model on multiple inputs, which is computationally infeasible for millions of models. We introduce an approach for in-the-wild charting.

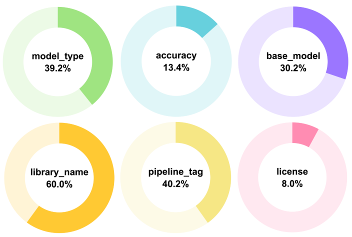


Figure 12: ***Documentation level in Hugging Face***

Charting typically begins by measuring pairwise model distances, which requires a representation for each model as well as a distance function. Common model representations include: weights [21, 61, 63, 64], activations [61], gradients [63], outputs [26, 32], metadata [33] or a combination of the above [63]. We represent each model i by its weights w_i. We measure the distance between two models using the Euclidean distance of their representations: $D_{ij} = \|w_i - w_j\|^2$.

The core challenge is charting the atlas structure from this distance matrix. While previous approaches used simple, generic solutions, such as greedy nearest neighbor or minimal directed spanning tree, this does not generalize to in-the-wild Model Atlases. The reasons for this include: i) models with the nearest weight distance are not always connected by an edge, ii) the Model Atlas structure is generally not a tree. Instead, we propose a set of charting rules motivated by special patterns in model repos: duplication, quantization near-duplications, checkpoint trajectories, hyperparameter search, and merging. Identifying and handling these patterns can dramatically improve charting accuracy.

Naive strategy. Our preliminary strategy first splits models into non-overlapping subsets using a standard clustering algorithm on the weight distance matrix. Previous works [15, 19, 21] showed that each subset corresponds to the nodes within a single connected component of the atlas DAG, i.e., we can recover the structure of each connected component separately. The naive strategy first assigns source nodes (this will be elaborated on below), then, at each step, it looks for the unassigned node with the shortest distance to the assigned nodes, and connects the two with a directed edge. The algorithm stops when there are no unassigned nodes.

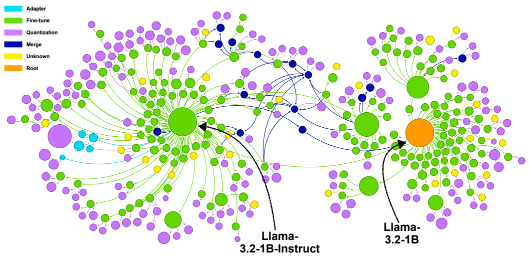


Figure 13: ***Quantizations are leaves:*** Our analysis of over 400,000 documented model relationships reveals that 99.41% of quantized models are leaf nodes. This figure shows this for a subset of the Llama-based models. Indeed, quantized models (magenta) are nearly always leaf nodes, corroborating the statistical finding.

Duplicates and near-duplicates. Real-world model repositories contain many duplicates and near-duplicates. Exact duplicates occur when users download a popular base model and re-upload it without modification. In this, both the original and duplicates have *identical* distances to all models. This makes the greedy distance-based algorithm arbitrarily decide between the true parent and its duplicates, which obviously reduces accuracy. To mitigate this, we identify models with zero distance as exact duplicates. We retain a single representative instance, designating duplicates as leaves.

Duplication Assumption

Exact duplicates and quantized versions of models are treated as leaf nodes in the atlas.

Model quantization compresses models to reduce memory and storage as well as accelerate their inference. It transforms a model into a near-duplicate, as the values of each weight are typically very similar yet not identical to the original one. This creates a similar ambiguity to exact duplicate models, as the distance of the original and quantized model to any other model are almost the same. Moreover, many models have multiple quantizations, which greatly increases this ambiguity.

To overcome this ambiguity, we identify a pattern of quantization models from real-world atlases. Our analysis reveals that 99.41% of quantized models on HF are leaf nodes, i.e., they have no child models. See Fig. 13 for an illustration on the Llama 3.2-1B CC. Intuitively, unquantized models have better performance than their quantized versions, and practitioners typically use the highest-performing models for further fine-tuning or merging. Fortunately, detecting quantization is often straightforward. Quantized models have reduced precision data types (e.g., int8, float16 instead of float32) and fewer unique weight values. Therefore, we simply identify quantized models and designate them as leaves.

