TopoTuner: Topological Finetuning of Large Language Models

Anonymous authors
Paper under double-blind review

Abstract

Full fine-tuning makes all pretrained parameters trainable and can be computationally expensive. LoRA reduces the number of trainable parameters, but it does not directly answer which pretrained components should be trained and which can be frozen during adaptation. We introduce TopoTuner, a topology-guided fine-tuning framework for selective freezing of attention projection matrices. TopoTuner treats each projection matrix as a row cloud and uses Wasserstein distances between persistence diagrams to measure how its topology changes during fine-tuning.

TopoTuner learns a reusable freezing profile from a source dataset and reuses the selected matrix indices to efficiently fine-tune models on unseen target datasets, both within and outside the source domain. We evaluate whether the same freezing decisions remain effective across question answering and sentiment analysis tasks.

Across LLaMA-3.1-8B, Mistral-7B-v0.3, and Qwen3-8B-Base, TopoTuner is competitive with full fine-tuning while updating only 0.56%–4.85% of pretrained parameters and outperforms LoRA in 8 out of 9 model-dataset settings. Along with minimized updates, TopoTuner reduces training time by 16.5% relative to full fine-tuning and 5.3% relative to LoRA on average. TopoTuner opens a new direction for Efficient AI where matrix-freezing decisions learned on one dataset can be reused across multiple tasks.

1 Introduction

Fine-tuning Kumar et al. (2025); Xu et al. (2026) is the standard mechanism for adapting pretrained large language models (LLMs) to downstream tasks, such as question answering Yue (2025). Full fine-tuning updates all model weights and remains a strong baseline, but it is expensive and can damage capabilities acquired during pretraining with risks of catastrophic forgetting Luo et al. (2023), overfitting to narrow task distributions Huang et al. (2025), and instability in alignment behavior Liu et al. (2025).

Parameter-efficient methods (PEFT) Xu et al. (2026), such as LoRA Hu et al. (2022), reduce the number of trainable parameters and often recover strong task performance. However, despite using low-rank updates, LoRA still modifies a substantial fraction (17.02% of all weights in our experiments) of the pretrained model without explicit control over which components are changed. Furthermore, matching downstream accuracy does not answer a basic structural question: which parts of the pretrained model actually need to change during adaptation?

The question matters because fine-tuning is rarely single-task in practice Hu et al. (2025). Models are adapted across domains and often reused across datasets. A method that only controls the number of trainable parameters does not show which parts of the model change, whether the same parts change across tasks, or at what epoch the main structural change has already occurred. Standard summaries such as spectra, norms, and sparsity measure the size or rank structure of an update, but they do not directly describe how the structure of a weight matrix changes Shuttleworth et al. (2025).

Topology Carlsson (2009) offers a principled way to study how model parameters reorganize during fine-tuning. We treat each attention projection matrix as a point cloud formed by its rows and use topological data analysis Chazal & Michel (2021) to obtain stable multiscale summaries of this geometry. The distance between the persistence diagrams of pretrained and fine-tuned matrices then gives a matrix-level measure of structural change, which we use to rank matrices for freezing.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

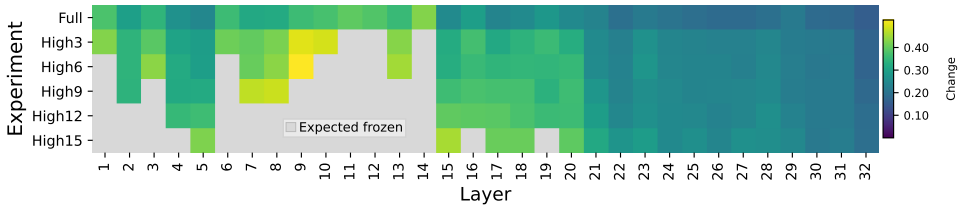


Figure 1: LLaMA O-projection change under high-distance Wasserstein freezing (3 to 15 layers). As more matrices are frozen during training, the remaining trainable matrices absorb updates in a more concentrated manner; selective freezing redistributes adaptation across layers.

This view also suggests a sufficient condition for reusing such freezing profiles across tasks: the fine-tuning trajectories need not be identical; it suffices that the two task-induced gradient fields remain close and the source ranking has a sufficient margin.

This insight allows us to introduce TopoTuner, a topology-guided Efficient AI Menghani (2023) framework for measuring and controlling LLM fine-tuning. The method tracks topological distance, the structural change of each weight matrix relative to its pretrained state, and uses this signal to build reusable freezing profiles and to stop training once the main structural change has occurred. This reframes fine-tuning from a parameter-count problem into a structural-control problem: instead of asking only how many parameters to update, TopoTuner asks which pretrained components should be modified during adaptation.

We evaluate TopoTuner on three language models, using profile reuse across question-answering and sentiment-analysis tasks and broader retention evaluation spanning six task families: question answering, sentiment analysis, information retrieval, summarization, instruction following, and code generation. However, for a fixed model, we show that a profile learned from one dataset can be reused on unseen datasets, reducing full fine-tuning time by 16.5% and LoRA fine-tuning time by 5.3% on average while maintaining competitive accuracy.

Our contributions are as follows.

- We introduce TopoTuner, a topology-guided framework that measures how attention projection matrices reorganize during fine-tuning and uses this signal to observe and control which matrices remain trainable. **To our knowledge, this is the first work that uses topological summaries of LLM weights to construct reusable freezing profiles across tasks.**
- We define matrix-level topological change through Wasserstein distances between persistence diagrams of pretrained and fine-tuned weights. We show for the first time that topological change can guide reusable matrix-level freezing decisions across unseen in-domain and out-of-domain tasks, opening a new direction in efficient AI based on structural control of adaptation.
- TopoTuner achieves the best rank against selective adaptation baselines. Across profile-reuse experiments, TopoTuner outperforms LoRA in 8 out of 9 model-dataset settings while reducing wall-clock training time by 16.5% relative to full fine-tuning and updating only 0.56%–4.85% of pretrained parameters. TopoTuner stops fine-tuning after two to three epochs in most runs, reducing the six-epoch budget by 50%–66.7% while retaining near-final validation accuracy.

2 Related Work

Topological methods. Persistent homology (PH) has emerged as a powerful tool for analyzing neural network representations. Pérez-Fernández et al. (2021) treat a neural network as an abstract simplicial complex and extract its topological fingerprint via persistent homology. The authors define a PH-based descriptor that can serve as a similarity measure between trained models. Moreover, persistent homology can reveal layer-wise organizational differences. Purvine et al. analyze hidden representations of CNNs by treating each layer's activations as a high-dimensional point cloud Purvine et al. (2023). These works establish that persistent homology, via persistence diagrams and their distances, provides a rigorous framework to quantify structural similarities in deep networks. Prior topology-guided work also modifies soft prompts Huang et al. (2026). However, to the best of our knowledge, no prior work has used topological signals to guide LLM fine-tuning via selective freezing.

Fine-tuning strategies. Fine-tuning large transformer models can be approached at the granularity of individual weight matrices or entire layers. A variety of matrix-level adaptation techniques, often termed parameter-efficient fine-tuning (PEFT Xu et al. (2026)), restrict which parameters are updated to reduce computational cost. A prominent example is LoRA Hu et al. (2022), which inserts low-rank update matrices into weight layers. Building on this idea, "Deep LoRA" Yaras et al. (2024) applies a low-rank decomposition to multiple transformer layers simultaneously, to confine the fine-tuning updates to a low-dimensional subspace at each layer. This strategy mitigates overfitting with limited data. On the other end of the spectrum, layer-level strategies decide which whole layers to train or freeze. Zheng et al. (2025) freeze the bottom 25% of transformer layers, and achieve better downstream performance in continual learning scenarios. In addition to static freezing, recent methods dynamically adjust training at the layer level. A representative approach is DropBP, which randomly omits gradient backpropagation through certain layers at each update, guided by a sensitivity metric that estimates each layer's impact on the training loss Woo et al. (2024). Layers with lower sensitivity receive a higher drop rate, effectively being temporarily frozen during some training steps. This adaptive schedule trains only the most "critical" layers at full frequency and reduces the time costs. In contrast, we use topological distance to measure how each attention projection matrix reorganizes relative to the pretrained model, and use this signal to build reusable freezing profiles and to decide when fine-tuning has stabilized.

3 PRELIMINARIES

Let f_θ denote a pretrained large language model, parameterized by a collection of weight matrices $\theta = \{W_i\}_{i=1}^m$, where $W_i \in \mathbb{R}^{d_i \times d_i'}$. We use $\mathcal{I} \subseteq \{1, \dots, m\}$ to denote the index set of matrices.

Task and fine-tuning. Given a downstream task $\mathcal{T}$ with training data $\mathcal{D}_\mathcal{T} = \{(x_r, y_r)\}_{r=1}^N$, fine-tuning solves $\tilde{\theta} = \arg\min_\vartheta \frac{1}{N} \sum_{r=1}^N \ell\big(y_r,\ f_\vartheta(x_r)\big)$, where ℓ is a supervised loss function. The resulting model is denoted by $f_{\tilde{\theta}}$ with parameters $\tilde{\theta} = \{\tilde{W}_i\}_{i=1}^m$.

Topological tools. Persistent homology is a tool from topological data analysis that summarizes how topology evolves across a filtration (Chazal & Michel, 2021; Hensel et al., 2021). Let $\mathcal{X} = x_1, \dots, x_n \subset \mathbb{R}^d$ be a point cloud equipped with a distance function δ. We construct the Vietoris–Rips filtration $\{\mathrm{VR}_\varepsilon(\mathcal{X})\}_{\varepsilon \geq 0}$, where simplices are added according to pairwise distance at scale ε Chazal & Michel (2021). For simplicity, we write this filtration as $\mathcal{K}_1 \subset \mathcal{K}_2 \subset \cdots \subset \mathcal{K}_n$.

Topological features and persistence diagrams. For each complex $\mathcal{K}_i$ in the filtration, let $H_k(\mathcal{K}_i)$ denote its kth homology group, whose nontrivial classes represent k-dimensional topological features such as connected components for $k = 0$, loops for $k = 1$, and voids for $k = 2$. As the filtration grows, features appear and disappear; $q_j = (b_j, d_j) \in \mathrm{PD}(\mathcal{X})$ represents the birth and death times of a topological feature in $\mathcal{X}$ within a Persistence Diagram (PD). The kth persistence diagram, denoted by $\mathrm{PD}_k(\mathcal{X})$, is the multiset of birth-death pairs for all such features. Features with larger persistence $d_j - b_j$ are often interpreted as more salient.

p-Wasserstein distance. Given two persistence diagrams $\mathrm{PD_k}(\mathcal{X})$ and $\mathrm{PD_k}(\mathcal{X}')$ in the same homological dimension k, their p-Wasserstein distance is $\mathcal{W}ass_p(\mathrm{PD_k}(\mathcal{X}), \mathrm{PD_k}(\mathcal{X}')) = \min_\phi \left(\sum_{\mathrm{j}} \|\mathrm{q_j} - \phi(\mathrm{q_j})\|_\infty^{\mathrm{p}}\right)^{1/\mathrm{p}}$, where ϕ ranges over partial matchings with diagonal matching allowed. This distance measures how much the underlying topological summaries are different between two point clouds.

4 TOPOLOGICAL CHARACTERISTICS OF LLM FINE-TUNING

Topological data analysis is based on the principle that data has shape and shape has meaning Carlsson (2009). Accordingly, we treat each projection (i.e., weight matrix) in an LLM as a geometric object and study how its internal structure evolves during fine-tuning across epochs. Our topology-based hypothesis is that persistence diagrams of a projection provide a topological summary of this organization, and that the distance between pre- and post-fine-tuning diagrams yields a useful projection-level measure of structural change. Our goal is to use this measure to make fine-tuning more controlled: identify which projections can be frozen, reuse those freezing profiles across datasets, and stop training once the remaining trainable projections have stabilized.

Let f_θ be a pretrained transformer with layer set $\mathcal{L} = \{1, \ldots, L\}$. For each layer $l \in \mathcal{L}$, let $W_l^{(q)}$, $W_l^{(k)}$, $W_l^{(v)}$, and $W_l^{(o)}$ denote the query, key, value, and output projections. Let $\mathcal{I}$ denote the collection of matrices considered for ranking and freezing.

Topological sensitivity pipeline. For each matrix $W_i \in \mathbb{R}^{d_i \times d_i'}$, we form the row cloud $X_i = \{x_1^{(i)}, \ldots, x_{d_i}^{(i)}\} \subset \mathbb{R}^{d_i'}$, where $x_a^{(i)}$ is the ath row of W_i. We equip X_i with a distance metric δ_i and, for each threshold $\varepsilon \geq 0$, build the graph $\mathcal{G}_\varepsilon^{(i)}$ with vertex set X_i and edges between two data points (a, b) whenever their distance $\delta_i(a, b) \leq \varepsilon$. Promoting every clique in $\mathcal{G}_\varepsilon^{(i)}$ to a simplex gives the Vietoris–Rips complex $\mathrm{VR}_\varepsilon^{(i)}$ Hensel et al. (2021), and the nested family $\{\mathrm{VR}_\varepsilon^{(i)}\}_{\varepsilon \geq 0}$ defines a filtration. From this filtration we compute persistence diagrams $PD_k(X_i)$ in homological dimensions $k \in \{0, 1, \ldots\}$.

Row cloud. We use rows because each row defines an output channel, whereas columns describe input-coordinate contributions across output channels. Therefore row cloud directly represents the geometry of the projection's output directions.

Definition 4.1 (Topological distance to pretrained model)**.** Let $W_i^{(0)}$ be the pretrained version of matrix $i \in \mathcal{I}$, and let $\tilde{W}_i^{(t)}$ denote its version at the fine-tuning epoch $t \geq 1$. For a fixed homological dimension k, we define the topological distance of matrix i at t by $D_i^{(t)} := \mathcal{W}ass_p(\mathrm{PD}_k(X_i^{(0)}), \mathrm{PD}_k(X_i^{(t)}))$.

All distances are measured from the pretrained diagram. We define drift as the normalized change in this distance across epochs.

Dimension. We use H_0, whose finite death times correspond to minimum-spanning-tree edge weights, equivalently the merge scales of single-linkage clustering. We also evaluated H_1, but it led to higher computational cost and produced weaker results than H_0.

Definition 4.2 (Topological drift)**.** Let $D_i^{(t)}$ denote the fine-tuning induced topological distance for matrix i at the fine-tuning epoch t. The first-epoch distance normalizes this change across matrices; we define the topological drift at $t \geq 2$ for $D_i^{(1)} > 0$ by $\Delta_i^{(t)} := \frac{D_i^{(t)} - D_i^{(t-1)}}{D_i^{(1)}}$.

Stability. The usefulness of Δ_i rests on the stability principle of PH: small perturbations in the underlying metric induce small perturbations in the resulting diagrams. In our setting, this means that if fine-tuning changes pairwise distances between rows only slightly, then the corresponding topological change must also remain small. The next proposition makes this statement explicit.

