WARP: Weight-Space Analysis for Recovering Training Data Portfolios

Tzu-Heng Huang*† Aditya Goyal*† John Cooper† Frederic Sala†

†University of Wisconsin-Madison

July 3, 2026

Abstract

Foundation models are routinely released to the public, yet the data recipes used to train them—such as domain mixture weights that determine how different sources are sampled—are rarely disclosed. This creates an access asymmetry: researchers study the resulting models but lack visibility into the training distribution that produces them. Prior works for inferring training data, such as membership inference, detect at the level of individual samples and thus cannot characterize the global composition of the training corpus. We introduce WARP, a framework that recovers a fine-tuned model's training mixtures directly from its released weights. WARP interpolates between the base and fine-tuned models using model merging, generating *pseudo-checkpoints* that approximate the missing training trajectory and expose a geometric footprint of the training data in the weight space. From these simulated footprints, WARP extracts geometric features and maps them to domain proportions using either a parameter-free softmax readout or an MLP projector trained on synthetic mixtures. In controlled experiments with BERT and GPT-2, WARP recovers domain mixtures with an average MAE as low as 0.046 and 0.104 respectively, outperforming membership inference and a variant with access to the true training trajectory.

1 Introduction

Foundation models are trained on massive, heterogeneous corpora spanning web text, source code, and scientific literature. One primary determinant of a model's ultimate capability is its ***domain mixture***: the proportions in which samples are drawn from these diverse sources [1]. Recent studies have shown that tuning this data mixture is not merely a matter of ensuring knowledge variety, but of training efficiency—a well-designed mixture can yield models that substantially outperform naive sampling while requiring fewer optimization steps [2; 3; 4; 5].

Yet the mixing recipes behind most frontier models remain proprietary and opaque. This secrecy reflects a broader ***access asymmetry*** in modern AI: general-purpose models are increasingly released to the public, while the carefully curated datasets used to produce them are not. Such opacity creates structural barriers for the research community. Practitioners who continually fine-tune a released model without knowing its original distribution risk unintended capability drift, and more broadly, recovering a model's data portfolio is useful for auditing data contamination [6; 7; 8] and for interpreting divergent model behaviors [9].

Breaking through this barrier, however, is far from straightforward. Existing approaches to domain mixture optimization operate primarily in the ***forward*** direction—***from data to model***. They either establish an optimized portfolio for a given dataset via gradient information [3; 5], or train multiple proxy models that capture the relationship from mixtures to downstream performance [10; 11; 12; 13]. These natural designs make it hard to characterize already-released weights. The alternative—working

*Equal Contribution. Corresponding Authors: <thuang273, agoyal33>@wisc.edu

[1]This work appears in the ICML 2026 Workshop on Weight-Space Symmetries (WSS): from Foundations to Practical Applications.

[2]Our source code is available here.

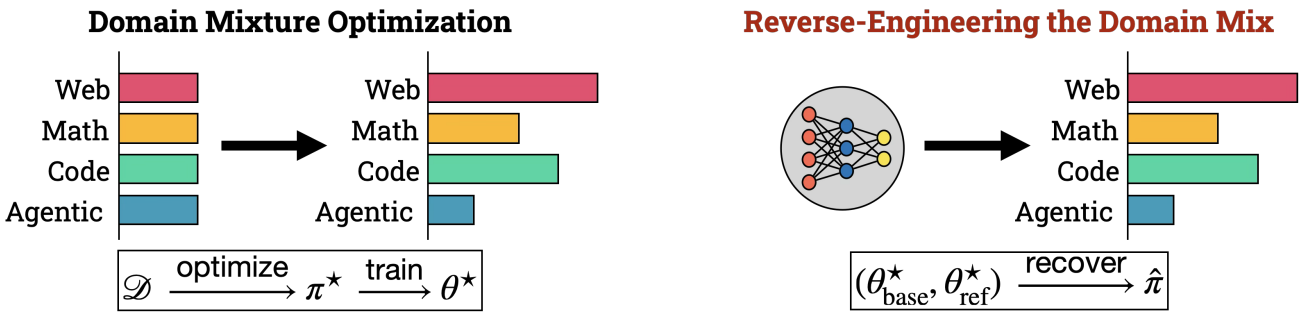

Figure 1: **Two directions of the domain mixture problem. Left:** the well-studied *forward* problem—given a corpus, search for an optimized mixture $\pi^\star$ over domains (e.g., web, math, code, agentic data) and train a model $\theta^\star$ under it, whose final weights are then publicly released. **Right:** the *inverse* problem we study—given only the released endpoints $(\theta^\star_{\text{base}}, \theta^\star_{\text{ref}})$ and a small probing dataset, with the corpus and true training trajectory withheld, recover an estimate $\hat{\pi}$ of the domain proportions that produced the fine-tuned model.

backward from a model to its training data—has traditionally been framed as Membership Inference (MI) [14; 15; 16; 17], which asks whether a particular sample appeared in the training pool, typically by comparing its attribution (e.g., loss or output logits) against a reference distribution. However, MI is fundamentally a sample-level detection task and offers no holistic view of underlying domain proportions.

One promising direction has recently appeared: using *weight-space geometry*, the one artifact always available, to recover insights about training data [18; 19; 20]. For example, the Mimic Score [20] combines a directional vector in weight-space with training gradients to estimate sample utility for data curation. These methods share a simple premise: ***a model's parameters encode traces of its training history, and therefore of the data that produced it***. Yet they typically require access to *intermediate* training checkpoints to reconstruct the learning trajectory—a luxury rarely available in practice. For the vast majority of released models, only the final weights (e.g., INSTRUCT, CHAT versions) and the base model (e.g., pretrained-only) can be obtained: the origin and the destination, with no record of the path between them.