Temporal Dynamics Assumption

Models with earlier timestamps are assumed to be topologically higher in the atlas. For snake patterns, the parent is the closest preceding model; for fan patterns, it is the earliest model among the top K weight-based neighbors.

Temporal dynamics for inferring fine structure. While the weight distance between models is inversely correlated to the likelihood of having an edge between them, it does not reveal the direction of the edge. Previous approaches predicted edge direction using supervised learning methods or by unlearned metrics, such as gradients or weight kurtosis. However, we find that these methods often do not generalize effectively to real-world model repositories. A key advantage of real-world repositories, unused by prior works, is the availability of model creation or upload timestamps. We find that the vast majority of parent models have earlier timestamps than their children, specifically, in the HF model repository, this occurs for 99.73% of all observed parent-child pairs. We visualize this phenomenon for the

Llama-3.2-1B CC in Fig. 14. This is a powerful temporal constraint on atlas structure, and in particular identifies the source nodes as the earliest ones.

Another limitation of the weight distance is that it may have limited sensitivity in fine-grained atlas structures. Two common scenarios that challenge charting methods based on weight distance are hyperparameter sweeps and checkpoint trajectories. In hyperparameter sweeps, multiple models are trained from a single parent, each with different hyperparameters. Minor hyperparameter changes can result in sibling models having nearer weight distances to each other than to their common parent, defying the nearest neighbor algorithm. Conversely, checkpoint trajectories represent a sequence of models saved during a single training run. In this case, edges should connect consecutive checkpoints in a chain-like manner, rather than connecting each checkpoint directly to the initial model (as in the hyperparameter sweep scenario). We term the former pattern a "fan" pattern and the latter a "snake" pattern, illustrated in Fig. 15, and find that the weight distance often confuses these two patterns.

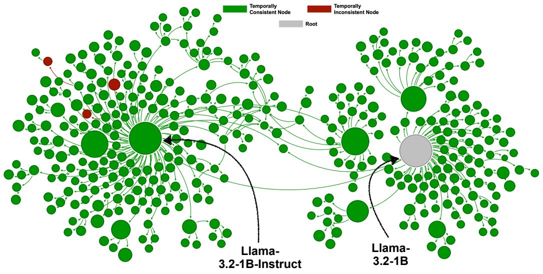


Figure 14: ***Temporal dynamics indicate edge directionality:*** We analyzed over 400,000 documented model relationships and observed that in 99.73% of cases, earlier upload times correlate with topologically higher positions in the DAG. Here, we visualize this trend on a subset of the Llama model family. Green nodes indicate models where earlier upload times align with topological order, while red nodes represent exceptions to this trend. The source (in gray) vacuously satisfied this assumption. It is clear that nearly all nodes satisfy our assumption.

Our key observation is that the temporal model weight evolution can discriminate between these patterns. In snake patterns, temporal proximity strongly correlates with weight proximity, due to the sequential evolution. However, this is not the case in fan patterns, where the closest siblings are not necessarily the closest in time. This leads to a simple yet effective decision rule. If the weight distance of a model to its K nearest neighbors is highly correlated to their absolute temporal distance, we classify the pattern as a snake; otherwise, a fan. Then, we connect the node to one of its nearest neighbors depending on the detected structure. In snakes, we connect the node to its first nearest neighbor based on weight distance, while in fans, we connect the node to its oldest nearest neighbor, aiming for the fan's origin.

Merged models. Prior work assumes that models lie in a directed tree, however, as merged models have multiple parents, they have more than one incoming edge. This cannot be described by trees, but can be described by non-tree DAGs. We found that many merged models are created using a few popular libraries, which typically document the parent nodes in the created model card, hence, the multiple incoming edges are often known. The challenge left is to chart a DAG instead of a tree. We propose a greedy charting algorithm. We first sort all nodes by upload times. Then, we process each node in temporal order, predict its parents, and add those edges to the graph. In the case where the parents are known, we use these edges instead of predicting them. The algorithm stops where there are no more nodes to process. The runtime complexity is $O(n^2)$, where n is the number of models. It is much faster than [21] (which is typically $O(n^3)$). In practice, we recover graphs with thousands of nodes in seconds.