Proposition 4.3 (Stability of topological sensitivity)**.** *Let $X = \{x_1, \ldots, x_d\}$ and $\tilde{X} = \{\tilde{x}_1, \ldots, \tilde{x}_d\}$ be the row clouds of W_i and $\tilde{W}_i$, indexed consistently. Let d and $\tilde{d}$ be the pairwise dissimilarity functions used to construct the Vietoris–Rips filtrations of X and $\tilde{X}$. If $\|d - \tilde{d}\|_\infty := \max_{a,b} |d(a, b) - \tilde{d}(a, b)| \leq \eta$, then for every $k \geq 0$ we have $d_B(PD_k(X), PD_k(\tilde{X})) \leq \eta$, and therefore $W_p(PD_k(X), PD_k(\tilde{X})) \leq N_k^{1/p}\eta$, where d_B is the bottleneck distance and N_k is the maximum number of off-diagonal points in the two diagrams. Due to space limitations, the proof is given in Appendix A.*

5 Freezing Strategies Based on Topological Distances

We use topological distance to guide efficient adaptation and to study how matrix topology reorganizes during fine-tuning. Our framework is visualized in Figure 2. We consider two settings: prior-based freezing and dynamic stopping. The first uses a prior fine-tuning task to construct a fine-tuning profile, which is then reused for fast selective fine-tuning on new tasks and datasets. The second uses topological drift observed during the ongoing fine-tuning run to track structural change across epochs and terminate fine-tuning as early as possible. An algorithm for the freezing procedure is provided in Appendix B.

Scenario A: Fine-tuning with Freezing Priors

We posit that reusable freezing profiles do not require identical fine-tuning trajectories across tasks. However, for a fixed pretrained model, certain projection matrices can repeatedly rank among the most strongly changed across different datasets, making their identities reusable across tasks.

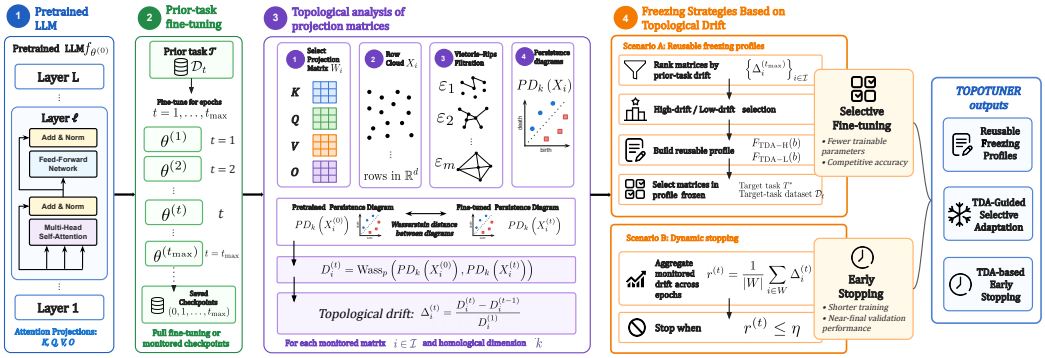


Figure 2: **Overview of TopoTuner.** TopoTuner uses a prior fine-tuning task to compute the topological distance of attention projection matrices. The computed topological distance is used to (i) build reusable freezing profiles for selective fine-tuning on new datasets and tasks, and to (ii) track structural change across epochs for early stopping. Freezing profiles are provided in Appendix C.

Let $\theta^{(t)}$ denote the model obtained by fine-tuning the pretrained network f_θ on a single prior task $\mathcal{T}$ for $t = \{1, \ldots, t_{max}\}$ epochs. For each matrix $W_i \in \mathcal{I}$, we first compute its prior-task topological distance $D_i := \mathcal{W}ass_p(\mathrm{PD}_k(X_i^{(0)}), \mathrm{PD}_k(X_i^{(t_{max})}))$, where $X_i^{(t_{max})}$ is the point cloud of the post-fine-tuning version of W_i. Next, we also measure the magnitude change of W_i by $M_i := \frac{1}{|W_i|}\sum_{a,b}\left|\frac{W_{i,ab}^{(t_{max})} - W_{i,ab}^{(0)}}{|W_{i,ab}^{(0)}|}\right|$ element-wise. We use this magnitude-based change as a baseline to discuss the usefulness of a TDA-guided strategy. Using the prior-task model, we build distance-based matrix rankings that are later reused on a new task $\mathcal{T}^*$.

We now rank matrices from the prior task and reuse the resulting freezing set on the target task. For a budget b, let $\mathcal{F}_{\text{prior}}^{\text{TDA-H}}(b)$ be the b matrices with the largest values of D_i, and let $\mathcal{F}_{\text{prior}}^{\text{TDA-L}}(b)$ be the b matrices with the smallest values. The magnitude baseline uses the same ranking procedure with M_i. The selected matrices are frozen from the start of fine-tuning on $\mathcal{T}^*$.

Profile. Each target task starts from the original pretrained model and reuses only the matrix indices (*e.g.,* K:{1,3,5}; V: {2,8,9}) selected from the source task with a budget. We call this set of indices the freezing profile. **Profile reuse does not transfer weight matrices, task-specific updates, or persistence diagrams; it reuses only the matrix identities to be frozen during target-task fine-tuning**. The theorem below gives a sufficient condition under which the prior-task ranking is preserved on $\mathcal{T}^*$, making the freezing profile reusable.

Theorem 5.1 (Stability of reusable freezing profiles)**.** *Fix a pretrained transformer $f_{\theta^{(0)}}$ and a monitored matrix set $\mathcal{I}$. Let $\mathcal{T}$ be a prior task and let $\mathcal{T}^*$ be a target task. For each task, let $\theta_{\mathcal{T}}^{(t)} = \{W_{i,\mathcal{T}}^{(t)}\}_{i=1}^m$ and $\theta_{\mathcal{T}^*}^{(t)} = \{W_{i,\mathcal{T}^*}^{(t)}\}_{i=1}^m$ denote the fine-tuned parameters after t optimization steps, both initialized at $\theta^{(0)}$. Let $L_{\mathcal{T}}(\theta)$ and $L_{\mathcal{T}^*}(\theta)$ be the corresponding population fine-tuning objectives, and write $G_{\mathcal{T}}(\theta) = \nabla_\theta L_{\mathcal{T}}(\theta)$ and $G_{\mathcal{T}^*}(\theta) = \nabla_\theta L_{\mathcal{T}^*}(\theta)$. Assume both trajectories remain in a ball $B(\theta^{(0)}, R)$, both gradient fields are L-Lipschitz on this ball, and $\sup_{\theta\in B(\theta^{(0)},R)} \|G_{\mathcal{T}}(\theta) - G_{\mathcal{T}^*}(\theta)\|_2 \leq \epsilon$. If both tasks are fine-tuned with the same gradient descent update and step size α, then for every $t \leq t_{\max}$ we have $\|\theta_{\mathcal{T}}^{(t)} - \theta_{\mathcal{T}^*}^{(t)}\|_2 \leq R_t$, where $R_t := \alpha\epsilon\sum_{r=0}^{t-1}(1+\alpha L)^r$.*

Assume that the same row-cloud dissimilarity δ_i is used across tasks and C_δ-Lipschitz under row perturbations. For each $i \in \mathcal{I}$ and fixed homological dimension k, define $D_{i,\mathcal{T}}^{(t)} := \mathcal{W}ass_p(\mathrm{PD}_k(X_i^{(0)}), \mathrm{PD}_k(X_{i,\mathcal{T}}^{(t)}))$ and $D_{i,\mathcal{T}^*}^{(t)} := \mathcal{W}ass_p(\mathrm{PD}_k(X_i^{(0)}), \mathrm{PD}_k(X_{i,\mathcal{T}^*}^{(t)}))$. Then $|D_{i,\mathcal{T}}^{(t)} - D_{i,\mathcal{T}^*}^{(t)}| \leq 2C_\delta N_{i,k}^{1/p} R_t$, where $N_{i,k}$ is the maximum number of off-diagonal points in the two PDs for matrix i in dimension k.

Take $k = 0$ and define the prior and target ranking scores by $S_i^{\text{prior}} := D_{i,\mathcal{T}}^{(t_{\max})}$ and $S_i^* := D_{i,\mathcal{T}^*}^{(t_{\max})}$. Let $S_{(1)}^{\text{prior}} \geq \cdots \geq S_{(|\mathcal{I}|)}^{\text{prior}}$ be the sorted prior scores, and let $g_H(b) := S_{(b)}^{\text{prior}} - S_{(b+1)}^{\text{prior}}$ be the high-drift margin for freezing budget b. Define $\mathcal{F}_*^{\text{TDA-H}}(b)$ and $\mathcal{F}_*^{\text{TDA-L}}(b)$ analogously from the target scores S_i^*. Let $\delta_{\max} := 2C_\delta \max_{i\in\mathcal{I}} N_{i,0}^{1/p} R_{t_{\max}}$. If $g_H(b) > 2\delta_{\max}$, then $F_{\text{prior}}^{\text{TDA-H}}(b) =$

$F_*^{\mathrm{TDA-H}}(b)$. *The same statement holds for* $F_{\mathrm{prior}}^{\mathrm{TDA-L}}(b)$ *after sorting the scores in increasing order and using the corresponding low-drift margin.*

Theorem 5.1 applies to both high- and low-distance profiles and does not determine which end of the ranking should be frozen. We adopt High-3 as an empirical regularization rule that constrains the largest changes from the pretrained model while allowing the remaining trainable matrices to absorb adaptation, as illustrated in Figure 1. Appendix D provides the detailed motivation and analysis.

Scenario B: Fine-tuning with Dynamic Stopping

We now consider the second setting, where no prior fine-tuning profile is available. Let $\{\theta^{(t)}\}_{t=0}^{t_{\max}}$ denote the checkpoint sequence of a single full fine-tuning run. In this setting, our goal is to stop fine-tuning once the monitored matrices have stabilized topologically. Prior work offers evidence that "a few epochs of fine-tuning are typically sufficient to achieve strong performance" Sun & Raskutti (2026), with later epochs often providing limited additional gains.

Using the topological distance $D_i^{(t)}$ from Definition 4.1, we compute the normalized epoch-to-epoch drift $\Delta_i^{(t)}$ from Definition 4.2. For a monitored set of matrices $\mathcal{W} \subseteq \mathcal{I}$, we average this drift into a model-level stabilization score as follows.

Definition 5.2 (Topology-based stopping score)**.** Let $\mathcal{W} \subseteq \mathcal{I}$ be the monitored set of matrices. For $t \geq 2$, we define $r^{(t)} := \frac{1}{|\mathcal{W}|} \sum_{i \in \mathcal{W}} |\Delta_i^{(t)}|$. This score measures the average normalized change in distance from initialization across consecutive checkpoints.

Given a threshold $\eta \in (0, 1)$, we choose the stopping epoch $t^* := \min\{t \geq 2 : r^{(t)} \leq \eta\}$. If no such epoch is found within the training budget, we stop at the final epoch.

Computational complexity. TopoTuner is computationally efficient. For each monitored matrix $W_i \in \mathbb{R}^{d_i \times d_i'}$, computing pairwise row distances costs $O(d_i^2 d_i')$. Constructing the Vietoris–Rips filtration and computing persistence has worst-case exponential complexity in d_i, since the number of simplices can grow exponentially. In practice, we compute only low-dimensional persistence, *i.e.,* H_0 and limit the analysis to selected attention projection matrices. Across t checkpoints and monitored matrices $\mathcal{W}$, the post-hoc topological analysis costs $O\big(t \sum_{i \in \mathcal{W}} (d_i^2 d_i' + \mathrm{PH}(d_i))\big)$, where $\mathrm{PH}(d_i)$ denotes the persistence computation cost for the row cloud of matrix i.

6 Experiments

Hardware and implementation. All experiments were conducted on NVIDIA H200 GPUs and AMD EPYC 9555 64-Core Processor CPUs with our PyTorch-based code. We compute persistence diagrams with Ripser Tralie et al. (2018) and evaluate diagram distances using Gudhi's Wasserstein implementation Maria et al. (2014) with $p = 1$ (the infinite H_0 class contributes zero cost, rows are not normalized before cosine distance). Our implementation is available at https://anonymous.4open.science/r/TopoTuner/.

Datasets. We evaluate TopoTuner on a diverse set of downstream tasks spanning question answering, sentiment analysis, information retrieval, summarization, instruction following, and code generation. Table 1 summarizes the datasets, splits, and evaluation metrics. We use `QA` and `SA` datasets for fine-tuning and evaluation, and all twelve datasets for catastrophic forgetting after fine-tuning. For large datasets, training is capped at 20,000 examples, except for `SA:IMDB`, where we use 25,000 training examples.

Models and training procedure. We evaluate three open-source pretrained language models: `LLaMA-3.1-8B` (2024; LLaMA), `Mistral-7B-v0.3` (2023; Mistral), and `Qwen3-8B-Base` (2025; Qwen). For each model, we run both full fine-tuning and LoRA-based fine-tuning on each downstream task. We also tune the prominent models Spectrum Hartford et al. (2024) and DropBP Woo et al. (2024), as explained in Appendix E. We use cross-entropy loss and a fixed epoch budget of six for all runs to ensure consistency across models and tasks. Persistent homology uses the cosine distance and H_0. The experimental setup and hyperparameters are provided in the Appendix F.

Evaluation metrics. Our goal is to achieve the highest accuracy with the fewest parameter changes and the shortest fine-tuning time. We evaluate each method along three axes: downstream task performance, trainable-parameter efficiency, and training time. Downstream performance is measured on held-out evaluation sets using the task-specific metrics listed in Table 1. Efficiency is measured by i) the percentage of directly optimized parameters (Train.%) and ii) the percentage of pretrained parameters that actually change (Upd.%). Computational cost is measured by wall-clock training time in minutes on fixed hardware. Accuracy (Acc.) is computed on the full held-out evaluation split, while Std is estimated from three fixed, non-overlapping random subsets of the same evaluation split.

Table 1: Datasets and evaluation metrics used in our experiments. QA: question answering, SA: sentiment analysis, IR: information retrieval, SU: summarization, IF: instruction following, CG: code generation.

Task	Dataset	Train size	Eval size	Evaluation metric
QA	GSM8K Cobbe et al. (2021)	7,473	1,319	Accuracy
QA	MMLU Hendrycks et al. (2020)	99,842	14,042	Accuracy
SA	IMDB Maas et al. (2011)	25,000	25,000	Accuracy
SA	SST-2 Socher et al. (2013)	67,349	872	Accuracy
Forgetting				
IR	HotpotQA Yang et al. (2018)	90,447	7,405	EM and F1
IR	SQuAD v1.1 Rajpurkar et al. (2016)	87,599	10,570	EM and F1
SU	XSum Narayan et al. (2018)	204,045	11,334	ROUGE-1, ROUGE-2, ROUGE-L
SU	CNN/DailyMail See et al. (2017)	287,113	11,490	ROUGE-1, ROUGE-2, ROUGE-L
IF	Databricks-dolly-15k Conover et al. (2023)	15,011	15,011	ROUGE-1/2/L, BLEU, METEOR
IF	Alpaca Taori et al. (2023)	52,002	52,002	ROUGE-1/2/L, BLEU, METEOR
CG	HumanEval Chen et al. (2021)	164	164	Pass@1
CG	MBPP Austin et al. (2021)	257	257	Pass@1

6.1 Fine-tuning with Freezing Priors

We use the question-answering dataset QA:GSM8K as the prior because it provides a standard reasoning task with clean, exact-answer evaluation. However, as Figure 3 shows, for a fixed LLM, we find that the topological change patterns observed on QA:GSM8K remain consistent across the other datasets in our benchmark (topological changes of all models on datasets in Appendix G).

We compare full fine-tuning, LoRA Hu et al. (2022), Spectrum Hartford et al. (2024), DropBP Woo et al. (2024), the magnitude-based selector, and our TopoTuner and report our metrics.