In this work, we break this asymmetry with **WARP** (**W**eight-space **A**nalysis for **R**ecovering Training Data **P**ortfolios), a framework that recovers a fine-tuned model's domain mixtures directly from its released weights. We argue that ***though the true training trajectory is lost, alternative interpolants between the two endpoints still expose usable structure in the surrounding weight-space geometry.*** Specifically, we use model merging techniques to interpolate between the base and fine-tuned models, generating ***pseudo-checkpoints*** that stand in for the unobserved parameter evolution. From these simulated trajectories, their extracted geometric features allow us to capture a distributional footprint of the training data. Given a small probing dataset, **(i)** a parameter-free softmax readout can recover a coarse mixture estimate; or **(ii)** a stronger variant uses synthetic mixtures to build an MLP for learning the projection from geometric features to predicted mixture.

We empirically validate WARP in a controlled setting by training BERT [21] and GPT-2 [22] under known mixtures and recovering the proportions from each fine-tuned checkpoint. Across forty experimental trials, WARP achieves a mean absolute error (MAE) as low as 0.046 on BERT and 0.104 on GPT-2 averaged across four text datasets, outperforming sample-level MI baselines and a variant that allows using the true training trajectory. Moreover, the WARP estimator remains accurate on early-stop, converged, and overtrained checkpoints—mirroring today's post-training regime, where released models are routinely pushed beyond compute-optimal budgets, demonstrating its robustness to different training recipes.

We summarize our contributions as follows:

- **Closing the Access Asymmetry.** We formalize the recovery of a fine-tuned model's domain mixture from its weight-space footprints, relying only on artifacts available for most of the released models.

- **Simulated Training Trajectories.** We propose WARP, which sidesteps the need for intermediate checkpoints by reconstructing the missing path through model merging, then projects the resulting weight-space geometry to domain mixture estimates.
- **Strong Empirical Performance.** Across controlled fine-tuning runs, WARP outperforms sample-level MI baselines and even a variant that has access to the true training trajectory.
- **Robustness Across Training Recipes.** WARP remains accurate when recovering converged and overtrained checkpoints, demonstrating robustness to diverse training recipes.

2 Related Work

Our work sits at the intersection of three research threads: **(i)** domain mixture optimization, **(ii)** training data inference, and **(iii)** model merging techniques.

Domain Mixture Optimization. Establishing an ideal data mixture plays a critical role in training foundation models [1]. Rather than relying on grid search to discover the best configuration, a recent line of work has developed principled methods for optimizing the data mixture [2; 4; 23]. These methods broadly fall into two groups: (i) static mixing and (ii) dynamic reweighting. Static methods fix the mixture before training, typically using proxy models to predict how mixture weights affect downstream performance—for example, by fitting a regression model that maps weights to performance [10; 11; 12; 13], or by leveraging gradient information from a held-out set to optimize the weights [5]. Dynamic methods instead adapt the mixture weights throughout training, for example by constructing a gradient-alignment matrix (R&B) [3] or by casting data mixing as a multi-armed bandit problem [24]. ***All of these methods operate in the forward direction: given the data, they find the best mixture.*** In contrast, we address the reverse problem: given a model, we aim to recover the mixture that produced it.

Training Data Inference. Inferring properties of a model's training data, especially for large language models, is of growing interest, motivated both by privacy concerns—over the unauthorized use of copyrighted and personal information [6; 25]—and by the practical utility of extracting high-quality data insights for next-round data curation [20]. A common formulation is *membership inference* [14; 15; 16; 17], which detects whether a particular sample was used in training by comparing the model's behavior against a reference distribution, for instance, by examining the difference in output logits between an untuned and a fine-tuned model. However, membership inference operates at the level of individual samples and therefore lacks a holistic view of how domains interact. Even if the detected samples are aggregated to recover domain mixtures, the resulting estimate is confined to the probing dataset and remains suboptimal. In this work, we propose a framework that ***recovers a model's training trajectory in weight-space and leverages domain-level geometric features to recover the training mixture.***

Model Merging Techniques. Model merging combines two or more models into a single fused model that inherits the capabilities of each [26; 27; 28]. It operates in weight space, where model weights can be permuted [29; 30], interpolated [31; 32], stitched [33], or averaged [34] to yield versatile skills. Most work focuses on resolving conflicts between models to maximize the fused model's performance. WARP departs from this view: rather than treating merging as a way to combine capabilities, ***we use merging as a mechanism for simulating an unobserved training trajectory—constructing pseudo-checkpoints that stand in for the missing path between a base model and its fine-tuned descendant***.

3 Framework

We propose **WARP** to recover the domain mixture of a fine-tuned model under two practical constraints: (i) the final weights and the corresponding base model are available, and (ii) the practitioner has access to a data source S from which the fine-tuning data is drawn (e.g., the web datasets), and can freely sample a small set of labeled examples from it. We argue that ***the geometry***