Structural Assumption

The atlas structure is a non-tree directed acyclic graph (DAG).

The most expensive part of the algorithm in terms of compute and storage is the distance matrix calculation, as it scales with the number of network weights. Therefore, we propose to represent each model by subsampling the full weight representation to a much smaller number of neurons. Specifically, we find that keeping just 100 is typically sufficient for significantly speeding up the runtime while keeping accuracy nearly unchanged.

Full algorithm. We present the full algorithm in python-like pseudo code in Alg. 1. The input is a set of models M, each with weights w, creation time t, known parents P (or $P = None$), and whether it is quantized q. The algorithm has 3 hyperparameters: K - the number of neighbors, K_{th}, ρ_{th} - thresholds for snake-fan classification. The output of the algorithm is the predicted parents for each node $m.P$. Note that we can run the algorithm online with a small modification, by computing the distance function and kNN as new models arrive. See App. D for implementation.

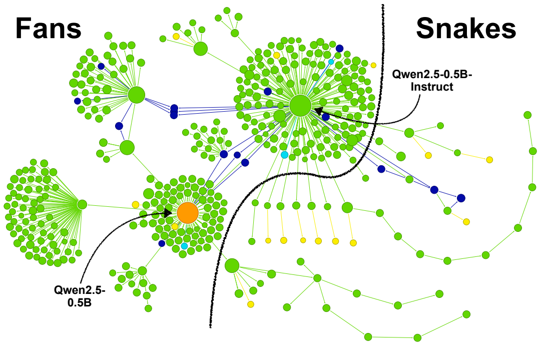


Figure 15: ***Snake vs. Fan patterns:*** Snake patterns often arise from sequential training checkpoints, while fan patterns typically result from hyperparameter sweeps. In both structures, the model weight variance is low. However, in snake patterns, the weight distance has a high correlation with model upload time, whereas in fan patterns, the correlation is lower. Note colors are the same as Fig. 13

C.1 Charting results

Datasets. We created an atlas charting dataset based on the ground truth model relation found on the "hub-stats"[2] dataset. It consists of 3 atlas connected components: `Qwen2.5-0.5B` (Qwen), `Llama-3.2-1B` (Llama), and `Stable-Diffusion-2` (SD). We also created another attribute prediction dataset. Its task is to predict 6 attributes: `accuracy`, `pipeline_tag`, `library_name`, `model_type`, `license`, `relation_type`. The ground truth was obtained by a combination of model metadata and running an LLM on the model cards. We detail the dataset creation process in App. E and will publicly release the datasets.

Baselines. We compared a weight-agnostic classical method and a newer method that depends on weights. *MoTHer [21]* - a recent weight-dependent method. Importantly, it predicts edge direction using weight kurtosis and relies on the structure being a tree. We could not scale other weight-based methods to operate on our in-the-wild dataset. We also included the following structure-only baselines: *random* edge assignment, *majority vote* (assigning all nodes to the parent with the highest out-degree), and *Price's model [6]*. The latter is a preferential attachment algorithm [1], popular for citation networks, that assigns edge probabilities based on node out-degree. It is essentially a probabilistic version of majority vote that allows for new "hub" formation. Note that all baselines (except random) assume the structure is a tree, not a DAG. To accommodate the majority and Price's models, we assumed a graph stem with known edges, containing 10% of the nodes in the connected component. We evaluated all methods by their accuracy on the remaining 90% of models.