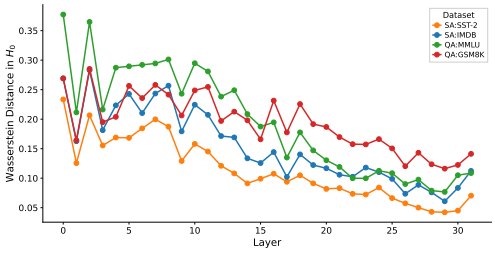


Figure 3: Epoch-6 Wasserstein H_0 distances for the V projection of LLaMA across datasets. Full Wasserstein distance profiles across models are reported in Appendix H.

We evaluate low-distance and high-distance freezing over budgets $b \in \{3, 6, 9, 12, 15\}$ per-projection-family. The low-distance strategy tests the hypothesis that stable matrices are less necessary for adaptation. The high-distance strategy tests the hypothesis that blocking the dominant adaptation path can constrain task-specific movement, reduce overfitting, and force the remaining trainable matrices to absorb the update. Neither end of the distance ranking dominates uniformly. We adopt High3 as a fixed empirical regularization rule, motivated by constraining the largest departures from the pretrained model; we report the low-distance results in Appendix I.

We freeze MLP weights and selectively freeze only the attention projections K, Q, V, O. This keeps the trainable budget comparable to PEFT methods, since MLP weights form most of the model parameters (*e.g.,* 70.2% for LLaMA). Our projection-level ranks in Appendix Table 11 show that the dominant adaptation route is model-dependent: LLaMA and Mistral concentrate changes in V/O, while Qwen concentrates the change in K/Q; therefore, the candidate set includes all four attention projections.

We run both *perfect-knowledge* and *profile-reuse* experiments. The perfect-knowledge setting selects frozen projections from the same QA:GSM8K full-fine-tuning profile used for evaluation; this setup is mainly for diagnostics. As shown in Appendix I, TDA-High3 outperforms LoRA by 2.43 points on LLaMA and 3.59 points on Mistral while changing only 2.64%–2.82% of model parameters.

In the second and main setting, Table 2 evaluates whether a freezing profile learned from QA:GSM8K can be reused on new datasets. We report TDA-High3 as TopoTuner because it is the smallest high-distance budget and requires no target-specific profile search. TopoTuner outperforms LoRA in 8 of 9 settings: on all three datasets for LLaMA and Mistral, and on SA:SST-2 and SA:IMDB for Qwen. It also reduces training time and changes only 0.56%–4.85% of pretrained parameters; other budgets are reported in Appendix Tables 8, 9, and 10. Among LoRA, DropBP, Spectrum, and

TOPOTUNER, TOPOTUNER achieves the highest accuracy in 6 of 9 settings, with average accuracy ranks of 1.00 on SST-2, 2.00 on IMDB (tied with Spectrum), and 1.33 on MMLU.

Table 2: **Profile reuse.** Fine-tuning results across `SA:SST-2`, `SA:IMDB`, and `QA:MMLU`. TOPOTUNER selective tuning freezes projections learned from `QA:GSM8K`. Avg. Rank: mean accuracy rank across models within each dataset (↓; best: **bold**, second: underlined). Upd.% denotes the fraction of pretrained parameters that actually change.

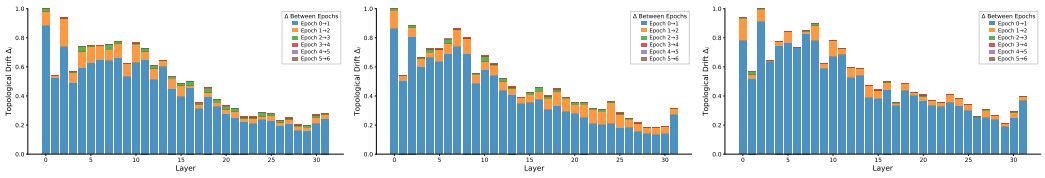
Dataset	Method	Qwen3-8B-Base				LLaMA-3.1-8B				Mistral-7B-v0.3				Avg.
		Train.%	Upd.%	Acc.±Std. (↑)	Time (min)	Train.%	Upd.%	Acc.±Std. (↑)	Time (min)	Train.%	Upd.%	Acc.±Std. (↑)	Time (min)	Rank ↓
IMDB	Full Fine-tuning	100	8.34	82.83 ± 0.31	195.1	100	24.95	90.66 ± 0.21	172.4	100	28.67	83.10 ± 0.18	164.8	-
	LoRA	0.19	18.38	82.89 ± 0.35	177.2	0.17	16.67	83.36 ± 0.30	153.2	0.19	18.49	82.72 ± 0.16	147.9	3.33
	DropBP	0.19	18.29	70.79 ± 0.30	214.6	0.17	16.63	**95.30 ± 0.20**	158.4	0.19	18.40	93.64 ± 0.22	164.2	2.67
	Spectrum	17.55	0.82	81.62 ± 0.28	172.0	15.80	2.56	94.98 ± 0.22	150.9	6.71	1.82	**94.82 ± 0.20**	143.6	**2.00**
	TOPOTUNER	8.45	0.56	**94.72 ± 0.08**	164.1	7.57	4.65	94.88 ± 0.05	143.8	8.39	4.85	94.30 ± 0.08	138.2	**2.00**
MMLU	Full Fine-tuning	100	9.39	73.22 ± 2.15	157.6	100	31.29	60.03 ± 0.05	139.6	100	35.38	61.85 ± 2.62	132.2	-
	LoRA	0.19	18.41	**74.13 ± 2.96**	141.9	0.17	16.67	61.86 ± 1.67	122.9	0.19	18.49	56.82 ± 2.85	118.4	1.67
	DropBP	0.19	18.19	37.54 ± 0.36	165.1	0.17	16.61	54.19 ± 0.91	122.3	0.19	18.42	50.01 ± 0.51	126.1	4.00
	Spectrum	17.55	0.66	66.88 ± 0.62	143.3	15.80	2.81	54.79 ± 0.94	126.2	6.71	2.00	52.31 ± 0.63	119.0	3.00
	TOPOTUNER	8.45	0.68	72.83 ± 1.97	131.7	7.57	2.49	**63.75 ± 1.79**	115.4	8.39	2.98	**60.68 ± 3.06**	110.3	**1.33**
SST-2	Full Fine-tuning	100	13.01	95.76 ± 1.00	32.7	100	21.88	96.10 ± 0.53	30.3	100	25.00	96.10 ± 0.87	32.9	-
	LoRA	0.19	11.02	95.75 ± 1.78	24.8	0.17	16.62	95.64 ± 2.22	21.5	0.19	18.41	96.10 ± 1.44	24.7	2.50
	DropBP	0.19	18.19	85.66 ± 1.12	33.9	0.17	16.61	95.53 ± 0.34	26.4	0.19	18.42	96.10 ± 0.21	34.1	3.50
	Spectrum	17.55	0.47	92.78 ± 0.68	30.2	15.80	1.24	95.76 ± 0.53	26.7	6.71	1.23	95.53 ± 0.92	30.8	3.00
	TOPOTUNER	8.45	1.15	**95.99 ± 0.41**	23.9	7.57	2.02	**95.98 ± 1.44**	21.2	8.39	2.44	**96.21 ± 0.70**	24.0	**1.00**

Across all nine settings, TOPOTUNER achieves the shortest fine-tuning time and changes substantially fewer pretrained parameters than Full FT, LoRA, and DropBP, while Spectrum achieves lower update fractions in some settings. The forgetting results in Appendix Tables 14, 15, and 16 show the same pattern: TOPOTUNER often preserves stronger out-of-domain performance on `QA`, `SA`, and `IR` evaluations, with more mixed effects on generation-style tasks.

Time efficiency. TOPOTUNER is the most time-efficient method, with a total end-to-end time (including profile construction) of 882.62 minutes across all nine runs of Table 2. This is 5.3% faster than the best competitor, LoRA, which requires 932.50 minutes, and 6.4% faster than the third-best method, Spectrum, which requires 942.70 minutes.

6.2 FINE-TUNING WITH DYNAMIC STOPPING

We next test whether topological drift can decide when fine-tuning should stop. For each fine-tuning epoch, we compute the normalized epoch-to-epoch topological drift of the monitored projection matrices and aggregate it into the stopping score from Definition 5.2. Training stops once the aggregated drift falls below a threshold, indicating topological stabilization.



`QA:MMLU` `SA:SST-2` `SA:IMDB`

Figure 4: Epoch-to-epoch topological drift bars for the V projections of LLaMA under H_0 during full fine-tuning. Most layers stabilize by epoch 2, with subsequent epochs showing only small residual changes. See all datasets and models in Appendix G.

Figure 4 shows the epoch-to-epoch topological drift for LLaMA on `QA:MMLU`, `SA:SST-2`, and `SA:IMDB`. Appendix G reports the complete drift bars together with the epoch-wise Wasserstein distance curves for all models and datasets. The drift values show that fine-tuning on `SA:SST-2` and `SA:IMDB` can stop after the second epoch. The accuracy curves in Appendix Figure 14 yield the same conclusion regarding performance: accuracy rises early and then remains within a narrow range while topological drift has already decayed. Training beyond this point mostly involves small residual updates.

Figure 5 shows how the selected stopping epoch changes with η on SA:SST-2. The rule first stops at epoch 3 for LLaMA and Qwen at $\eta = 0.05$, and reaches epoch-2 stopping for Qwen at $\eta = 0.15$. Appendix N provides the stopping-epoch curves for SA:IMDB and QA:MMLU across the three model families.

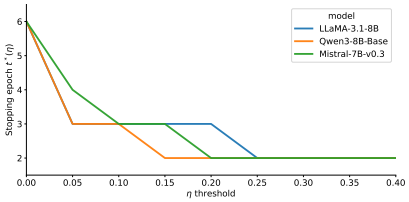


Figure 5: Topology-based stopping epochs under full fine-tuning. The plots show selected stopping epochs across η thresholds for SA:SST-2 across the three model families.

6.3 Profiles for Efficient Fine-Tuning

Profile stability and the role of topology. We directly measure whether the GSM8K-derived rankings remain stable on SST-2, IMDB, and MMLU using the same projection families and ranking procedure used to construct the operational High3 profiles. Across 18 model–projection–target comparisons (3 models $\times$ 2 projection families per model $\times$ 3 target datasets), at least two of the three source-selected matrices are retained (in top-3) in 12 cases, with median full-ranking Spearman correlations of 0.44, 0.92, and 0.93 for Qwen, LLaMA, and Mistral, respectively (0.89 overall). These results provide empirical support for the ranking-stability behavior characterized by Theorem 5.1. We further compare H_0-Wasserstein with spectral change and pairwise row-distance change using the same matrices and ranking procedure. Table 3 shows that H_0 produces the most stable rankings across all four measures; the alternative High3 profiles overlap with the H_0 profiles by only 0.51–0.56 on average.

Prior-task selection. We use QA:GSM8K as a source task because it is a standard reasoning dataset with clean exact-answer evaluation and is the smallest dataset in our profile-learning experiments. However, the cross-task analysis in Section 6.3 shows that, for a fixed pretrained model, similar high-distance matrix identities recur across SST-2, IMDB, and MMLU. Thus, we emphasize that GSM8K is not a privileged source task; it only serves as a convenient dataset from which the model-specific freezing profile is learned.

Table 3: Cross-task stability of alternative matrix-ranking summaries over 18 comparisons. O@3: source Top-3 retained in target Top-3; Exact: identical Top-3 sets; ρ: median Spearman correlation; Disp.: mean absolute rank displacement.

Statistic	O@3 ↑	Exact ↑	ρ ↑	Disp. ↓
H_0-Wasserstein	**0.593**	**5/18**	**0.894**	**4.60**
Spectral change	0.519	1/18	0.628	6.88
Row-distance change	0.537	1/18	0.723	5.62

Efficient fine-tuning. We propose a reusable profile card that summarizes which attention projections consistently absorb topological change during a prior fine-tuning run. The card records the source task, ranking strategy, selected frozen projection IDs, stopping threshold, downstream tasks, parameter update rates, and forgetting behavior. Figure 7 shows an example profile learned from QA:GSM8K for LLaMA-3.1-8B using a cosine distance-Wasserstein H_0 High-3 strategy. The same selected matrix indices are reused across sentiment analysis, question answering, and summarization tasks without recomputing a freezing search, making the profile card a compact model-specific prior for subsequent fine-tuning runs.

Limitations. TopoTuner requires access to checkpoints and attention projection weights, so closed-weight API models cannot be studied without provider-side access to weights and freezing controls. Future work should test larger open-weight models, broader task families, and profile reuse across longer fine-tuning sequences.

7 Conclusion

We have introduced TopoTuner, a topology-guided approach that uses matrix-level structural change for selective freezing and early stopping. Across nine profile-reuse settings, TopoTuner outperforms LoRA in 8 of 9 settings, reduces training time by 16.5% relative to full fine-tuning and 5.3% relative to LoRA, and updates only 0.56%–4.85% of pretrained parameters. Our results suggest that parameter selection can be treated as a reusable property of a pretrained model, reducing the need to repeat matrix-selection search for every new dataset. This opens a direction toward model-specific adaptation profiles that characterize how a model tends to change during fine-tuning. Future work should test whether these profiles persist across longer sequences of tasks in continual learning.

AI USE STATEMENT

In this work, we did not use generative AI tools for research activities requiring disclosure, including the implementation of methods, formulation of mathematical claims, or interpretation of experimental results. Other required-disclosure tasks were not applicable to this work. We used generative AI tools for recommended-disclosure activities, including language refinement and LaTeX formatting. We take full responsibility for the final content of this work, including all scientific claims, experimental results, and research artifacts.

ETHICS STATEMENT

TOPOTUNER investigates topology-guided fine-tuning of large language models, with the goal of improving computational efficiency and providing greater control over model adaptation. The proposed method does not involve human subjects or the collection of personal or sensitive data. Our experiments focus on publicly available models and established benchmarks. We are committed to responsible research practices and reproducibility.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide the implementation of TOPOTUNER in an anonymous repository at `https://anonymous.4open.science/r/TopoTuner/`. The main paper describes the methodology and experimental protocol, while Appendix F provides the experimental setup and hyperparameters, and Appendix B details the algorithms for topological ranking and selective freezing. Additional experimental results and analyses are provided in the remaining appendix sections.

REFERENCES

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pp. 7319–7328. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.ACL-LONG.568. URL https://doi.org/10.18653/v1/2021.acl-long.568.

Jacob Austin, Augustus Odena, Maxwell I. Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with large language models. *CoRR*, abs/2108.07732, 2021. URL https://arxiv.org/abs/2108.07732.

Gunnar Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009. doi: 10.1090/S0273-0979-09-01249-X. URL https://doi.org/10.1090/S0273-0979-09-01249-X.

Frédéric Chazal and Bertrand Michel. An introduction to topological data analysis: Fundamental and practical aspects for data scientists. *Frontiers Artif. Intell.*, 4:667963, 2021. doi: 10.3389/FRAI.2021.667963. URL https://doi.org/10.3389/frai.2021.667963.

Frédéric Chazal, Vin de Silva, Marc Glisse, and Steve Y. Oudot. *The Structure and Stability of Persistence Modules*. Springer Briefs in Mathematics. Springer, 2016. ISBN 978-3-319-42543-6. doi: 10.1007/978-3-319-42545-0. URL https://doi.org/10.1007/978-3-319-42545-0.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol,

Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021. URL https://arxiv.org/abs/2107.03374.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. Free dolly: Introducing the world's first truly open instruction-tuned llm, 2023. URL https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm.