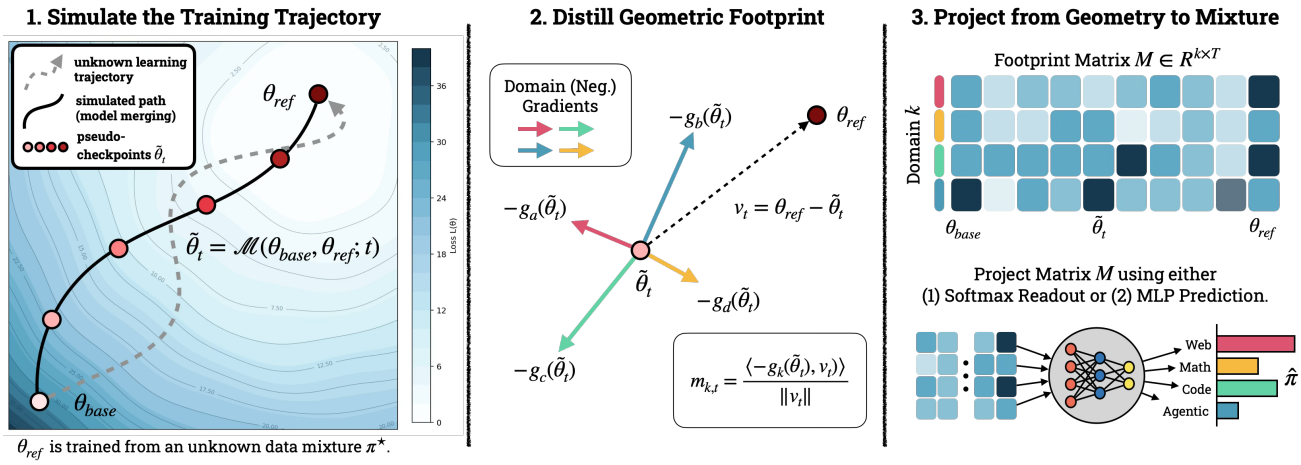

Figure 2: **Overview of WARP. (1) Simulate the Training Trajectory:** since the true fine-tuning path is unobserved, we first approximate it with a sequence of pseudo-checkpoints obtained via model merging. **(2) Distill Geometric Footprint:** at each pseudo-checkpoint, we compute the Mimic Score by projecting the gradient of probing samples from each domain onto the direction pointing to reference model. This measures how strongly each domain aligns with the path actually taken during fine-tuning. **(3) Project from Geometry to Mixture:** the resulting footprint matrix is mapped to a predicted domain mixture, either via an unsupervised softmax readout or a supervised MLP projector trained on synthetic mixture-footprint pairs.

traced between a base model and its fine-tuned descendant encodes a distributional footprint of the training data, and this footprint can be read out by measuring how samples from each domain align with the geometry.

We begin with notation, then describe how we bypass the unobserved learning trajectory through simulation (§3.1), how the resulting weight-space geometry is converted into a domain-level signal (§3.2), and finally how this signal is mapped to a predicted domain mixture (§3.3). We demonstrate our framework workflow in Figure 2. Moreover, we provide a unified algorithm in Appendix A.

Notation. Let θ_{base} denote the parameters of a publicly available base model (pretrained-only), and θ_{ref} those of a released fine-tuned model derived from it. Its fine-tuning data D_{ref} is sampled from a source S that spans K domains, with unknown mixture proportions $\pi^\star = (\pi_1^\star, \ldots, \pi_K^\star) \in \Delta^{K-1}$, where Δ^{K-1} is the probability simplex. Our goal is to estimate $\pi^\star$ from θ_{base} and θ_{ref} alone, without access to D_{ref} or any intermediate training checkpoint. To reach this estimation, we assume the ability to sample examples from S in our own way, allowing us to construct a small probing dataset $D_{\text{probe}} = \bigcup_{k=1}^{K} D_k$, where each $D_k = \{(x_i, y_i)\}_{i=1}^{n_k}$ contains samples with known domain label k and $|D_{\text{probe}}| \ll |D_{\text{ref}}|$. Crucially, D_{probe} need not follow the same mixture as D_{ref}—it need only cover the K domains of interest. For any parameter vector θ, we write each sample's gradient as $g_i(\theta) := \nabla_\theta \ell(x_i, y_i)$.

3.1 Simulating the Training Trajectory

To exploit the full geometric information along a training path, one would ideally have access to its intermediate checkpoints. In practice, however, the true trajectory from θ_{base} to θ_{ref} is rarely released, leaving us with only its two endpoints. To recover a usable approximation, we draw on recent advances in model merging [35; 27] and interpolate between these endpoints to construct a sequence of ***pseudo-checkpoints***: $\tilde{\theta}_t := \mathcal{M}(\theta_{\text{base}}, \theta_{\text{ref}}; t)$, where $\mathcal{M}$ is a merging operator. For linear interpolation,

TIES [31], and other merging methods, it is common to have $\{\alpha_t\}_{t=1}^{T} \subset (0,1)$ with

$$\tilde{\theta}_t = (1-\alpha_t)\theta_{\text{base}} + \alpha_t \theta_{\text{ref}}, \qquad t = 1,...,T,$$

as a schedule of mixing coefficients that places each pseudo-checkpoint between the two endpoints. Intuitively, α_t plays the role of a normalized training progress: values near 0 yield an *early-stage* state close to θ_{base}, where the model has only begun to absorb the fine-tuning data, while values near 1 yield a *late-stage* state near θ_{ref}, where most of the adaptation has already taken place. Each $\tilde{\theta}_t$ thus serves as a proxy for the parameter state at an intermediate point along the simulated trajectory.

3.2 Distilling Geometric Footprints

With a simulated trajectory in hand, we now extract a domain-level signal from pseudo-checkpoints' weight-space geometry.

Sample Alignment in Weight-space. The Mimic Score [20] measures how well a sample's negative gradient aligns with a direction in the weight space pointing toward a more desirable parameter set: *samples whose updates can push the model along this direction receive higher scores and are treated as higher-value training examples.* We adapt this quality metric to our setting by treating each pseudo-checkpoint $\tilde{\theta}_t$ as the current state and θ_{ref} as the desired target, which together define a time-varying direction vector $v_t := \theta_{\text{ref}} - \tilde{\theta}_t$. For each probing sample $(x_i, y_i) \in D_{\text{probe}}$, the Mimic Score at step t is the projection of its negative gradient onto the unit vector along v_t:

$$m_{i,t} := \frac{\langle -g_i(\tilde{\theta}_t), v_t \rangle}{\|v_t\|}. \tag{1}$$

Intuitively, $m_{i,t}$ reflects how strongly sample i would have pulled the model along the direction actually taken during fine-tuning. ***Samples from domains heavily represented in D_{ref} should therefore yield relatively larger alignment than those from underrepresented ones***. Since $\|v_t\|$ shrinks as $\alpha_t \to 1$, raw scores at different steps are not directly comparable; we therefore apply min-max normalization to $\{m_{i,t}\}_i$ within each step t.