Algorithm 1: Atlas structure recovery (psuedo-code)

```
# Input: M = {(w, q, t, P)}, K, K_th, rho_th
# w = weights, t = creation time, P = known parents
M = M.sort(key=m.t) # Sort M by timestamp t
D = compute_distance_matrix(M)
D.diag = inf

for i, m in enumerate(M):
    if is_quantized(m.w): # Set quantized model as leaf node
        D[i, :] = inf
    if m.P: # Use existing parents if available
        continue
    for j in range(i+1, len(M)): # Deduplication
        if D[i, j] == 0:
            D[j, :] = inf
            M[j].P = M[i].P

    # Find k-NN
    k_ind = argsort(D[:, i])[1:K]
    temporal_k_ind = argsort(D[i+1:, i])[1:K]
    k_dist = D[k_ind, i]
    k_times = [M[k].t for k in k_ind]
    temporal_k_times = [M[k].t for k in temporal_k_ind]


    # Check spread of distances
    if k_dist[-1] - k_dist[0] > K_th:
        m.P = temporal_k_ind[0]
    else:
        corr = correlation(k_dist, abs(k_times - m.t))
        if corr > rho_th: # Snake
            m.P = temporal_k_ind[0]
        else: # Fan
            m.P = temporal_k_ind[argmin(temporal_k_times)]
```

Baseline comparison. We use our method to chart three atlas connected components. As we can see in Tab. 2, the baselines perform better than random but still poorly. The majority approach models the graph as depth-1 centered around the root. This obtains low but non-trivial results, as this model is too simplistic. Price's model is more powerful and can create a more realistic looking structure. However, it fails as it is data agnostic and connects the wrong nodes. MoTHer performs

[2] huggingface.co/datasets/cfahlgren1/hub-stats

Table 2: ***Atlas recovery results:*** Our method outperforms the baselines by a significant margin, even for in-the-wild models.

Method	Qwen	Llama	SD
Random - root	1.77	1.84	0.22
Random	0.98	0.67	0.75
Majority	15.03	25.00	36.75
Price's [6]	2.28	5.08	8.50
MoTHer [21]	32.81	19.32	50.51
Ours	**78.87**	**80.44**	**85.10**

Table 3: ***Subtractive ablation:*** We remove each of our assumptions individually, indeed, we see that each one contributes to our final high recovery accuracy.

	Method	Qwen	Llama	SD
Ours	– Greedy Alg.	77.34	78.98	Failed
	– Quantization	70.57	36.59	84.85
	– Deduplication	67.85	75.28	**88.76**
	– Temporal Consistency	75.47	80.13	80.86
	– Fans vs. Snakes	75.09	76.03	71.72
	– Merges	76.95	76.61	84.85
	Ours	**78.87**	**80.44**	85.10

better on some DAGs, but its directional prediction is weaker and it lacks our other priors, resulting in lower performance.

D Implementation details

To visualize the atlas, we use the open-source software Gephi [3]. We set $K = 5$ nearest neighbors in all DAGs and determine the snake correlation threshold (ρ_{th}) dynamically for each DAG as the 60th percentile of all node correlations. The distance spreading correlation threshold (K_{th}) is fixed at 0.05.

Additionally, in all our experiments, we use 100 random neurons as model features. For Stable Diffusion, we select these neurons exclusively from the attention layers, as these layers typically undergo LoRA fine-tuning. In all DAGs, we allocate 10% of the actual DAG for training and reserve the remaining 90% as a test set. These hyperparameters are applied consistently across all three tested DAG structure recovery settings.

E Dataset details

We construct two datasets from the Hugging Face "hub-stats" dataset[3]: (i) Metadata Imputation and (ii) DAG Recovery. We will make our dataset publicly available on Hugging Face. To process the data, we first used the `base_model` attribute to group models into connected components. Since many models lacked this attribute, we manually inspected the largest 800 connected components and filled in the missing values. This step helped merge large components with incomplete information and reduced data noise.

The metadata imputation dataset includes all 1.3 million models. The tasks involve imputing the following attributes: `pipeline_tag`, `library_name`, `model_type`, `license`, and `relation_type`, as well as estimating model evaluation metrics. To obtain the evaluation metrics, we downloaded model cards and extracted the metrics using *deepseek-ai/DeepSeek-R1-Distill-Qwen-7B* in a zero-shot setting on models from *mistralai/Mistral-7B-v0.1* CC.

To construct the charting dataset, we created a graph of all models on Hugging Face. Using the `base_model` attribute, we established ground truth edges between models that reference each other. We removed all nodes with no parent or child (i.e., CCs with a single model), reducing the initial 1.3 million models to approximately 400,000 models across different connected components. We defined sources as nodes with no incoming edges and considered weakly connected components, ensuring that each CC has a single source node.