Eric Hartford, Lucas Atkins, Fernando Fernandes Neto, and David Golchinfar. Spectrum: Targeted training on signal to noise ratio. *CoRR*, abs/2406.06623, 2024. doi: 10.48550/ARXIV.2406.06623. URL https://doi.org/10.48550/arXiv.2406.06623.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *CoRR*, abs/2009.03300, 2020. URL https://arxiv.org/abs/2009.03300.

Felix Hensel, Michael Moor, and Bastian Rieck. A survey of topological machine learning methods. *Frontiers Artif. Intell.*, 4:681108, 2021. doi: 10.3389/FRAI.2021.681108. URL https://doi.org/10.3389/frai.2021.681108.

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. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Yi Hu, Shijia Kang, Haotong Yang, Haotian Xu, and Muhan Zhang. Beyond single-task: Robust multi-task length generalization for llms. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/3c7435bc2086a5fe2421dfd1dfc6e407-Abstract-Conference.html.

Wenke Huang, Jian Liang, Guancheng Wan, Didi Zhu, He Li, Jiawei Shao, Mang Ye, Bo Du, and Dacheng Tao. Be confident: Uncovering overfitting in MLLM multi-task tuning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/huang25r.html.

Zhenzhen Huang, Chaoning Zhang, Haoyu Bian, Songbo Zhang, Chi-Lok Andy Tai, Jiaquan Zhang, Caiyan Qin, Jingjing Qu, Yalan Ye, Yang Yang, and Heng Tao Shen. Optimizing soft prompt tuning via structural evolution. *CoRR*, abs/2602.16500, 2026. doi: 10.48550/ARXIV.2602.16500. URL https://doi.org/10.48550/arXiv.2602.16500.

Gabriel Ilharco, Marco Túlio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=6t0Kwf8-jrj.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *CoRR*, abs/2310.06825, 2023. doi: 10.48550/ARXIV.2310.06825. URL https://doi.org/10.48550/arXiv.2310.06825.

Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=UYneFzXSJWh.

Komal Kumar, Tajamul Ashraf, Omkar Thawakar, Rao Muhammad Anwer, Hisham Cholakkal, Mubarak Shah, Ming-Hsuan Yang, Phillip H. S. Torr, Salman H. Khan, and Fahad Shahbaz Khan. LLM post-training: A deep dive into reasoning large language models. *CoRR*, abs/2502.21321, 2025. doi: 10.48550/ARXIV.2502.21321. URL https://doi.org/10.48550/arXiv.2502.21321.

Yoonho Lee, Annie S. Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. Surgical fine-tuning improves adaptation to distribution shifts. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=APuPRxjHvZ.

Guozhi Liu, Weiwei Lin, Qi Mu, Tiansheng Huang, Ruichao Mo, Yuren Tao, and Li Shen. Targeted vaccine: Safety alignment for large language models against harmful fine-tuning via layer-wise perturbation. *IEEE Trans. Inf. Forensics Secur.*, 20:10806–10817, 2025. doi: 10.1109/TIFS.2025.3615412. URL https://doi.org/10.1109/TIFS.2025.3615412.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *CoRR*, abs/2308.08747, 2023. doi: 10.48550/ARXIV.2308.08747. URL https://doi.org/10.48550/arXiv.2308.08747.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1015.

Clément Maria, Jean-Daniel Boissonnat, Marc Glisse, and Mariette Yvinec. The gudhi library: Simplicial complexes and persistent homology. In Hoon Hong and Chee Yap (eds.), *Mathematical Software - ICMS 2014 - 4th International Congress, Seoul, South Korea, August 5-9, 2014. Proceedings*, volume 8592 of *Lecture Notes in Computer Science*, pp. 167–174. Springer, 2014. doi: 10.1007/978-3-662-44199-2_28. URL https://doi.org/10.1007/978-3-662-44199-2_28.

Gaurav Menghani. Efficient deep learning: A survey on making deep learning models smaller, faster, and better. *ACM Comput. Surv.*, 55(12):259:1–259:37, 2023. doi: 10.1145/3578938. URL https://doi.org/10.1145/3578938.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 1797–1807. Association for Computational Linguistics, 2018. doi: 10.18653/V1/D18-1206. URL https://doi.org/10.18653/v1/d18-1206.

David Pérez-Fernández, Asier Gutiérrez-Fandiño, Jordi Armengol-Estapé, and Marta Villegas. Characterizing and measuring the similarity of neural networks with persistent homology, 2021. URL https://arxiv.org/abs/2101.07752.

Emilie Purvine, Davis Brown, Brett A. Jefferson, Cliff A. Joslyn, Brenda Praggastis, Archit Rathore, Madelyn Shapiro, Bei Wang, and Youjia Zhou. Experimental observations of the topology of convolutional neural network activations. In Brian Williams, Yiling Chen, and Jennifer Neville (eds.), *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pp. 9470–9479. AAAI Press, 2023. doi: 10.1609/AAAI.V37I8.26134. URL https://doi.org/10.1609/aaai.v37i8.26134.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Jian Su, Kevin Duh, and Xavier Carreras (eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264.

Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1073–1083, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1099. URL https://aclanthology.org/P17-1099/.

Reece Shuttleworth, Jacob Andreas, Antonio Torralba, and Pratyusha Sharma. Lora vs full fine-tuning: An illusion of equivalence. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/ff541950d1e885af90f523571564a401-Abstract-Conference.html.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D13-1170.

Zexuan Sun and Garvesh Raskutti. A theoretical framework for LLM fine-tuning using early stopping for non-random initialization. *CoRR*, abs/2602.13942, 2026. doi: 10.48550/ARXIV.2602.13942. URL https://doi.org/10.48550/arXiv.2602.13942.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Llama Team. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024. doi: 10.48550/ARXIV.2407.21783. URL https://doi.org/10.48550/arXiv.2407.21783.

Qwen Team. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025. doi: 10.48550/ARXIV.2505.09388. URL https://doi.org/10.48550/arXiv.2505.09388.

Junjiao Tian, Chengyue Huang, and Zsolt Kira. Rethinking weight decay for robust fine-tuning of foundation models. In Amir Globerson, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/281a6b49c66b94f81f3b1ae8f1a0dea4-Abstract-Conference.html.

Christopher J. Tralie, Nathaniel Saul, and Rann Bar-On. Ripser.py: A lean persistent homology library for python. *J. Open Source Softw.*, 3(29):925, 2018. doi: 10.21105/JOSS.00925. URL https://doi.org/10.21105/joss.00925.

Sunghyeon Woo, Baeseong Park, Byeongwook Kim, Minjung Jo, Se Jung Kwon, Dongsuk Jeon, and Dongsoo Lee. Dropbp: Accelerating fine-tuning of large language models by dropping backward propagation. In Amir Globerson, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/240225294cdd2c9b692c2519d3278a08-Abstract-Conference.html.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 23965–23998. PMLR, 2022. URL https://proceedings.mlr.press/v162/wortsman22a.html.

Lingling Xu, Haoran Xie, S. Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *IEEE Trans. Pattern Anal. Mach. Intell.*, 48(6):6107–6126, 2026. doi: 10.1109/TPAMI.2026.3657354. URL https://doi.org/10.1109/TPAMI.2026.3657354.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 2369–2380. Association for Computational Linguistics, 2018. doi: 10.18653/V1/D18-1259. URL https://doi.org/10.18653/v1/d18-1259.

Can Yaras, Peng Wang, Laura Balzano, and Qing Qu. Compressible dynamics in deep overparameterized low-rank learning & adaptation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 56946–56965. PMLR, 2024. URL https://proceedings.mlr.press/v235/yaras24a.html.

Murong Yue. A survey of large language model agents for question answering. *CoRR*, abs/2503.19213, 2025. doi: 10.48550/ARXIV.2503.19213. URL https://doi.org/10.48550/arXiv.2503.19213.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pp. 1–9. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.ACL-SHORT.1. URL https://doi.org/10.18653/v1/2022.acl-short.1.

Junhao Zheng, Xidi Cai, Shengjie Qiu, and Qianli Ma. Spurious forgetting in continual learning of language models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=ScI7IlKGdI.

APPENDIX

A PROOFS

Stability motivation. The usefulness of Δ_i rests on the standard stability principle of persistent homology: small perturbations in the underlying metric induce small perturbations in the resulting diagrams. In our setting, this means that if fine-tuning changes pairwise distances between rows only slightly, then the corresponding topological drift must also remain small. The next proposition makes this statement explicit.

Proposition A.1 (Stability of topological sensitivity)**.** *Let $X = \{x_1, \dots, x_d\}$ and $\tilde{X} = \{\tilde{x}_1, \dots, \tilde{x}_d\}$ be the row clouds of W_i and $\tilde{W}_i$, indexed consistently. Let d and $\tilde{d}$ be the metrics used to construct the Vietoris–Rips filtrations of X and $\tilde{X}$. If $\|d - \tilde{d}\|_\infty := \max_{a,b} |d(a,b) - \tilde{d}(a,b)| \le \eta$, then for every $k \ge 0$ we have $d_B(PD_k(X), PD_k(\tilde{X})) \le \eta$, and therefore $W_p(PD_k(X), PD_k(\tilde{X})) \le N_k^{1/p}\eta$, where d_B is the bottleneck distance and N_k is the maximum number of off-diagonal points in the two diagrams.*

Proof. For any simplex $S \subseteq \{1, \dots, d\}$, let $\phi(S) = \max_{a,b\in S} d(a,b)$ and $\tilde{\phi}(S) = \max_{a,b\in S} \tilde{d}(a,b)$. The assumption $\|d - \tilde{d}\|_\infty \le \eta$ implies $|\phi(S) - \tilde{\phi}(S)| \le \eta$ for every simplex S. Hence $\mathrm{VR}_\varepsilon(X) \subseteq \mathrm{VR}_{\varepsilon+\eta}(\tilde{X})$ and $\mathrm{VR}_\varepsilon(\tilde{X}) \subseteq \mathrm{VR}_{\varepsilon+\eta}(X)$ for all $\varepsilon \ge 0$, so the two filtrations are η-interleaved. By the stability theorem for persistence diagrams under interleavings Chazal et al. (2016), it follows that $d_B(PD_k(X), PD_k(\tilde{X})) \le \eta$. Finally, a bottleneck matching with cost d_B moves at most N_k off-diagonal points by at most d_B in $\|\cdot\|_\infty$, so the total p-cost is at most $N_k d_B^p$. Taking the pth root yields $W_p(PD_k(X), PD_k(\tilde{X})) \le N_k^{1/p} d_B \le N_k^{1/p}\eta$. □

We use this stability result in the next section to justify when a prior-task topological profile remains stable under reuse to a new task.

We consider both highest-distance and lowest-distance selections. For a budget b, we define $\mathcal{F}_{\text{prior}}^{\text{TDA-H}}(b)$ as the b matrices with the largest values of S_i, and $\mathcal{F}_{\text{prior}}^{\text{TDA-L}}(b)$ as the b matrices with the smallest values. In the magnitude-based baseline, we analogously rank matrices using M_i. These selected matrices are frozen from the start of fine-tuning on the target task $\mathcal{T}^*$. This setting tests whether structural distance observed in one prior dataset can serve as a reusable fine-tuning profile for selective adaptation to other datasets, and whether topology-based rankings provide a different freezing signal from simple relative weight changes.

Theorem A.2 (Stability of reusable freezing profiles)**.** *Fix a pretrained transformer $f_{\theta^{(0)}}$ and a monitored matrix set $\mathcal{I}$. Let $\mathcal{T}$ be a prior task and let $\mathcal{T}^*$ be a target task. For each task, let $\theta_{\mathcal{T}}^{(t)} = \{W_{i,\mathcal{T}}^{(t)}\}_{i=1}^m$ and $\theta_{\mathcal{T}^*}^{(t)} = \{W_{i,\mathcal{T}^*}^{(t)}\}_{i=1}^m$ denote the fine-tuned parameters after t optimization steps, both initialized at $\theta^{(0)}$. Let $L_{\mathcal{T}}(\theta)$ and $L_{\mathcal{T}^*}(\theta)$ be the corresponding population fine-tuning objectives, and write $G_{\mathcal{T}}(\theta) = \nabla_\theta L_{\mathcal{T}}(\theta)$ and $G_{\mathcal{T}^*}(\theta) = \nabla_\theta L_{\mathcal{T}^*}(\theta)$. Assume both trajectories remain in a ball $B(\theta^{(0)}, R)$, both gradient fields are L-Lipschitz on this ball, and $\sup_{\theta\in B(\theta^{(0)},R)} \|G_{\mathcal{T}}(\theta) - G_{\mathcal{T}^*}(\theta)\|_2 \le \epsilon$. If both tasks are fine-tuned with the same gradient descent update and step size α, then for every $t \le t_{\max}$ we have $\|\theta_{\mathcal{T}}^{(t)} - \theta_{\mathcal{T}^*}^{(t)}\|_2 \le R_t$, where $R_t := \alpha\epsilon \sum_{r=0}^{t-1}(1 + \alpha L)^r$.*

Assume that the same row-cloud dissimilarity δ_i is used across tasks and C_δ-Lipschitz under row perturbations. For each $i \in \mathcal{I}$ and fixed homological dimension k, define $D_{i,\mathcal{T}}^{(t)} := \mathcal{W}ass_p(\mathrm{PD}_k(X_i^{(0)}), \mathrm{PD}_k(X_{i,\mathcal{T}}^{(t)}))$ and $D_{i,\mathcal{T}^*}^{(t)} := \mathcal{W}ass_p(\mathrm{PD}_k(X_i^{(0)}), \mathrm{PD}_k(X_{i,\mathcal{T}^*}^{(t)}))$. Then $|D_{i,\mathcal{T}}^{(t)} - D_{i,\mathcal{T}^*}^{(t)}| \le 2C_\delta N_{i,k}^{1/p} R_t$, where $N_{i,k}$ is the maximum number of off-diagonal points in the two PDs for matrix i in dimension k.

Take $k = 0$ and define the prior and target ranking scores by $S_i^{\text{prior}} := D_{i,\mathcal{T}}^{(t_{\max})}$ and $S_i^* := D_{i,\mathcal{T}^*}^{(t_{\max})}$. Let $S_{(1)}^{\text{prior}} \ge \cdots \ge S_{(|\mathcal{I}|)}^{\text{prior}}$ be the sorted prior scores, and let $g_H(b) := S_{(b)}^{\text{prior}} - S_{(b+1)}^{\text{prior}}$ be the high-drift margin for freezing budget b. Define $\mathcal{F}_*^{\text{TDA-H}}(b)$ and $\mathcal{F}_*^{\text{TDA-L}}(b)$ analogously from the target scores S_i^*. Let $\delta_{\max} := 2C_\delta \max_{i\in\mathcal{I}} N_{i,0}^{1/p} R_{t_{\max}}$. If $g_H(b) > 2\delta_{\max}$, then $F_{\text{prior}}^{\text{TDA-H}}(b) =$

$F_*^{\mathrm{TDA-H}}(b)$. *The same statement holds for* $F_{\mathrm{prior}}^{\mathrm{TDA-L}}(b)$ *after sorting the scores in increasing order and using the corresponding low-drift margin.*

Proof. Let $e_t := \|\theta_{\mathcal{T}}^{(t)} - \theta_{\mathcal{T}^*}^{(t)}\|_2$. Since both runs start from the same pretrained model, $e_0 = 0$.