Domain-level Pooling. After normalization, we aggregate per-sample derived scores into a domain-level signal by averaging within each domain k at each pseudo-checkpoint:

$$\bar{m}_{k,t} := \frac{1}{|D_k|} \sum_{i \in D_k} m_{i,t}.$$

Stacking these pooled scores across all K domains and T pseudo-checkpoints yields a feature matrix $M \in \mathbb{R}^{K \times T}$, which serves as the ***geometric footprint*** of the simulated trajectory: ***row k traces how strongly domain k aligns with the fine-tuning direction as the model evolves from θ_{base} toward θ_{ref}.***

3.3 Mapping from Geometry to Mixture

The final step is to translate the geometric footprint M into a predicted mixture $\hat{\pi} \in \Delta^{K-1}$. We instantiate this mapping in two ways: using an *unsupervised* variant that reads the mixture directly off M, or leveraging a *supervised* variant that learns the mapping from synthetic labeled pairs.

Unsupervised Readout. The simplest mapping converts M into a mixture estimate without any learnable parameters, by averaging the pooled scores across pseudo-checkpoints and applying a softmax over domains:

$$\hat{\pi}_k = \frac{\exp(\dot{m}_k/\tau)}{\sum_{k'=1}^{K}\exp(\dot{m}_{k'}/\tau)}, \quad \dot{m}_k = \frac{1}{T}\sum_{t=1}^{T}\bar{m}_{k,t}, \tag{2}$$

where $\tau > 0$ is a temperature controlling the sharpness of the predicted distribution. This readout is *fast and label-free*, but by collapsing the temporal dimension it cannot exploit how domain influence varies across early and late learning.

Supervised Projection. Our second variant trains a projector using synthetic pairs constructed from the same data source S. Concretely, we draw J random mixtures $\pi^{(j)}$ from a distribution Π over the probability simplex, designed so that $\|\pi^{(j)} - \frac{1}{K}\mathbf{1}\|_1$ is uniform. For each sampled mixture, we draw a new dataset from S accordingly and fine-tune the base model on it to obtain a synthetic reference model, then compute its footprint $M^{(j)}$ using the probing dataset D_{probe}. Moreover, since each fine-tune is only a means of *generating training signal* for the projector—not a model intended for downstream use—*we do not need to optimize these mixtures*. ***That is, cheap, short fine-tuning runs are acceptable, making synthetic pairs efficient to produce at scale***. On these pairs, we train an MLP with a softmax output layer, $f_\phi : \mathbb{R}^{K\times T} \to \Delta^{K-1}$, that maps the full feature matrix to a predicted mixture $\hat{\pi} = f_\phi(M)$, using a standard regression loss against the synthetic ground-truth mixture $\pi^{(j)}$. By operating on the full M rather than its temporal average, this learns to weight different learning stages adaptively, capturing how domain influence evolves along the simulated trajectory.

4 Experiments

We evaluate WARP in a controlled setting where the ground-truth domain mixture is known, allowing us to directly measure the recovery effectiveness. Our goal is to verify:

1. **Effectiveness of Weight-Space Geometry.** Recovering domain mixtures from weight-space geometry substantially outperforms random guessing and sample-level membership inference baselines.
2. **Efficiency of Unsupervised Readout.** Even without any learned projector, a parameter-free softmax readout can recover domain proportions accurately.
3. **Simulated Paths Suffice.** Pseudo-checkpoints obtained via model merging can simulate the true learning path, matching—and sometimes outperforming—the variant granted access to real checkpoints.
4. **Robustness Across Training Recipes.** WARP remains accurate on both converged and overtrained checkpoints, mirroring the post-training regime where current models are pushed beyond compute-optimal budgets.

Setups. We use four text datasets to evaluate WARP, spanning different domain structures: (i) SNLI [36] (3 classes, natural language inference), (ii) AGNEWS [37] (4 classes, news topic classification), (iii) YELP [37] (5 classes, review sentiment), and (iv) YAHOO [37] (10 classes, topic classification). We treat each class as a representative domain. For every dataset, we sample 40 ground-truth mixtures $\pi^\star$ over Δ^{K-1} to construct a 5,000-sized dataset and fine-tune both BERT-BASE and GPT-2-SMALL on data drawn according to each $\pi^\star$ to obtain θ_{ref}, using the corresponding pretrained weights as θ_{base}. The probing dataset D_{probe} is drawn separately from the same source S, and $|D_{\text{probe}}| = 2{,}500$ examples sampled uniformly per domain. Pseudo-checkpoints are constructed at $T = 15$ evenly spaced interpolation steps between θ_{base} and θ_{ref}. Recovery performance is reported using mean absolute error (MAE) between $\hat{\pi}$ and $\pi^\star$, averaged across the 40 trials. We provide our training configurations in Appendix B.

Baselines. We compare WARP against *three* baselines:

- **Random Guess**: A mixture sampled from Δ^{K-1}, serving as an uninformed lower bound.
- **Centroid Guess**: A uniform mixture $\hat{\pi} = \frac{1}{K}\mathbf{1}$, acting as the best constant predictor without observing θ_{ref}.

Table 1: Domain mixture recovery error (MAE between predicted $\hat{\pi}$ and ground-truth $\pi^\star$, averaged over 40 trials) on BERT and GPT-2-Small across datasets. Lower is better. Best results within each model block are **bolded**; second-best are underlined.