To facilitate model downloads, we pruned the CCs by randomly sampling K leaf nodes if a node had more than K leaf nodes, filtering out the rest. This allowed us to download hundreds rather than tens of thousands of models. We set $K = 3$. If a download failed, we removed the model from the ground truth atlas.

During the download process, quantized model weights required special handling to convert them back into workable weight files. The conversion was based on the tensor type, which indicates whether a model has undergone quantization. While reviewing the Hugging Face model cards, we

[3] https://huggingface.co/datasets/cfahlgren1/hub-stats

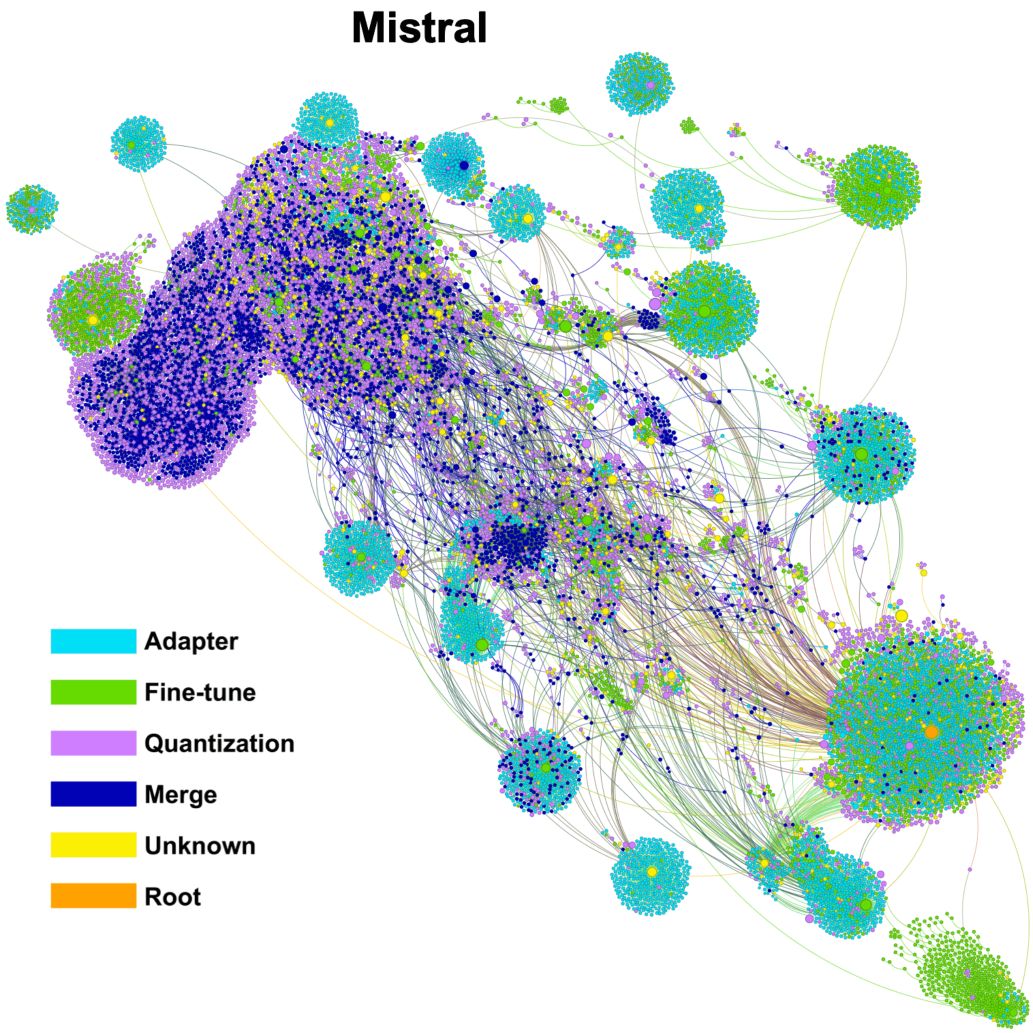


Figure 16: ***Individual connected components - 5/6***

discovered multiple tagging inconsistencies; some models were incorrectly labeled as quantized, while others were missing the label. We propose using the tensor type as a reliable indicator of quantization status.