Theorem 5.1 gives an idealized sufficient condition under gradient descent. Our experiments use AdamW Loshchilov & Hutter (2019), so we use the result to motivate ranking stability, not as a guarantee for the experimental optimizer. The two update equations are $\theta_{\mathcal{T}}^{(t+1)} = \theta_{\mathcal{T}}^{(t)} - \alpha G_{\mathcal{T}}(\theta_{\mathcal{T}}^{(t)})$ and $\theta_{\mathcal{T}^*}^{(t+1)} = \theta_{\mathcal{T}^*}^{(t)} - \alpha G_{\mathcal{T}^*}(\theta_{\mathcal{T}^*}^{(t)})$. Therefore $e_{t+1} \le \|\theta_{\mathcal{T}}^{(t)} - \theta_{\mathcal{T}^*}^{(t)} - \alpha(G_{\mathcal{T}}(\theta_{\mathcal{T}}^{(t)}) - G_{\mathcal{T}}(\theta_{\mathcal{T}^*}^{(t)}))\|_2 + \alpha\|G_{\mathcal{T}}(\theta_{\mathcal{T}^*}^{(t)}) - G_{\mathcal{T}^*}(\theta_{\mathcal{T}^*}^{(t)})\|_2$. By Lipschitzness and task-gradient closeness, $e_{t+1} \le (1+\alpha L)e_t + \alpha\epsilon$. Iterating this recurrence gives $e_t \le \alpha\epsilon \sum_{r=0}^{t-1}(1+\alpha L)^r = R_t$.

For a monitored matrix W_i, the block difference is bounded by the full parameter difference, so $\|W_{i,\mathcal{T}}^{(t)} - W_{i,\mathcal{T}^*}^{(t)}\|_F \le R_t$. Hence every row satisfies $\|x_{a,\mathcal{T}}^{(i,t)} - x_{a,\mathcal{T}^*}^{(i,t)}\|_2 \le R_t$. Since the chosen row-cloud metric is fixed across tasks and stable under row perturbations, we have $\|\delta_{i,\mathcal{T}}^{(t)} - \delta_{i,\mathcal{T}^*}^{(t)}\|_\infty \le 2C_\delta R_t$.

Applying Proposition 4.3 with $\eta = 2C_\delta R_t$ gives $\mathcal{W}ass_p(\mathrm{PD}_k(X_{i,\mathcal{T}}^{(t)}), \mathrm{PD}_k(X_{i,\mathcal{T}^*}^{(t)})) \le 2C_\delta N_{i,k}^{1/p} R_t$. Since both distances $D_{i,\mathcal{T}}^{(t)}$ and $D_{i,\mathcal{T}^*}^{(t)}$ are measured from the same pretrained diagram $\mathrm{PD}_k(X_i^{(0)})$, the triangle inequality for $\mathcal{W}ass_p$ gives $|D_{i,\mathcal{T}}^{(t)} - D_{i,\mathcal{T}^*}^{(t)}| \le \mathcal{W}ass_p(\mathrm{PD}_k(X_{i,\mathcal{T}}^{(t)}), \mathrm{PD}_k(X_{i,\mathcal{T}^*}^{(t)})) \le 2C_\delta N_{i,k}^{1/p} R_t$.

Set $\delta_{\max} := 2C_\delta \max_{i\in\mathcal{I}} N_{i,0}^{1/p} R_{t_{\max}}$. Then every Scenario A ranking score changes by at most $\delta_{\max}$ between the prior task and the target task. If the high-distance gap $g_H(b)$ is larger than $2\delta_{\max}$, no matrix outside the prior top-b set can pass a matrix inside that set after moving all scores by at most $\delta_{\max}$. Hence $F_{\mathrm{prior}}^{\mathrm{TDA-H}}(b) = F_*^{\mathrm{TDA-H}}(b)$. The low-drift claim is identical after sorting in increasing order. □

Theorem 5.1 gives a sufficient condition for reusable freezing profiles while avoiding the claim that all datasets produce identical fine-tuning dynamics. It states that, for a fixed pretrained model $f_{\theta^{(0)}}$, the profile learned on a prior task remains valid on a target task when the two tasks induce close gradient fields in the local fine-tuning region and the prior ranking has a nonzero margin. The fixed model matters because the local Jacobians, attention geometry, and row-cloud geometry are inherited from $\theta^{(0)}$. The dataset changes the coefficients of the update, but under the gradient-field condition it cannot move the monitored matrices far enough to change their topological ranking. Proposition 4.3 then converts this parameter-level closeness into persistence-diagram closeness, and the margin condition converts persistence-diagram closeness into equality of the selected freezing profile.

Connection to existing evidence. Our theorem is consistent with prior evidence that fine-tuning remains constrained near a pretrained initialization. Intrinsic-dimension results show that language-model adaptation can often be described by a low-dimensional update subspace Aghajanyan et al. (2021). LoRA shows that strong adaptation can be achieved through low-rank updates to pretrained weight matrices Hu et al. (2022), while BitFit shows that changing only bias parameters can be competitive in several language-understanding settings Zaken et al. (2022). These results support the premise that downstream adaptation does not use the full parameter space uniformly. Task arithmetic and model soups give a more direct precedent for reusing fine-tuning changes from a shared initialization: task vectors behave as reusable weight-space objects, and independently fine-tuned models from the same initialization can often be averaged successfully Ilharco et al. (2023); Wortsman et al. (2022). The topological part of our argument relies on the standard stability theorem for persistence diagrams Chazal et al. (2016), and previous neural-network studies have already used persistent homology to compare trained models and layer representations Pérez-Fernández et al. (2021); Purvine et al. (2023). In this setting, TopoTuner adds a matrix-level conclusion: if two tasks remain in the same local adaptation regime of $f_{\theta^{(0)}}$, then their Wasserstein distance scores remain close, and the same high-distance or low-distance freezing profile can be reused.

B FREEZING PSEUDOCODE

Algorithms 1–3 summarize the full TOPOTUNER selection pipeline. Algorithm 1 computes Wasserstein distances for all $K/Q/V/O$ projection matrices and identifies the projection families with the largest aggregate change. Algorithm 2 selects the High-B or Low-B layers within these two families, and Algorithm 3 applies the resulting frozen sets during selective fine-tuning.

Algorithm 1 Computing TDA-Based Wasserstein Layer Orderings

Require: Pretrained model $f_{\theta^{(0)}}$; fully fine-tuned model $f_{\theta^{(T)}}$; projection set $\mathcal{P} = \{K, Q, V, O\}$; Wasserstein order p
Ensure: Ordered layer lists $\pi_K, \pi_Q, \pi_V, \pi_O$ and selected projection families $\mathcal{P}^\star$

for $P \in \mathcal{P}$ **do**
  for each transformer layer ℓ **do**
    Extract pretrained matrix $W_{\ell,P}^{(0)}$ and fine-tuned matrix $W_{\ell,P}^{(T)}$
    $X_{\ell,P}^{(0)} \leftarrow \mathrm{rows}(W_{\ell,P}^{(0)}), \quad X_{\ell,P}^{(T)} \leftarrow \mathrm{rows}(W_{\ell,P}^{(T)})$
    $D_{\ell,P}^{(0)} \leftarrow \mathrm{PD}(X_{\ell,P}^{(0)}), \quad D_{\ell,P}^{(T)} \leftarrow \mathrm{PD}(X_{\ell,P}^{(T)})$
    $s_{\ell,P} \leftarrow W_p\left(D_{\ell,P}^{(0)}, D_{\ell,P}^{(T)}\right)$ ▷ Wasserstein distance between persistence diagrams
  end for
  $\pi_P \leftarrow \mathrm{argsort}_\ell(s_{\ell,P})$ ▷ low to high Wasserstein score
end for
for $P \in \mathcal{P}$ **do**
  $S_P \leftarrow \frac{1}{L}\sum_{\ell=1}^{L} s_{\ell,P}$
end for
$\mathcal{P}^\star \leftarrow \mathrm{Top2}_{P\in\mathcal{P}}\, S_P$
return $\pi_K, \pi_Q, \pi_V, \pi_O, \mathcal{P}^\star$

Algorithm 2 Constructing Frozen Layer Sets from Wasserstein Layer Orderings

Require: Ordered layer lists $\{\pi_P\}_{P\in\{K,Q,V,O\}}$ sorted from low to high Wasserstein score; selected projection families $\mathcal{P}^\star$; number of layers L; per-family freezing budget B; regime $r \in \{\textsc{Low}, \textsc{High}\}$
Ensure: Frozen layer sets $\{\mathcal{F}_P\}_{P\in\mathcal{P}^\star}$

for $P \in \mathcal{P}^\star$ **do**
  if $r = \textsc{Low}$ **then**
    $\mathcal{F}_P \leftarrow \{\pi_P[0], \ldots, \pi_P[B-1]\}$
  else if $r = \textsc{High}$ **then**
    $\mathcal{F}_P \leftarrow \{\pi_P[L-B], \ldots, \pi_P[L-1]\}$
  end if
end for
return $\{\mathcal{F}_P\}_{P\in\mathcal{P}^\star}$

Algorithm 3 Selective $K/Q/V/O$ Tuning from Frozen Layer Sets

Require: Model f_θ; selected projection families $\mathcal{P}^\star$; frozen layer sets $\{\mathcal{F}_P\}_{P\in\mathcal{P}^\star}$
Ensure: Model with unfrozen matrices from the selected projection families trainable

for all parameters $p \in \theta$ **do**
    p.requires_grad $\leftarrow$ False
end for
for all named parameters (n, p) in f_θ **do**
    Parse transformer layer index ℓ from parameter name n
    Parse projection family P and transformer layer index ℓ from parameter name n
    if $P \in \mathcal{P}^\star$ and $\ell \notin \mathcal{F}_P$ **then**
        p.requires_grad $\leftarrow$ True
    end if
end for
Keep all parameters outside the selected projection families frozen
return f_θ

C Reusable Freezing Profiles

The following two figures summarize the reusable freezing profiles produced by TopoTuner. Figure 6 defines the profile fields, while Figure 7 reports the three profiles used in our experiments. Each profile is learned from a prior fine-tuning task, ranks attention projection matrices by prior-task topological distance, and freezes the selected matrices during target-task fine-tuning.

Profile Definition

Profile ID	Identifier of the reusable freezing profile, combining the freezing strategy and model.
Model	Pretrained model f_θ for which the freezing profile is learned.
Prior task	Prior task $\mathcal{T}$ and prior dataset $\mathcal{D}_\mathcal{T}$ used to compute the matrix-level ranking scores.
Freezing strategy	Ranking score and selection rule used to choose frozen matrices from the candidate set, e.g., Cosine-Wasserstein for ranking score and High-3 for selection rule.
Frozen IDs	Selected matrices or layers frozen from the start of target-task fine-tuning.
Reused in	Acc±Std, ETA (with GPU Used), Upd.%, and Δ_f, denoting accuracy with standard deviation, total training time, updated-parameter percentage, and forgetting datasets. ✓denotes our profile card is forgetting less than full fine-tuning and ✗denotes the opposite.
Stopping η	Topology-based stopping threshold. Fine-tuning stops at the first epoch where the normalized epoch-to-epoch topological drift falls below η.

Figure 6: **Profile Definition.** We define a reusable freezing profile learned from a prior-task fine-tuning run. Each profile records the pretrained model, prior task, freezing strategy, selected frozen IDs, target-task reuse outcomes, and topology-based stopping threshold.

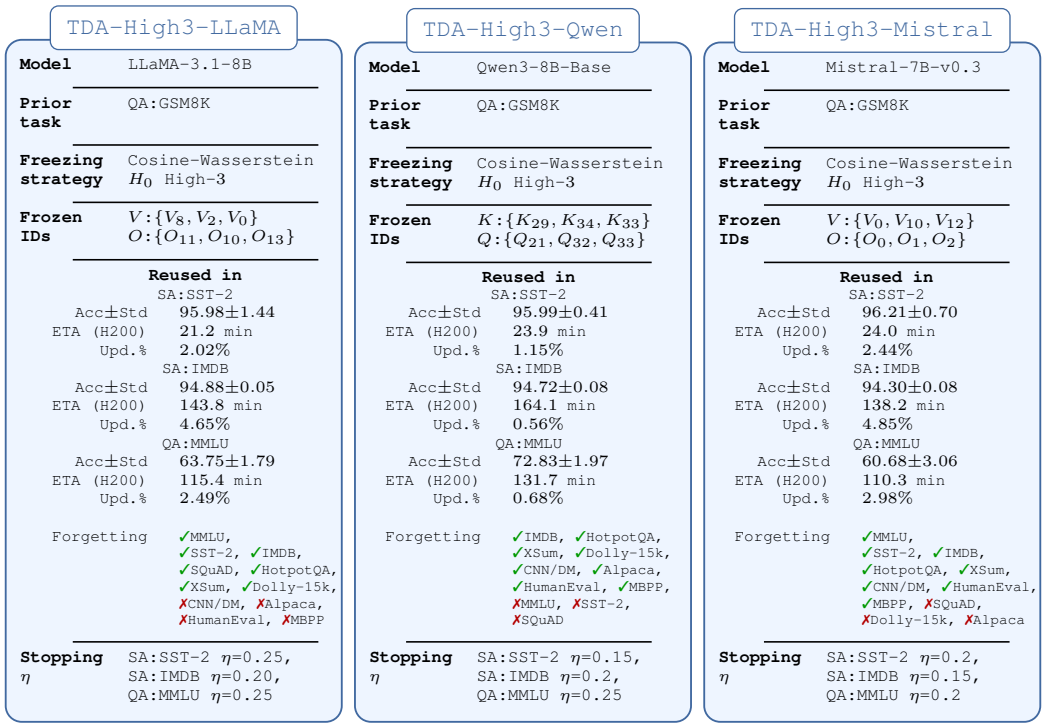


Figure 7: **Reusable Freezing Profiles.** Each model card summarizes the model, prior task, candidate matrix set, freezing strategy, selected frozen IDs, and outcome on target datasets. We report three profiles used in our experiments: TDA-High3-LLaMA, TDA-High3-Qwen, and TDA-High3-Mistral.

D HIGH- AND LOW- DISTANCE FREEZING

The distance ranking can be used in two freezing strategies: low and high. Low-distance freezing removes matrices that undergo little structural change during source-task fine-tuning. If a matrix changes little under unconstrained adaptation, freezing it may impose only a weak restriction on the optimization. High-distance freezing follows a different principle: matrices with the largest distance undergo the strongest change from their pretrained model, so freezing them explicitly constrains task-specific movement away from the pretrained model. Theorem 5.1 applies to both cases and does not establish that either end of the ranking is preferable.

At a fixed freezing budget, the two strategies have essentially the same trainable-parameter and runtime cost; the distinction is therefore which adaptation directions are constrained. Figure 1 shows that when high-distance matrices are frozen, the remaining trainable matrices absorb larger updates, indicating that optimization can redistribute adaptation under the constraint. This behavior motivates us to view high-drift freezing as structural regularization: the model retains substantial adaptation capacity while its largest source-task departures are not allowed. Our source-task experiments also show that low-drift freezing can be preferable in some settings, so High3 is an empirical design choice rather than a universal optimum.

Our choice was influenced by prior evidence that unrestricted fine-tuning can distort pretrained features and reduce out-of-distribution performance Kumar et al. (2022), that restricting fine-tuning to selected parts of a pretrained network can improve generalization under distribution shift Lee et al. (2023), and that selectively constraining departures from pretrained initialization can improve robustness and generalization Tian et al. (2024). In our overfitting analysis, TDA-High3 also yields a smaller train–test gap than TDA-Low3 in the analyzed LLaMA setting (Section P). We therefore use High3 as a fixed regularization rule in the profile-reuse experiments, while reporting both ends of the ranking in the ablations.