Method	BERT					GPT-2-Small				
	SNLI (3 cls.)	**AGNews** (4 cls.)	**Yelp** (5 cls.)	**Yahoo** (10 cls.)	**Avg. MAE**	**SNLI** (3 cls.)	**AGNews** (4 cls.)	**Yelp** (5 cls.)	**Yahoo** (10 cls.)	**Avg. MAE**
Baselines										
Random Guess	0.294	0.239	0.192	0.090	0.204	0.294	0.239	0.192	0.090	0.204
Centroid Guess	0.220	0.179	0.149	0.069	0.154	0.220	0.179	0.149	0.069	0.154
Sample-level MI	**0.074**	0.084	0.064	**0.029**	0.063	0.214	0.200	**0.091**	**0.059**	0.141
Our Method: Unsupervised Softmax										
Real Trajectory	0.295	0.321	0.180	0.061	0.214	0.205	0.152	0.168	0.092	0.154
SLERP	0.202	0.142	0.106	0.068	0.130	0.362	0.308	0.262	0.125	0.264
LERP	0.091	0.118	0.086	0.049	0.086	0.137	0.185	0.131	0.064	0.130
TIES	0.091	0.118	0.087	0.049	0.086	0.138	0.175	0.130	0.064	0.127
Our Method: Supervised 2-Layer MLP										
Real Trajectory	0.142	0.049	0.048	0.055	0.074	0.151	0.127	0.137	0.069	0.121
SLERP	0.209	0.110	0.094	0.088	0.130	0.232	0.211	0.173	0.075	0.172
LERP	0.092	**0.031**	**0.022**	0.042	0.047	**0.106**	**0.115**	0.129	0.064	**0.104**
TIES	0.087	0.034	0.023	0.040	**0.046**	0.108	0.130	0.135	0.063	0.109

- **Sample-level MI**: We score each probing example by its loss reduction from θ_{base} to θ_{ref}, sweep the reduction threshold to *maximize detection accuracy*, and aggregate the resulting per-domain membership rates into a mixture estimate. We treat this as the closest sample-based dataset recovery to our weight-space approach.

To isolate the effect of *simulation*, we further include an *oracle variant* of WARP that operates on real intermediate checkpoints saved during fine-tuning, in place of pseudo-checkpoints from merging. For the merging operator $\mathcal{M}$, we evaluate three choices: **SLERP** (spherical interpolation), **LERP** (linear interpolation), and **TIES** [31].

Main Results. Table 1 reports WARP's recovery accuracy, supporting three observations. **First, weight-space geometry carries a strong signal for estimation.** WARP's best supervised variant attains 0.046 MAE on BERT and 0.104 on GPT-2-Small, reducing the strongest baseline's error by 35% and 30% respectively—well below random guessing and sample-level membership inference. **Second, unsupervised readout is competitive.** Even without a learned projector—which would require collecting an additional synthetic training set—the parameter-free softmax readout already outperforms all baselines and closes most of the gap to the supervised variant. **Third, simulated trajectories match or surpass real ones.** On BERT, supervised LERP and TIES both reach 0.048 avg. MAE versus 0.080 for the oracle on real checkpoints; on GPT-2-Small, LERP reaches 0.117 versus 0.138. We attribute this to a *smoothness gap*: real fine-tuning paths can be noisy—stochastic mini-batches, learning-rate warmup, and curriculum effects inject variance into per-step alignment scores. LERP and TIES instead create a clean monotone path from θ_{base} to θ_{ref}, yielding a smoother footprint matrix M from which the 2-layer MLP can more easily extract the mixture. Overall, we demonstrate the feasibility of leveraging weight-space geometry and the dataset footprints to reverse-engineer the training mixture.

Different Training Recipes. An interesting question when deploying our method is which reference checkpoint to distill alignment matrices from, since the right choice is often unclear in practice—practitioners may stop training early to save compute, train to convergence, or inadvertently overtrain. Table 2 evaluates this sensitivity by extracting geometric footprints at three different checkpoints along the GPT-2-Small fine-tuning trajectory on AG News: including short runs (9 epochs, mirroring early stopping), converged checkpoints (12 epochs), and overtrained checkpoints (18 epochs, well past convergence). Across

Table 2: Domain mixture recovery error on GPT-2-Small (using AG News and TIES merging) across different training stages. Both methods remain robust at recovering different types of checkpoint, with the supervised MLP consistently outperforming all baselines. Lower is better.

	Short Runs (9 ep.)		**Converged** (12 ep.)		**Overtrained** (18 ep.)	
Method	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
Baselines						
Random Guess	0.095	0.239	0.095	0.239	0.095	0.239
Centroid Guess	0.048	0.179	0.048	0.179	0.048	0.179
Sample-level MI	0.068	0.200	0.068	0.200	0.068	0.200
Our Methods						
Unsupervised Softmax	0.053	0.175	0.060	0.183	0.064	0.193
Supervised 2-Layer MLP	**0.033**	**0.130**	**0.040**	**0.147**	**0.030**	**0.124**

all three stages, both our unsupervised softmax and supervised MLP variants outperform the baselines, with the supervised MLP achieving the lowest recovery error. This demonstrates our framework's robustness when facing diverse training recipes.

5 Conclusion

In this work, we introduce WARP, a framework that recovers a fine-tuned model's training domain mixture from its weights alone. By using model merging to simulate the missing training trajectory, WARP extracts a geometric footprint of the training data and maps it to domain proportions. In controlled experiments, WARP achieves MAE as low as 0.046 on BERT and 0.104 on GPT-2, outperforming membership inference baselines and even an oracle with access to the true trajectory. Overall, we demonstrate the feasibility of recovering a model's training distribution from its weight-space geometry—a step toward greater transparency in an era where model weights are shared but data recipes are not.

References

[1] Chen, Z.; Miao, Y.; Xiong, D.; others Data Mixing for Large Language Models Pretraining: A Survey and Outlook. *arXiv preprint arXiv:2604.16380* **2026**,

[2] Xie, S. M.; Pham, H.; Dong, X.; Du, N.; Liu, H.; Lu, Y.; Liang, P. S.; Le, Q. V.; Ma, T.; Yu, A. W. Doremi: Optimizing data mixtures speeds up language model pretraining. *Advances in Neural Information Processing Systems* **2023**, *36*, 69798–69818.