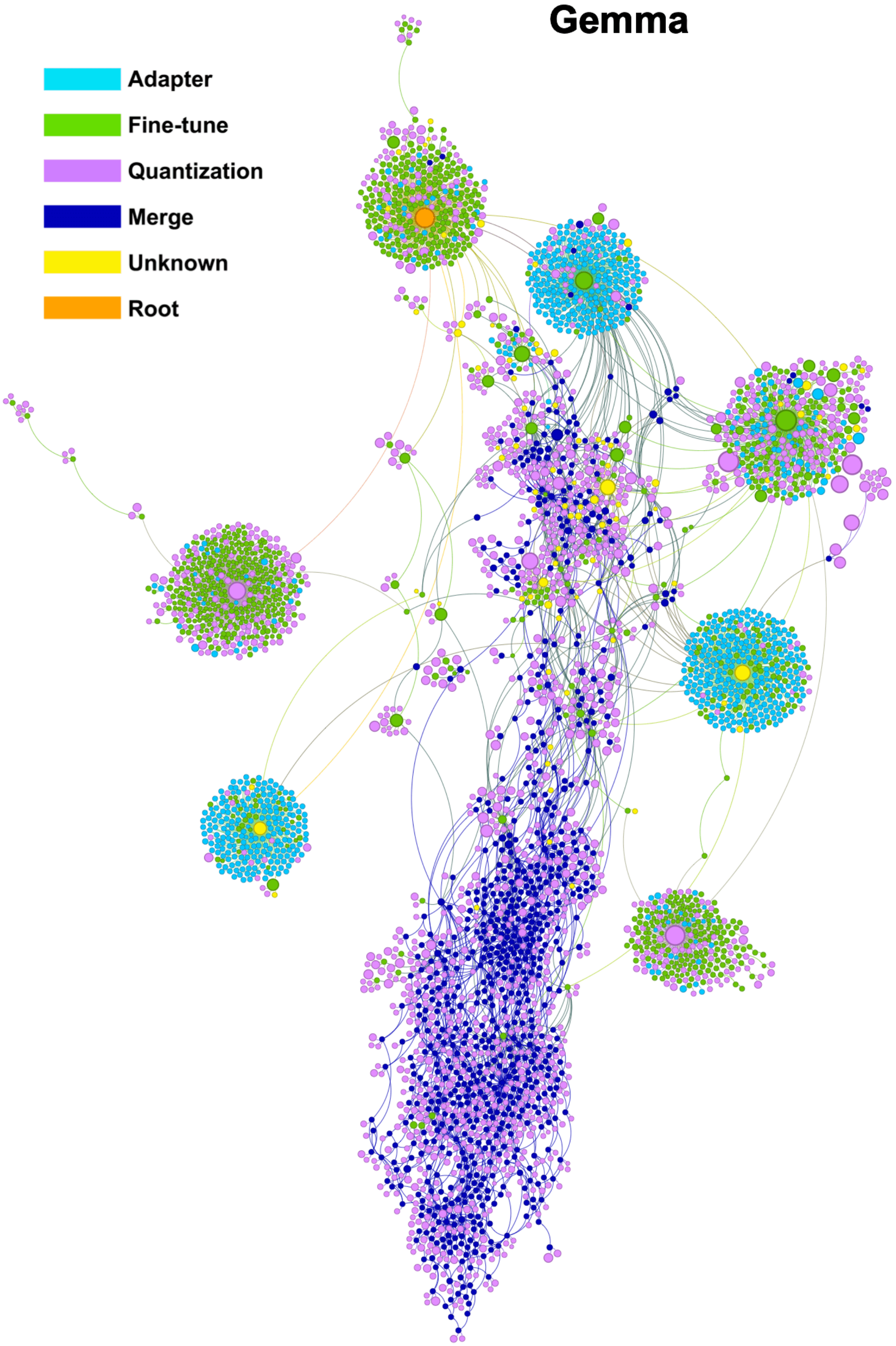


Figure 17: ***Individual connected components - 6/6***