E SPECTRUM AND DROPBP

Spectrum. Spectrum Hartford et al. (2024) is a parameter-selection baseline that uses signal-to-noise ratio (SNR) statistics to decide which parameters should remain trainable. In our implementation, we first tested several top-$N\%$ SNR configurations, including $N \in \{50, 10, 5, 3\}$, to understand how the selected trainable budget changes with the masking ratio. We report the $N = 5$ setting because its updatable parameter fraction is closest to TOPOTUNER, making the comparison more controlled in terms of effective parameter updates.

DropBP. DropBP Woo et al. (2024) is a compute- and memory-oriented training baseline that reduces backward computation during fine-tuning. Since DropBP can be combined with either full fine-tuning or LoRA, we considered both DropBP+Full and DropBP+LoRA as possible baselines. We use DropBP+LoRA in the main comparison because its updatable parameter fraction is closer to TOPOTUNER, making the LoRA-based DropBP setting the fairer comparison.

Experimental protocol. For both baselines, we keep the experimental setup aligned with the rest of our profile-reuse experiments. Spectrum is run with selective full-parameter fine-tuning over the SNR-selected unfrozen weights, while DropBP is run with LoRA adapters on the attention projections. We use the same model-dataset grid, training data regime, and evaluation pipeline as our main `SA:SST-2`, `SA:IMDB`, and `QA:MMLU` experiments. The results are provided in Table 2. Experiment setups and hyperparameter details are provided in the Appendix F.

F EXPERIMENTAL SETUP AND HYPERPARAMETERS

Tables 4 and 5 summarize the training and evaluation setup used in our experiments. Table 4 reports task-level training settings, model-specific learning rates, and LoRA configurations, while Table 5 reports the decoding and checkpoint-scoring settings. For Full FT, Spectrum, and TopoTuner, we used $(2\times10^{-5}, 2\times10^{-5}, 5\times10^{-6})$, $(5\times10^{-6}, 5\times10^{-5}, 1\times10^{-5})$, and $(1\times10^{-5}, 2\times10^{-5}, 5\times10^{-6})$ learning rates for Qwen, LLaMA, and Mistral, respectively. For LoRA and DropBP, we used 2×10^{-4} and 3×10^{-4} learning rates. For each target dataset, the learning rate was selected using validation performance, and the corresponding test result was reported.

Table 4: Training hyperparameters used in our fine-tuning experiments. Freeze denotes the selective tuning setting used by the Eltwise- and TDA-based methods.

Setting	`QA:GSM8K`	`SA:SST-2`	`SA:IMDB`	`QA:MMLU`
Task-level training settings				
Epochs	6	6	6	6
Train subset	7,473	20,000	25,000	20,000
Batch size	16	64	32	32
Gradient accumulation	1	1	1	1
Max sequence length	512	1024	1024	512
Precision	bf16	bf16	bf16	bf16
Scheduler	cosine	cosine	cosine	cosine
Optimizer	AdamW	AdamW	AdamW	AdamW
Warmup Ratio	0.03	0.03	0.03	0.03
Max grad norm	1.0	1.0	1.0	1.0
Gradient checkpointing	enabled	enabled	enabled	enabled
Seed	42/123/456	42/123/456	42/123/456	42/123/456
Model-specific learning rates				
Qwen3-8B-Base full FT LR	2×10^{-5}	5×10^{-6}	1×10^{-5}	1×10^{-5}
LLaMA-3.1-8B full FT LR	2×10^{-5}	5×10^{-5}	2×10^{-5}	2×10^{-5}
Mistral-7B-v0.3 full FT LR	5×10^{-6}	1×10^{-5}	5×10^{-6}	5×10^{-6}
LoRA LR	2×10^{-4}	2×10^{-4}	2×10^{-4}	2×10^{-4}
Spectrum	same as full FT LR	same as full FT LR	same as full FT LR	same as full FT LR
DropBP	same as LoRA LR	same as LoRA LR	same as LoRA LR	same as LoRA LR
Freeze LR	same as full FT LR	same as full FT LR	same as full FT LR	same as full FT LR
LoRA configuration				
Rank / alpha / dropout	16/32/0.05	16/32/0.05	16/32/0.05	16/32/0.05
Target modules	q, k, v, o	q, k, v, o	q, k, v, o	q, k, v, o

Table 5: Evaluation settings used for `SA:SST-2`, `SA:IMDB`, `QA:MMLU` checkpoint scoring in the profile-reuse experiments.

Setting	`SA:SST-2`	`SA:IMDB`	`QA:MMLU`
Split	validation	test	validation
Decoding	greedy	greedy	greedy
Sampling	disabled	disabled	disabled
Max new tokens	4	4	2
Prompt max length	256	512	1024
Metric	Accuracy	Accuracy	Accuracy
Subset reporting	three fixed folds	three fixed folds	three fixed folds

F.1 Computational Cost of Topological Analysis

We measure offline persistence diagram construction and Wasserstein distance computation using 64 parallel CPU workers on an AMD EPYC 9555 server. Table 6 distinguishes the GSM8K analysis used to learn the reusable freezing profile from the dataset-specific analyses used for dynamic stopping. These are total offline analysis times, not per-epoch measurements.

Table 6: Offline TDA computational cost across models and datasets. GSM8K is used for freezing-profile learning, while SST-2, IMDB, and MMLU report dataset-specific analysis used for dynamic stopping. Times are in minutes; peak CPU memory was measured on GSM8K.

Model	Dataset	Use	Persistence (min)	Wasserstein (min)	Total (min)	Peak RAM (GB)
LLaMA	GSM8K	Profile	2.85	0.38	3.23	70.6
	SST-2	Stopping	2.77	0.39	3.16	–
	IMDB	Stopping	2.83	0.32	3.15	–
	MMLU	Stopping	2.88	0.32	3.20	–
Qwen	GSM8K	Profile	3.19	0.40	3.59	81.0
	SST-2	Stopping	3.16	0.39	3.55	–
	IMDB	Stopping	3.12	0.41	3.53	–
	MMLU	Stopping	3.17	0.36	3.53	–
Mistral	GSM8K	Profile	2.85	0.35	3.20	73.0
	SST-2	Stopping	2.79	0.34	3.13	–
	IMDB	Stopping	2.77	0.36	3.13	–
	MMLU	Stopping	2.83	0.38	3.21	–

The complete offline analysis takes approximately 3.1–3.6 minutes across the evaluated settings. For reusable freezing profiles, only the GSM8K profiling cost is incurred to construct the model-specific profile. The remaining measurements characterize the dataset-specific topological analysis used for dynamic stopping.

G Topological Drift Across Models and Datasets

We use the topological distance to initialization and topological drift defined in Definition 4.1 and Definition 4.2 to visualize how attention projection matrices change during fine-tuning. Figures 8-11 show this quantity for the V and O projections across datasets and models. The drift-bar plots summarize epoch-to-epoch changes, while the line plots show the Wasserstein distance from initialization across layers for each epoch.

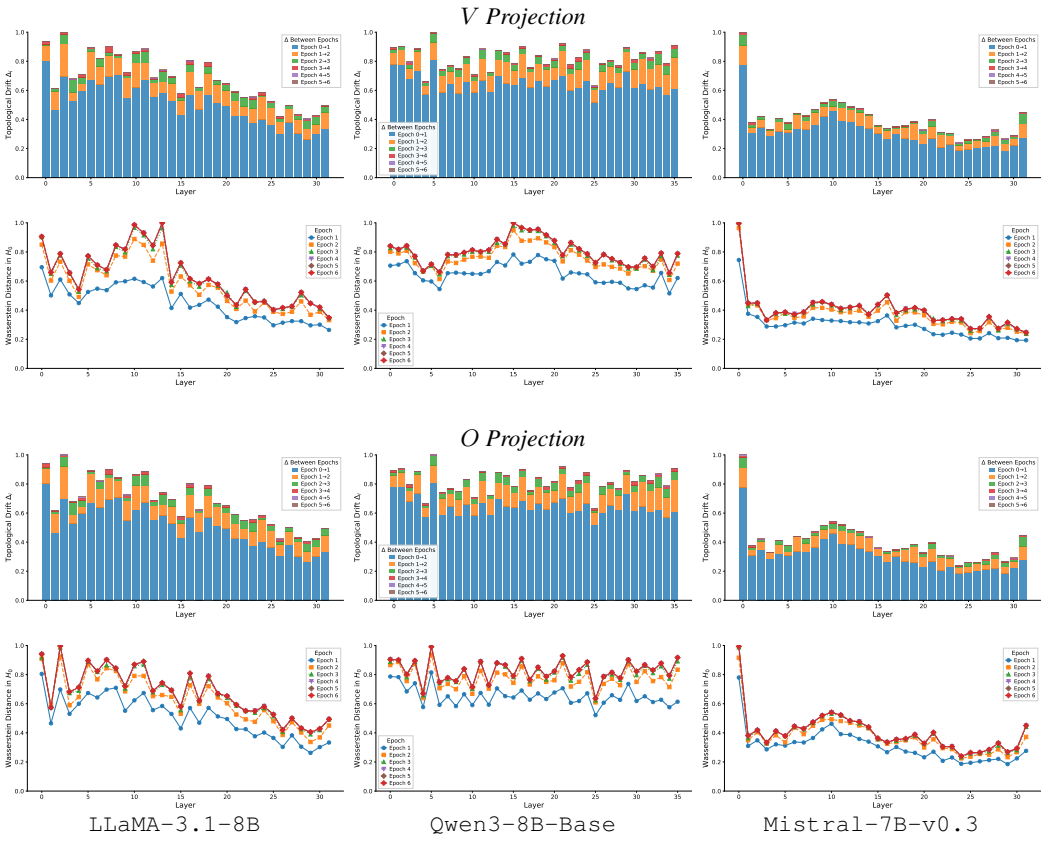


Figure 8: Topological drift of the V and O projection on `QA:GSM8K` under H_0. The top row shows epoch-to-epoch drift bars, and the bottom row shows Wasserstein distance curves across layers.

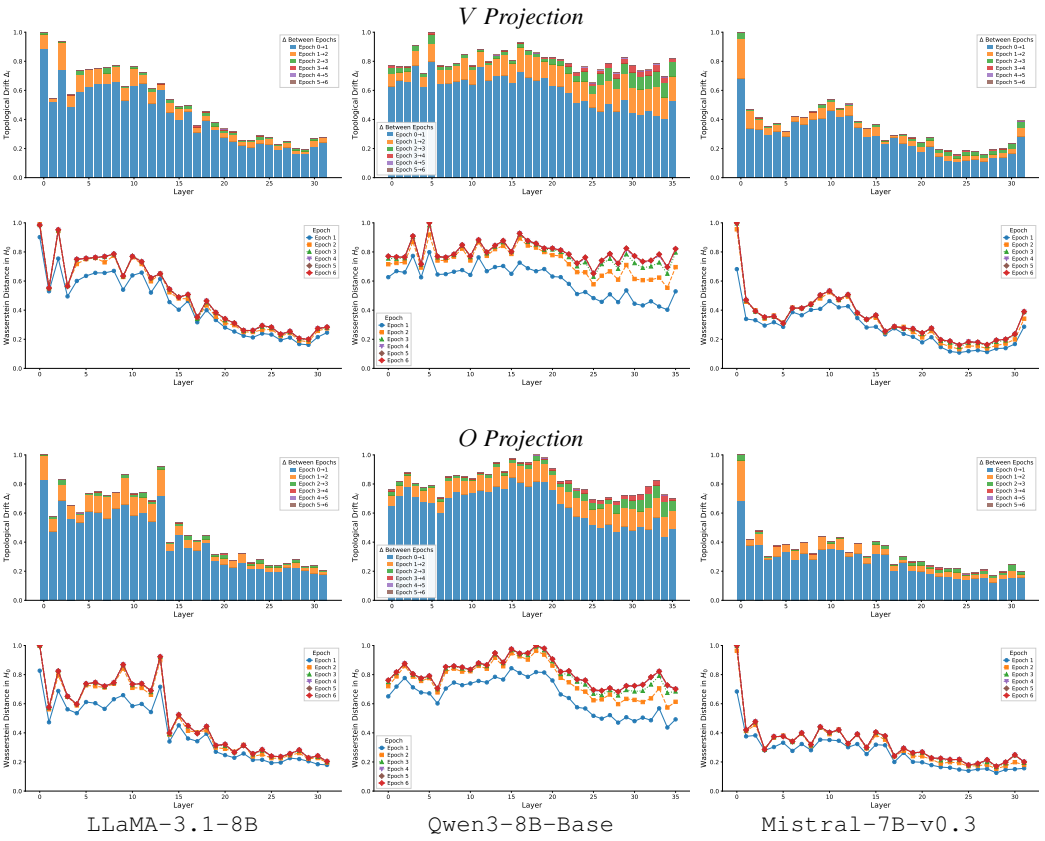


Figure 9: Topological drift of the V and O projection on `QA:MMLU` under H_0. The top row shows epoch-to-epoch drift bars, and the bottom row shows Wasserstein distance curves across layers.

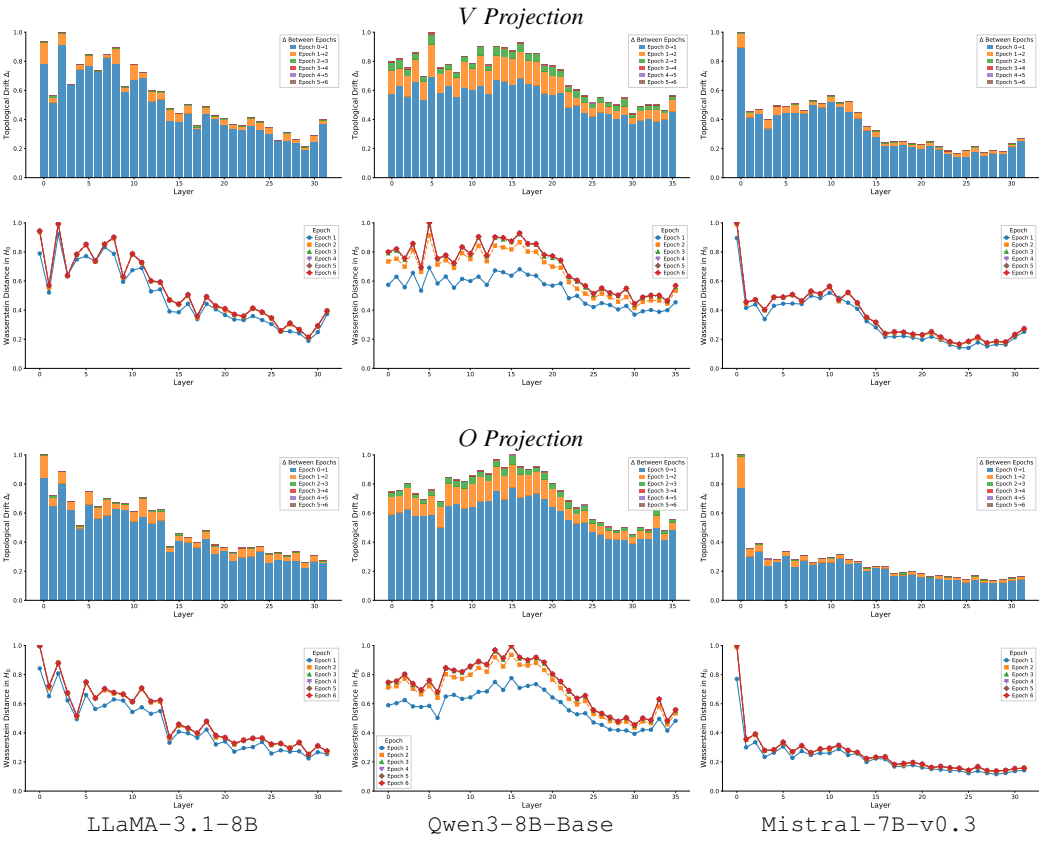


Figure 10: Topological drift of the V projection on `SA:IMDB` under H_0. The top row shows epoch-to-epoch drift bars, and the bottom row shows Wasserstein distance curves across layers.