[3] Ge, A.; Huang, T.-H.; Cooper, J.; Trost, A.; Chu, Z.; GNVV, S. S. S. N.; Cai, Z.; Park, K.; Roberts, N.; Sala, F. R&B: Domain Regrouping and Data Mixture Balancing for Efficient Foundation Model Training. *arXiv preprint arXiv:2505.00358* **2025**,

[4] Chen, M. F.; Hu, M. Y.; Lourie, N.; Cho, K.; Ré, C. Aioli: A unified optimization framework for language model data mixing. *arXiv preprint arXiv:2411.05735* **2024**,

[5] Fan, S.; Pagliardini, M.; Jaggi, M. Doge: Domain reweighting with generalization estimation. *arXiv preprint arXiv:2310.15393* **2023**,

[6] Golchin, S.; Surdeanu, M. Time Travel in LLMs: Tracing Data Contamination in Large Language Models. *CoRR* **2023**, *abs/2308.08493*.

[7] Golchin, S.; Surdeanu, M. Data Contamination Quiz: A Tool to Detect and Estimate Contamination in Large Language Models. *CoRR* **2023**, *abs/2311.06233*.

[8] Magar, I.; Schwartz, R. Data contamination: From memorization to exploitation. Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). 2022; pp 157–165.

[9] Singh, C.; Inala, J. P.; Galley, M.; Caruana, R.; Gao, J. Rethinking interpretability in the era of large language models. *arXiv preprint arXiv:2402.01761* **2024**,

[10] Wen, B.; Salekin, S.; Kang, F.; Howe, B.; Wang, L. L.; Movellan, J.; Bilkhu, M. MixAtlas: Uncertainty-aware Data Mixture Optimization for Multimodal LLM Midtraining. *arXiv preprint arXiv:2604.14198* **2026**,

[11] Liu, Q.; Zheng, X.; Muennighoff, N.; Zeng, G.; Dou, L.; Pang, T.; Jiang, J.; Lin, M. Regmix: Data mixture as regression for language model pre-training. International Conference on Learning Representations. 2025; pp 38305–38339.

[12] Kang, F.; Sun, Y.; Wen, B.; Chen, S.; Song, D.; Mahmood, R.; Jia, R. Autoscale: Scale-aware data mixing for pre-training llms. *arXiv preprint arXiv:2407.20177* **2024**,

[13] Shukor, M.; Bethune, L.; Busbridge, D.; Grangier, D.; Fini, E.; El-Nouby, A.; Ablin, P. Scaling laws for optimal data mixtures. *Advances in Neural Information Processing Systems* **2026**, *38*, 129554–129579.

[14] Fu, W.; Wang, H.; Gao, C.; Liu, G.; Li, Y.; Jiang, T. Membership inference attacks against fine-tuned large language models via self-prompt calibration. *Advances in Neural Information Processing Systems* **2024**, *37*, 134981–135010.

[15] Duan, M.; Suri, A.; Mireshghallah, N.; Min, S.; Shi, W.; Zettlemoyer, L.; Tsvetkov, Y.; Choi, Y.; Evans, D.; Hajishirzi, H. Do membership inference attacks work on large language models? *arXiv preprint arXiv:2402.07841* **2024**,

[16] Mattern, J.; Mireshghallah, F.; Jin, Z.; Schölkopf, B.; Sachan, M.; Berg-Kirkpatrick, T. Membership inference attacks against language models via neighbourhood comparison. Findings of the Association for Computational Linguistics: ACL 2023. 2023; pp 11330–11343.

[17] Shokri, R.; Stronati, M.; Song, C.; Shmatikov, V. Membership Inference Attacks against Machine Learning Models. 2017; https://arxiv.org/abs/1610.05820.

[18] Morris, J. X.; Yin, J. O.; Kim, W.; Shmatikov, V.; Rush, A. M. Approximating Language Model Training Data from Weights. *arXiv preprint arXiv:2506.15553* **2025**,

[19] Cazenavette, G.; Wang, T.; Torralba, A.; Efros, A. A.; Zhu, J.-Y. Dataset distillation by matching training trajectories. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022; pp 4750–4759.

[20] Huang, T.-H.; Bilkhu, M.; Cooper, J.; Sala, F.; Movellan, J. Evaluating Sample Utility for Efficient Data Selection by Mimicking Model Weights. *arXiv preprint arXiv:2501.06708* **2025**,

[21] Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers). 2019; pp 4171–4186.

[22] Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; others Language models are unsupervised multitask learners. *OpenAI blog* **2019**, *1*, 9.

[23] Chen, M. F.; Murray, T.; Heineman, D.; Jordan, M.; Hajishirzi, H.; Ré, C.; Soldaini, L.; Lo, K. Olmix: A framework for data mixing throughout lm development. *arXiv preprint arXiv:2602.12237* **2026**,

[24] Albalak, A.; Pan, L.; Raffel, C.; Wang, W. Y. Efficient online data mixing for language model pre-training. *arXiv preprint arXiv:2312.02406* **2023**,

[25] Maini, P.; Jia, H.; Papernot, N.; Dziedzic, A. LLM Dataset Inference: Did you train on my dataset? 2024.

[26] Yang, E.; Shen, L.; Guo, G.; Wang, X.; Cao, X.; Zhang, J.; Tao, D. Model merging in llms, mllms, and beyond: Methods, theories, applications, and opportunities. *ACM Computing Surveys* **2026**, *58*, 1–41.

[27] Goddard, C.; Siriwardhana, S.; Ehghaghi, M.; Meyers, L.; Karpukhin, V.; Benedict, B.; McQuade, M.; Solawetz, J. Arcee's MergeKit: A Toolkit for Merging Large Language Models. Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track. Miami, Florida, US, 2024; pp 477–485.

[28] Ilharco, G.; Ribeiro, M. T.; Wortsman, M.; Gururangan, S.; Schmidt, L.; Hajishirzi, H.; Farhadi, A. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089* **2022**,

[29] Ainsworth, S. K.; Hayase, J.; Srinivasa, S. Git re-basin: Merging models modulo permutation symmetries. *arXiv preprint arXiv:2209.04836* **2022**,

[30] Jordan, K.; Sedghi, H.; Saukh, O.; Entezari, R.; Neyshabur, B. Repair: Renormalizing permuted activations for interpolation repair. *arXiv preprint arXiv:2211.08403* **2022**,

[31] Yadav, P.; Tam, D.; Choshen, L.; Raffel, C. A.; Bansal, M. Ties-merging: Resolving interference when merging models. *Advances in neural information processing systems* **2023**, *36*, 7093–7115.

[32] Yu, L.; Yu, B.; Yu, H.; Huang, F.; Li, Y. Language Models are Super Mario: Absorbing Abilities from Homologous Models as a Free Lunch. International Conference on Machine Learning. 2024.

[33] He, Y.; Hu, Y.; Lin, Y.; Zhang, T.; Zhao, H. Localize-and-stitch: Efficient model merging via sparse task arithmetic. *arXiv preprint arXiv:2408.13656* **2024**,

[34] Wortsman, M.; Ilharco, G.; Gadre, S. Y.; Roelofs, R.; Gontijo-Lopes, R.; Morcos, A. S.; Namkoong, H.; Farhadi, A.; Carmon, Y.; Kornblith, S.; others Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. International conference on machine learning. 2022; pp 23965–23998.

[35] Yang, E.; Shen, L.; Guo, G.; Wang, X.; Cao, X.; Zhang, J.; Tao, D. Model Merging in LLMs, MLLMs, and Beyond: Methods, Theories, Applications, and Opportunities. *ACM Comput. Surv.* **2026**, *58*.

[36] Young, P.; Lai, A.; Hodosh, M.; Hockenmaier, J. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics* **2014**, *2*, 67–78.

[37] Zhang, X.; Zhao, J.; LeCun, Y. Character-level convolutional networks for text classification. *Advances in neural information processing systems* **2015**, *28*.

Appendix Roadmap

Our appendix is structured as follows. We use Appendix A to summarize our full framework in pseudocode. Appendix B documents the experimental setup and training configurations. Finally, Appendix C outlines directions for future work.

A Algorithm

Next, we summarize the full WARP procedure in Algorithm 1. WARP proceeds in three stages: simulating a trajectory between θ_{base} and θ_{ref} via model merging (Sec. 3.1), distilling each pseudo-checkpoint into a domain-level geometric footprint via Mimic Scores (Sec. 3.2), and mapping the resulting footprint matrix to a predicted mixture through either an unsupervised softmax readout or a supervised projector trained on synthetic pairs (Sec. 3.3).

Algorithm 1 WARP: Weight-Space Analysis for Recovering Data Portfolios

Require: Base model $\boldsymbol{\theta}_{\text{base}}$, fine-tuned model $\boldsymbol{\theta}_{\text{ref}}$, data source S over K domains, merging operator $\mathcal{M}$ with schedule $\{\alpha_t\}_{t=1}^{T}$, mode $\in \{\textsc{Unsup},\textsc{Sup}\}$, temperature τ, (SUP only) mixture distribution Π and # pairs J.
Ensure: Estimated domain mixture $\hat{\boldsymbol{\pi}} \in \Delta^{K-1}$.

Initialize: probing dataset $D_{\text{probe}} = \bigcup_{k=1}^{K} D_k$ sampled from S with known labels

// Stage 1: Simulate trajectory *// Stage 2: Distill footprint*

for $t = 1$ **to** T **do**
  $\tilde{\boldsymbol{\theta}}_t \leftarrow \mathcal{M}(\boldsymbol{\theta}_{\text{base}}, \boldsymbol{\theta}_{\text{ref}}; \alpha_t)$, $\quad \mathbf{v}_t \leftarrow \boldsymbol{\theta}_{\text{ref}} - \tilde{\boldsymbol{\theta}}_t$ ▷ Pseudo-checkpoint and direction
  $m_{i,t} \leftarrow \langle -\nabla \ell(x_i, y_i; \tilde{\boldsymbol{\theta}}_t), \mathbf{v}_t \rangle / \|\mathbf{v}_t\|$ for $(x_i, y_i) \in D_{\text{probe}}$ ▷ Mimic Score
  Min-max normalize $\{m_{i,t}\}_i$; $\quad \bar{m}_{k,t} \leftarrow \frac{1}{|D_k|} \sum_{i \in D_k} m_{i,t}$ ▷ Pool by domain
end for
$M \leftarrow [\bar{m}_{k,t}]_{K \times T}$ ▷ Geometric footprint

// Stage 3: Map footprint to mixture

if Mode is UNSUP **then**
  $\hat{\pi}_k \leftarrow \text{softmax}_k\left(\frac{1}{T} \sum_t \bar{m}_{k,t} / \tau\right)$ ▷ Label-free readout
else
  for $j = 1$ **to** J **do**
    Sample $\boldsymbol{\pi}^{(j)} \sim \Pi$, draw $D^{(j)}$ from S with mixture $\boldsymbol{\pi}^{(j)}$, fine-tune $\boldsymbol{\theta}_{\text{base}}$ on $D^{(j)}$ to get $\boldsymbol{\theta}_{\text{ref}}^{(j)}$
    $M^{(j)} \leftarrow$ Stages 1–2 with $(\boldsymbol{\theta}_{\text{base}}, \boldsymbol{\theta}_{\text{ref}}^{(j)}, D_{\text{probe}})$
  end for
  Train $f_\phi : \mathbb{R}^{K \times T} \to \Delta^{K-1}$ on $\{(M^{(j)}, \boldsymbol{\pi}^{(j)})\}_{j=1}^{J}$; $\quad \hat{\boldsymbol{\pi}} \leftarrow f_\phi(M)$
end if
return $\hat{\boldsymbol{\pi}}$

B Experimental Details

Here we provide additional details of our experimental setup. We discuss (i) how reference models are constructed, (ii) how the probing dataset is sampled, (iii) the supervised projector training pipeline, and (iv) the computational resources used.