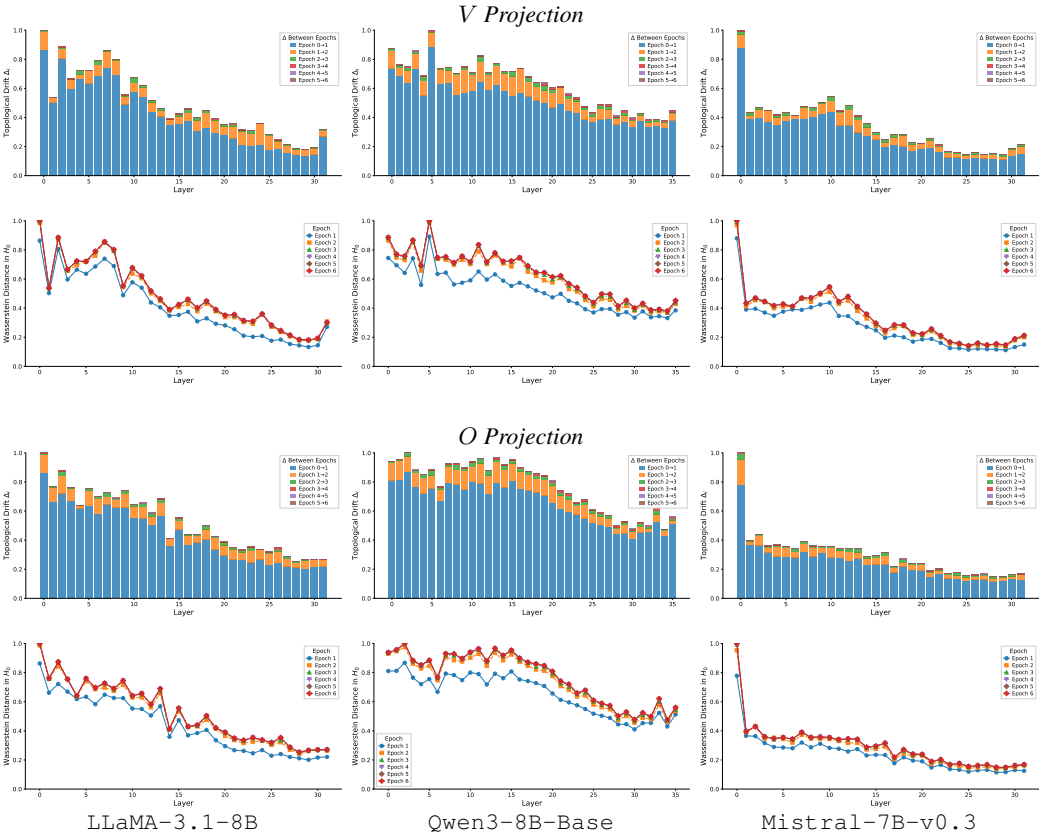


Figure 11: Topological drift of the V and O projection on `SA:SST-2` under H_0. The top row shows epoch-to-epoch drift bars, and the bottom row shows Wasserstein distance curves across layers.

H Wasserstein Distances in Models

Wasserstein distance. We also report the Wasserstein-based structural distance used by our topological analysis. As defined later in Definition 4.1, each projection matrix is treated as a row cloud and compared to its pretrained version through the Wasserstein distance between persistence diagrams. The Wasserstein plots in Figure 12 show this epoch-6 structural change for the V and O projections under the same model and dataset ordering as the entrywise plots.

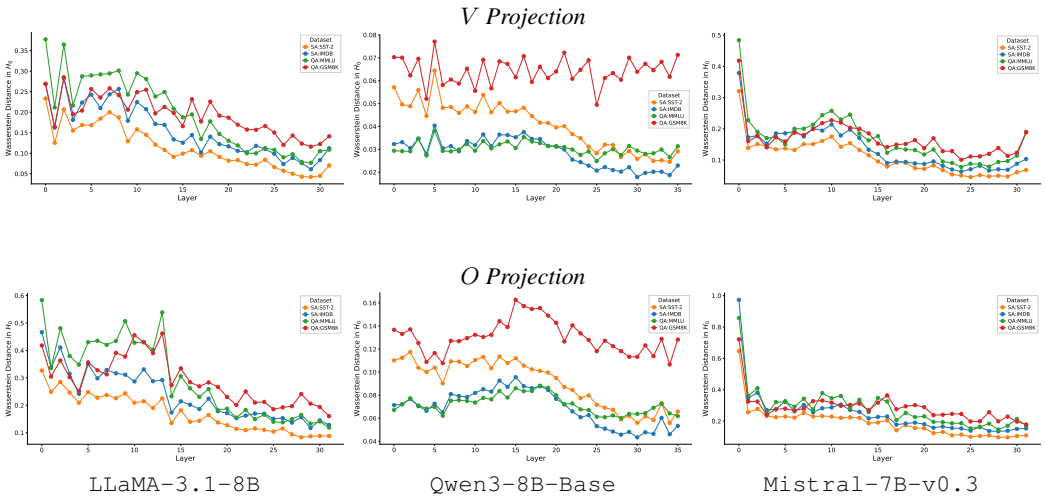


Figure 12: Epoch-6 Wasserstein distances for the V and O projection matrices under H_0.

I Fine-tuning with Perfect Knowledge

Table 7 reports the perfect-knowledge `QA:GSM8K` results for `Qwen3-8B-Base`, `LLaMA-3.1-8B`, and `Mistral-7B-v0.3`. Across all three models, selective freezing recovers much of the full-fine-tuning performance while training only about 5% to 8% of the weights. For the best selective configurations in `Qwen3-8B-Base`, TDA-High3 reaches 87.62 ± 0.18 with 8.45% trainable parameters, close to LoRA at 87.89 ± 0.90 and above full fine-tuning at 86.27 ± 0.75. For `LLaMA-3.1-8B`, TDA-High6 reaches 61.17 ± 0.75 with 6.79% trainable parameters, slightly above full fine-tuning at 60.70 ± 0.87 and above LoRA at 56.90 ± 1.92. For `Mistral-7B-v0.3`, TDA-High3 reaches 52.87 ± 0.46 with 8.39% trainable parameters, compared with full fine-tuning at 59.59 ± 0.85 and LoRA at 49.28 ± 1.94.

Table 7: **Perfect Knowledge.** Fine-tuning models on `QA:GSM8K` with selected V and O freezing. Wasserstein distances use H_0. Projection IDs and results of other k values are given in Tables 8, 9, and 10.

Method	Qwen3-8B-Base				Llama-3.1-8B				Mistral-7B-v0.3			
	Train.%	Upd. %	Acc.±Std (↑)	Time (min)	Train.%	Upd.	Acc.±Std (↑)	Time (min)	Train.%	Upd.	Acc.±Std (↑)	Time (min)
Full Fine-tuning	100	14.63	86.27 ± 0.75	47.3	100	27.54	60.70 ± 0.87	38.7	100	31.46	59.59 ± 0.85	43.3
PEFT												
LoRA	0.19	18.41	$\mathbf{87.89 \pm 0.90}$	35.4	0.17	16.70	56.90 ± 1.92	28.6	0.19	18.49	49.28 ± 1.94	33.5
TDA-based tuning (ours)												
High3	8.45	1.22	$\underline{87.62 \pm 0.18}$	34.5	7.57	2.64	$\underline{59.33 \pm 0.46}$	26.2	8.39	2.82	$\mathbf{52.87 \pm 0.46}$	30.5
High6	7.68	1.12	86.91 ± 0.46	34.4	6.79	2.33	$\mathbf{61.17 \pm 0.75}$	25.7	6.8	2.52	$\underline{52.64 \pm 0.99}$	30.5
High9	6.91	1.00	86.91 ± 0.16	34.4	6.01	2.05	58.97 ± 0.76	25.5	5.21	2.23	51.40 ± 0.92	30.3

These results show two patterns. First, selective freezing yields large reductions in updated parameters with competitive performance across all three models. Second, TDA-based freezing gives the best selective result for `Qwen3-8B-Base` and `Mistral-7B-v0.3`, and ties with `LLaMA-3.1-8B`. For more details, see Tables 8, 9, and 10.

Table 8: `LLaMA-3.1-8B` on `QA:GSM8K` with separate V and O freezing. Among selective freezing settings, the best configuration is tied at TDA-High6 (61.17%) and Eltwise-Low9 (61.17%).

Method	Frozen Layers	Trainable %	Avg. Accuracy±Std (↑)	Time (min)
Baselines				
Full	–	100	60.70 ± 0.87	38.7
LoRA	–	0.17	56.90 ± 1.92	28.6
Magnitude-based Tuning (ours)				
Eltwise	Low3: V:[9, 10, 21] | O:[29, 31, 23]	7.57	59.80 ± 2.00	26.4
Eltwise	Low6: V:[9, 10, 21, 29, 8, 18] | O:[29, 31, 23, 27, 7, 4]	6.79	59.33 ± 1.93	26.3
Eltwise	Low9: V:[9, 10, 21, 29, 8, 18, 7, 2, 5] | O:[29, 31, 23, 27, 7, 4, 3, 28, 12]	6.01	$\mathbf{61.17 \pm 1.42}$	26.2
Eltwise	Low12: V:[9, 10, 21, 29, 8, 18, 7, 2, 5, 30, 17, 23] | O:[29, 31, 23, 27, 7, 4, 3, 28, 12, 9, 6, 14]	5.22	60.57 ± 1.16	26.3
Eltwise	Low15: V:[9, 10, 21, 29, 8, 18, 7, 2, 5, 30, 17, 23, 19, 16, 4] | O:[29, 31, 23, 27, 7, 4, 3, 28, 12, 9, 6, 14, 8, 1, 19]	4.44	60.50 ± 1.04	26.2
Eltwise	High3: V:[27, 13, 20] | O:[26, 18, 25]	7.57	59.73 ± 2.06	26.2
Eltwise	High6: V:[25, 6, 28, 27, 13, 20] | O:[20, 16, 0, 26, 18, 25]	6.79	59.60 ± 0.35	26.3
Eltwise	High9: V:[24, 15, 14, 25, 6, 28, 27, 13, 20] | O:[30, 13, 10, 20, 16, 0, 26, 18, 25]	6.01	59.47 ± 0.97	26.2
Eltwise	High12: V:[12, 0, 1, 24, 15, 14, 25, 6, 28, 27, 13, 20] | O:[15, 17, 21, 30, 13, 10, 20, 16, 0, 26, 18, 25]	5.22	58.27 ± 0.31	26.2
Eltwise	High15: V:[26, 11, 31, 12, 0, 1, 24, 15, 14, 25, 6, 28, 27, 13, 20] | O:[24, 22, 5, 15, 17, 21, 30, 13, 10, 20, 16, 0, 26, 18, 25]	4.44	58.30 ± 1.60	26.1
TDA-based Tuning (ours)				
$H0$	Low3: V:[29, 26, 30] | O:[31, 30, 26]	7.57	58.17 ± 0.42	25.4
$H0$	Low6: V:[26, 27, 28, 29, 30, 31] | O:[24, 25, 26, 29, 30, 31]	6.79	57.43 ± 0.25	25.4
$H0$	Low9: V:[22, 23, 25, 26, 27, 28, 29, 30, 31] | O:[21, 23, 24, 25, 26, 27, 29, 30, 31]	6.01	58.60 ± 0.56	25.6
$H0$	Low12: V:[15, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31] | O:[20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31]	5.22	58.43 ± 1.40	25.8
$H0$	Low15: V:[29, 26, 30, 28, 31, 27, 25, 22, 23, 15, 24, 21, 17, 3, 1] | O:[31, 30, 26, 29, 25, 24, 27, 21, 23, 28, 22, 20, 16, 4, 19]	4.44	58.43 ± 0.58	25.1
$H0$	High3: V:[8, 2, 0] | O:[11, 10, 13]	7.57	59.33 ± 0.46	26.2
$H0$	High6: V:[0, 2, 7, 8, 10, 11] | O:[0, 5, 9, 10, 11, 13]	6.79	$\mathbf{61.17 \pm 0.75}$	25.7
$H0$	High9: V:[0, 2, 5, 6, 7, 8, 10, 11, 16] | O:[0, 2, 5, 8, 9, 10, 11, 12, 13]	6.01	58.97 ± 0.76	25.5
$H0$	High12: V:[0, 2, 5, 6, 7, 8, 10, 11, 12, 13, 16, 18] | O:[0, 1, 2, 5, 6, 7, 8, 9, 10, 11, 12, 13]	5.22	58.37 ± 0.60	25.9
$H0$	High15: V:[0, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 18] | O:[0, 1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 18]	4.44	57.80 ± 1.18	26.0

Table 9: `Qwen3-8B-Base` on `QA:GSM8K` with separate V and O freezing. Among selective freezing settings, the best configuration is TDA-High3 (%87.62), followed by Eltwise-High9 (%87.43) and Eltwise-Low12 (%87.40).