Constructing Reference Models. We validate WARP in a controlled setting where the ground-truth mixture $\pi^\star$ is known. For each dataset, we sample 40 mixtures $\{\pi_i^\star\}_{i=1}^{40}$ from the simplex Δ^{K-1}. For each $\pi_i^\star$, we draw 5,000 training examples whose per-domain proportions follow $\pi_i^\star$, and fine-tune both BERT and GPT-2-SMALL for 9 epochs starting from their publicly released pretrained weights, which we use as θ_{base}. For each reference model, we sweep the learning rate over

Table 3: Domain mixture recovery error (MSE between predicted $\hat{\pi}$ and ground-truth $\pi^\star$, averaged over 40 trials) on BERT and GPT-2-Small across datasets. Lower is better. Best results within each model block are **bolded**; second-best are underlined.

Method	BERT					GPT-2-Small				
	SNLI (3 cls.)	AGNews (4 cls.)	Yelp (5 cls.)	Yahoo (10 cls.)	Avg. MSE	SNLI (3 cls.)	AGNews (4 cls.)	Yelp (5 cls.)	Yahoo (10 cls.)	Avg. MSE
Baselines										
Random Guess	0.140	0.095	0.066	0.014	0.079	0.140	0.095	0.066	0.014	0.079
Centroid Guess	0.072	0.048	0.032	0.007	0.040	0.072	0.048	0.032	**0.007**	0.040
Sample-level MI	**0.009**	0.012	0.007	**0.001**	0.007	0.073	0.068	**0.017**	0.008	0.042
Our Method: Unsupervised Softmax										
Real Trajectory	0.156	0.163	0.062	0.007	0.097	0.070	0.046	0.058	0.019	0.048
SLERP	0.071	0.029	0.019	0.008	0.032	0.197	0.163	0.121	0.036	0.130
LERP	0.013	0.021	0.013	0.004	0.013	0.036	0.062	0.032	0.008	0.035
TIES	0.013	0.021	0.013	0.004	0.013	0.036	0.053	0.032	0.008	0.032
Our Method: Supervised 2-Layer MLP										
Real Trajectory	0.040	0.005	0.005	0.006	0.014	0.040	0.030	0.040	0.009	0.030
SLERP	0.070	0.018	0.018	0.014	0.034	0.080	0.077	0.053	0.011	0.055
LERP	0.017	**0.002**	**0.001**	0.004	0.006	**0.022**	**0.024**	0.036	0.008	**0.023**
TIES	0.014	**0.002**	**0.001**	0.003	**0.005**	0.023	0.033	0.037	0.008	0.025

$\{6e{-}6, 1e{-}5, 3e{-}5, 5e{-}5, 1e{-}4\}$ and select the value that minimizes training loss to obtain θ_{ref}. This yields 40 reference models per (dataset, architecture) pair.

Creating Probing Dataset. WARP makes a simple assumption about the domain distribution of the probing set; it requires only that D_{probe} shares the same K domains as D_{ref}. We sample $|D_{\text{probe}}| = 2{,}500$ examples from S, with $2{,}500/K$ examples drawn uniformly per domain. The probing dataset is fixed once per (dataset, architecture) pair and reused across all 40 mixtures. For each probing example, we compute its alignment score from the per-sample gradient and the directional vector pointing from each pseudo-checkpoint toward θ_{ref}.

Training Projector. For the supervised variant, the projector is a 2-layer MLP with ReLU activation, and a softmax output layer that produces a vector on Δ^{K-1}. The input is the matrix of domain-level alignment statistics aggregated from D_{probe}, and the target is the ground-truth mixture $\pi^\star$. We train it with a learning rate $1e{-}4$ for 200 epochs. Since we have 40 (mixture, reference-model) pairs per (dataset, architecture), we report results under 5-fold cross-validation: in each fold, 32 pairs train the projector and the held-out 8 pairs are used for evaluation. All the experiments are conducted on an NVIDIA RTX A6000 GPU.

Evaluation. We report MAE between $\hat{\pi}$ and $\pi^\star$, averaged over the 40 ground-truth mixtures in Table 1. We additionally report mean squared error (MSE) in Table 3.

C Future Work

WARP provides the ***first*** mechanism for recovering the domain mixture of a fine-tuned model from its released weights alone. We outline several directions that can extend its scope, including (i) interpreting the learned projector, (ii) strengthening the unsupervised readout, and (iii) scaling to large-scale LLMs.

Interpretability of the Learned Projector. The supervised variant of WARP encodes, in the weights of its MLP projector, an *implicit* model of how Mimic Scores across pseudo-checkpoints map to domain proportions. Understanding this mapping would clarify *which* stages of the simulated trajectory carry the strongest signal for each domain, and *why*. A starting point is to replace the MLP with shallower, fully linear projectors, whose coefficients can be read directly as per-stage importance weights. Combined with a larger set of synthetic pseudo-checkpoints and explicit regularization (e.g., sparsity or smoothness across t), this could begin to expose the temporal structure of domain influence as the model evolves from θ_{base} toward θ_{ref}.

Advanced Unsupervised Readouts. The current unsupervised variant uses two assumptions: that uniform averaging of pooled Mimic Scores across pseudo-checkpoints is a *reliable* aggregation, and that the resulting per-domain quantities behave like logits suitable for a softmax. A more principled alternative would treat the readout as inference in a graphical model over pseudo-checkpoints, where each step contributes a noisy vote on the underlying mixture and *weak supervision* techniques infer per-stage reliabilities without labels. Such unsupervised techniques could adaptively down-weight uninformative stages, potentially narrowing the gap to the supervised variant while preserving the label-free setting.

Scaling to Large-Scale LLMs. Finally, our controlled experiments fine-tune BERT and GPT-2 under known mixtures, which isolates the recovery problem but leaves open how WARP behaves at the scale of modern LLM training. LLMs are typically adapted through long, multi-stage pipelines—continued mid-training, supervised fine-tuning, and preference optimization—over corpora spanning dozens of domains. Extending WARP to these regimes raises two questions in particular: whether linear interpolation remains a useful proxy for trajectories that traverse qualitatively different objectives, and how the geometric footprint behaves when domains are numerous and more complex. These directions will help extend WARP from a controlled diagnostic tool toward a recovery tool applicable to frontier models.