Method	Frozen V/O Layers	Trainable %	Avg. Accuracy±Std (↑)	Time (min)
Baselines				
Full	–	100	86.27 ± 0.75	47.3
LoRA	–	0.19	87.89 ± 0.90	35.4
Magnitude-based Tuning (ours)				
Eltwise	Low3: V:[11, 24, 8] | O:[32, 34, 24]	8.45	87.30 ± 0.36	34.6
Eltwise	Low6: V:[11, 24, 8, 12, 18, 33] | O:[32, 34, 24, 10, 27, 33]	7.68	87.23 ± 0.12	34.5
Eltwise	Low9: V:[11, 24, 8, 12, 18, 33, 5, 34, 15] | O:[32, 34, 24, 10, 27, 33, 19, 6, 4]	6.91	87.23 ± 0.21	34.4
Eltwise	Low12: V:[11, 24, 8, 12, 18, 33, 5, 34, 15, 29, 9, 35] | O:[32, 34, 24, 10, 27, 33, 19, 6, 4, 13, 11, 20]	6.14	$\mathbf{87.40 \pm 0.44}$	34.2
Eltwise	Low15: V:[11, 24, 8, 12, 18, 33, 5, 34, 15, 29, 9, 35, 0, 2, 4] | O:[32, 34, 24, 10, 27, 33, 19, 6, 4, 13, 11, 20, 28, 29, 9]	5.38	86.80 ± 0.17	34.2
Eltwise	High3: V:[32, 22, 26] | O:[35, 2, 30]	8.45	87.33 ± 0.15	34.6
Eltwise	High6: V:[28, 3, 13, 32, 22, 26] | O:[21, 5, 15, 35, 2, 30]	7.68	87.17 ± 0.15	34.5
Eltwise	High9: V:[1, 19, 31, 28, 3, 13, 32, 22, 26] | O:[14, 8, 26, 21, 5, 15, 35, 2, 30]	6.91	$\mathbf{87.43 \pm 0.57}$	34.4
Eltwise	High12: V:[17, 27, 10, 1, 19, 31, 28, 3, 13, 32, 22, 26] | O:[17, 22, 16, 14, 8, 26, 21, 5, 15, 35, 2, 30]	6.14	86.83 ± 0.29	34.2
Eltwise	High15: V:[25, 20, 23, 17, 27, 10, 1, 19, 31, 28, 3, 13, 32, 22, 26] | O:[1, 3, 25, 17, 22, 16, 14, 8, 26, 21, 5, 15, 35, 2, 30]	5.38	86.37 ± 0.65	34.1
TDA-based Tuning (ours)				
$H0$	Low3: V:[25, 10, 4] | O:[6, 34, 29]	8.45	86.40 ± 0.38	34.7
$H0$	Low6: V:[25, 10, 4, 6, 12, 17] | O:[6, 34, 29, 25, 28, 4]	7.68	86.91 ± 0.27	34.6
$H0$	Low9: V:[25, 10, 4, 6, 12, 17, 8, 15, 7] | O:[6, 34, 29, 25, 28, 4, 30, 26, 31]	6.91	86.86 ± 0.42	34.4
$H0$	Low12: V:[25, 10, 4, 6, 12, 17, 8, 15, 7, 19, 26, 34] | O:[6, 34, 29, 25, 28, 4, 30, 26, 31, 32, 9, 35]	6.14	86.30 ± 0.35	34.0
$H0$	Low15: V:[25, 10, 4, 6, 12, 17, 8, 15, 7, 19, 26, 34, 24, 28, 30] | O:[6, 34, 29, 25, 28, 4, 30, 26, 31, 32, 9, 35, 27, 24, 5]	5.38	86.33 ± 0.19	34.1
$H0$	High3: V:[1, 5, 0] | O:[18, 17, 15]	8.45	$\mathbf{87.62 \pm 0.18}$	34.5
$H0$	High6: V:[31, 29, 1, 5, 0, 13] | O:[14, 19, 17, 18, 15, 16]	7.68	86.91 ± 0.46	34.4
$H0$	High9: V:[32, 3, 21, 13, 31, 29, 1, 5, 0] | O:[20, 13, 22, 16, 14, 19, 17, 18, 15]	6.91	86.91 ± 0.16	34.4
$H0$	High12: V:[16, 33, 35, 32, 3, 21, 13, 31, 29, 1, 5, 0] | O:[2, 11, 0, 20, 13, 22, 16, 14, 19, 17, 18, 15]	6.14	87.10 ± 0.12	34.0
$H0$	High15: V:[14, 20, 11, 16, 33, 35, 32, 3, 21, 13, 31, 29, 1, 5, 0] | O:[1, 12, 23, 2, 11, 0, 20, 13, 22, 16, 14, 19, 17, 18, 15]	5.38	86.88 ± 0.31	34.1

Table 10: `Mistral-7B-v0.3` on `QA:GSM8K` with separate V and O freezing. Among selective freezing settings, the best configuration is TDA-Low6 (54.31%), followed by Eltwise-High6 (53.96%).

Method	Frozen Layers	Trainable %	Avg. Accuracy±Std (↑)	Time (min)
Baselines				
Full	–	100	59.59 ± 0.85	43.3
LoRA	–	0.19	49.28 ± 1.94	33.5
Magnitude-based Tuning (ours)				
Eltwise	Low3: V:[6, 10, 26] | O:[14, 18, 19]	8.39	53.58 ± 0.55	30.5
Eltwise	Low6: V:[1, 6, 10, 14, 19, 26] | O:[11, 14, 18, 19, 27, 29]	6.8	53.78 ± 1.09	30.5
Eltwise	Low9: V:[1, 3, 6, 7, 10, 11, 14, 19, 26] | O:[1, 11, 13, 14, 18, 19, 21, 27, 29]	5.21	53.52 ± 0.77	30.4
Eltwise	Low12: V:[1, 3, 6, 7, 10, 11, 13, 14, 17, 19, 21, 26] | O:[1, 2, 8, 11, 13, 14, 18, 19, 21, 27, 29, 30]	3.62	52.29 ± 0.61	30.3
Eltwise	Low15: V:[1, 2, 3, 6, 7, 10, 11, 13, 14, 17, 19, 21, 23, 26, 31] | O:[1, 2, 5, 8, 11, 13, 14, 15, 18, 19, 21, 25, 27, 29, 30]	2.03	51.91 ± 0.42	30.1
Eltwise	High3: V:[8, 9, 30] | O:[4, 6, 12]	8.39	$\mathbf{53.90 \pm 2.63}$	31.3
Eltwise	High6: V:[4, 8, 9, 16, 29, 30] | O:[4, 6, 7, 12, 28, 31]	6.8	$\mathbf{53.96 \pm 1.02}$	30.4
Eltwise	High9: V:[0, 4, 8, 9, 16, 18, 25, 29, 30] | O:[0, 3, 4, 6, 7, 12, 22, 28, 31]	5.21	52.29 ± 1.54	30.3
Eltwise	High12: V:[0, 4, 8, 9, 12, 16, 18, 20, 24, 25, 29, 30] | O:[0, 3, 4, 6, 7, 10, 12, 16, 22, 23, 28, 31]	3.62	53.73 ± 1.05	30.3
Eltwise	High15: V:[0, 4, 5, 8, 9, 12, 16, 18, 20, 22, 24, 25, 28, 29, 30] | O:[0, 3, 4, 6, 7, 9, 10, 12, 16, 17, 20, 22, 23, 28, 31]	2.03	53.60 ± 0.23	30.1
TDA-based Tuning (ours)				
$H0$	Low3: V:[24, 25, 29] | O:[25, 28, 31]	8.39	$\mathbf{53.93 \pm 1.06}$	30.4
$H0$	Low6: V:[22, 24, 25, 26, 29, 30] | O:[22, 25, 28, 29, 30, 31]	6.8	$\mathbf{54.31 \pm 1.64}$	30.5
$H0$	Low9: V:[22, 23, 24, 25, 26, 27, 28, 29, 30] | O:[21, 22, 24, 25, 26, 28, 29, 30, 31]	5.21	53.88 ± 0.23	30.4
$H0$	Low12: V:[16, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31] | O:[17, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31]	3.62	52.79 ± 1.01	30.1
$H0$	Low15: V:[16, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31] | O:[3, 6, 17, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31]	2.03	52.82 ± 0.56	30.0
$H0$	High3: V:[0, 10, 12] | O:[0, 1, 2]	8.39	52.87 ± 0.46	30.5
$H0$	High6: V:[0, 8, 9, 10, 11, 12] | O:[0, 1, 2, 9, 11, 16]	6.8	52.64 ± 0.99	30.5
$H0$	High9: V:[0, 2, 6, 8, 9, 10, 11, 12, 13] | O:[0, 1, 2, 9, 10, 11, 13, 15, 16]	5.21	51.40 ± 0.92	30.3
$H0$	High12: V:[0, 2, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14] | O:[0, 1, 2, 5, 7, 9, 10, 11, 12, 13, 15, 16]	3.62	50.64 ± 0.95	30.2
$H0$	High15: V:[0, 1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15] | O:[0, 1, 2, 4, 5, 7, 8, 9, 10, 11, 12, 13, 15, 16, 19]	2.03	48.60 ± 0.60	30.0

J Matrix Types, Weight Changes, and Freezing Profiles

To quantify projection-level changes, we compare each epoch-6 fine-tuned matrix with its pretrained counterpart. For each projection matrix $W \in \{K, Q, V, O\}$, we compute mean element-wise relative distance:

$$d(W) = \frac{1}{n} \sum_{i=1}^{n} \frac{|W_{6,i} - W_{0,i}|}{|W_{0,i}|},$$

where W_0 and W_6 denote the pretrained and epoch-6 weights, n is the number of matrix entries.

We justify our matrix selection design by ranking all candidate matrices by their average distance to the pretrained model. We compute the average rank of each matrix family (K, Q, V, and O) and count how many K or Q matrices appear among the top distance matrices. As Table 11 and Table 12 show, empirically, K and Q occupy low positions in the ranking and are largely absent from the top-ranked set, which supports restricting selective tuning to V and O.

Table 11: Normalized average projection ranks (rank divided by total count). Lower values indicate larger distance. `LLaMA-3.1-8B`/ `Mistral-7B-v0.3` are normalized by 128, `Qwen3-8B-Base` by 144.

Dataset	Statistic	`LLaMA-3.1-8B`	`Mistral-7B-v0.3`	`Qwen3-8B-Base`
`QA:GSM8K`	Avg K rank	0.85	0.71	0.29
	Avg Q rank	0.58	0.48	0.33
	Avg V rank	0.26	0.41	0.75
	Avg O rank	0.32	0.41	0.65
`SA:IMDB`	Avg K rank	0.79	0.61	0.41
	Avg Q rank	0.51	0.53	0.44
	Avg V rank	0.35	0.44	0.62
	Avg O rank	0.36	0.43	0.55
`QA:MMLU`	Avg K rank	0.72	0.66	0.43
	Avg Q rank	0.54	0.52	0.42
	Avg V rank	0.37	0.43	0.64
	Avg O rank	0.39	0.41	0.53
`SA:SST-2`	Avg K rank	0.76	0.61	0.39
	Avg Q rank	0.51	0.51	0.44
	Avg V rank	0.36	0.44	0.60
	Avg O rank	0.39	0.45	0.59

Table 12: Top-N projection counts under mean element-wise relative drift. Each cell reports K Q $(K{+}Q)$ | V O $(V{+}O)$ among the top-N most changed projections. `LLaMA-3.1-8B`/`Mistral-7B-v0.3` runs contain 128 projection matrices from 32 layers, while `Qwen3-8B-Base` runs contain 144 projection matrices from 36 layers.

Dataset	Top-N	`LLaMA-3.1-8B`	`Mistral-7B-v0.3`	`Qwen3-8B-Base`
`QA:GSM8K`	Top-10	1 1 (2) | 6 2 (8)	1 4 (5) | 3 2 (5)	8 2 (10) | 0 0 (0)
	Top-20	1 1 (2) | 15 3 (18)	3 7 (10) | 7 3 (10)	14 6 (20) | 0 0 (0)
	Top-30	1 1 (2) | 21 7 (28)	3 10 (13) | 12 5 (17)	16 14 (30) | 0 0 (0)
`SA:IMDB`	Top-10	1 1 (2) | 6 2 (8)	0 0 (0) | 7 3 (10)	6 4 (10) | 0 0 (0)
	Top-20	1 1 (2) | 12 6 (18)	1 0 (1) | 13 6 (19)	9 11 (20) | 0 0 (0)
	Top-30	1 2 (3) | 14 13 (27)	1 1 (2) | 14 14 (28)	15 13 (28) | 0 2 (2)
`QA:MMLU`	Top-10	1 1 (2) | 6 2 (8)	1 0 (1) | 6 3 (9)	6 4 (10) | 0 0 (0)
	Top-20	1 1 (2) | 13 5 (18)	1 0 (1) | 12 7 (19)	10 7 (17) | 1 2 (3)
	Top-30	1 1 (2) | 15 13 (28)	1 0 (1) | 14 15 (29)	10 12 (22) | 4 4 (8)
`SA:SST-2`	Top-10	1 1 (2) | 6 2 (8)	0 0 (0) | 7 3 (10)	8 2 (10) | 0 0 (0)
	Top-20	1 1 (2) | 11 7 (18)	0 0 (0) | 12 8 (20)	13 7 (20) | 0 0 (0)
	Top-30	1 2 (3) | 14 13 (27)	0 1 (1) | 15 14 (29)	17 11 (28) | 1 1 (2)

K Model Weight-Change Statistics

We report full-model weight-change statistics by comparing each fine-tuned checkpoint against its pretrained initialization. Changed Params. quantifies how widely the checkpoint differs from the pretrained model by counting parameters whose absolute update exceeds a fixed threshold:

$$|w_{\text{final}} - w_{\text{base}}| > 10^{-6}.$$

Mean Rel. Δ measures the average relative magnitude of the parameter update:

$$\text{Mean Rel. } \Delta = \text{mean}\left(\frac{|w_{\text{final}} - w_{\text{base}}|}{|w_{\text{base}}|}\right).$$

Together, these statistics capture both the spread of parameter changes across the model and the average size of those changes relative to the pretrained weights. Table 13 reports these statistics across all 12 model–dataset settings and shows the substantially smaller update footprint of TDA-High3 relative to LoRA.

Table 13: Model weight-change statistics relative to the pretrained checkpoint. TDA-High3 uses 86.21% less parameter updates than LoRA across the 12 LLM-dataset settings.

Method	Qwen3-8B-Base				LLaMA-3.1-8B				Mistral-7B-v0.3			
	Total Params.	Changed Params.	Upd. (%)	Mean Rel. Δ (%)	Total Params.	Changed Params.	Upd. (%)	Mean Rel. Δ (%)	Total Params.	Changed Params.	Upd. (%)	Mean Rel. Δ (%)
QA:GSM8K												
Full	8.19B	1.198B	14.63	12.57	8.03B	2.211B	27.54	26.12	7.25B	2.280B	31.46	25.57
LoRA	8.19B	1.508B	18.41	3.89	8.03B	1.341B	16.70	9.13	7.25B	1.340B	18.49	36.57
TDA-High3	8.19B	0.100B	1.22	1.65	8.03B	0.212B	2.64	3.17	7.25B	0.204B	2.82	3.02
TDA-High6	8.19B	0.091B	1.12	1.55	8.03B	0.187B	2.33	2.89	7.25B	0.183B	2.52	2.79
TDA-High9	8.19B	0.082B	1.00	1.45	8.03B	0.165B	2.05	2.60	7.25B	0.162B	2.23	2.55
Eltwise-High3	8.19B	0.099B	1.21	1.62	8.03B	0.220B	2.74	3.17	7.25B	0.213B	2.94	3.11
Eltwise-High6	8.19B	0.091B	1.11	1.52	8.03B	0.196B	2.44	2.88	7.25B	0.194B	2.67	2.85
Eltwise-High9	8.19B	0.082B	1.01	1.41	8.03B	0.175B	2.18	2.63	7.25B	0.167B	2.30	2.53
SA:SST-2												
Full	8.19B	1.066B	13.01	8.78	8.03B	1.757B	21.88	11.07	7.25B	1.812B	25.00	12.56
LoRA	8.19B	0.903B	11.02	1.01	8.03B	1.335B	16.62	2.03	7.25B	1.334B	18.41	15.87
TDA-High3	8.19B	0.094B	1.15	0.93	8.03B	0.162B	2.02	1.27	7.25B	0.177B	2.44	1.44
SA:IMDB												
Full	8.19B	0.683B	8.34	6.69	8.03B	2.004B	24.95	16.87	7.25B	2.078B	28.67	16.88
LoRA	8.19B	1.505B	18.38	3.11	8.03B	1.338B	16.67	5.04	7.25B	1.340B	18.49	39.57
TDA-High3	8.19B	0.046B	0.56	0.78	8.03B	0.373B	4.65	1.88	7.25B	0.352B	4.85	1.96
QA:MMLU												
Full	8.19B	0.770B	9.39	7.16	8.03B	2.513B	31.29	21.72	7.25B	2.564B	35.38	21.22
LoRA	8.19B	1.508B	18.41	4.02	8.03B	1.339B	16.67	5.33	7.25B	1.340B	18.49	34.39
TDA-High3	8.19B	0.056B	0.68	0.86	8.03B	0.200B	2.49	2.06	7.25B	0.216B	2.98	2.13

L MAGNITUDE CHANGES IN MODELS

Entrywise weight change. We first report entrywise weight movement as a magnitude-based baseline. For each fine-tuned matrix, we compare the epoch-6 weights against the pretrained weights and summarize the raw mean absolute change for the V and O projections. The entrywise plots in Figure 13 provide a direct view of parameter movement across datasets and model families. This magnitude-based view is later compared with the topology-based distance used for matrix ranking and freezing.

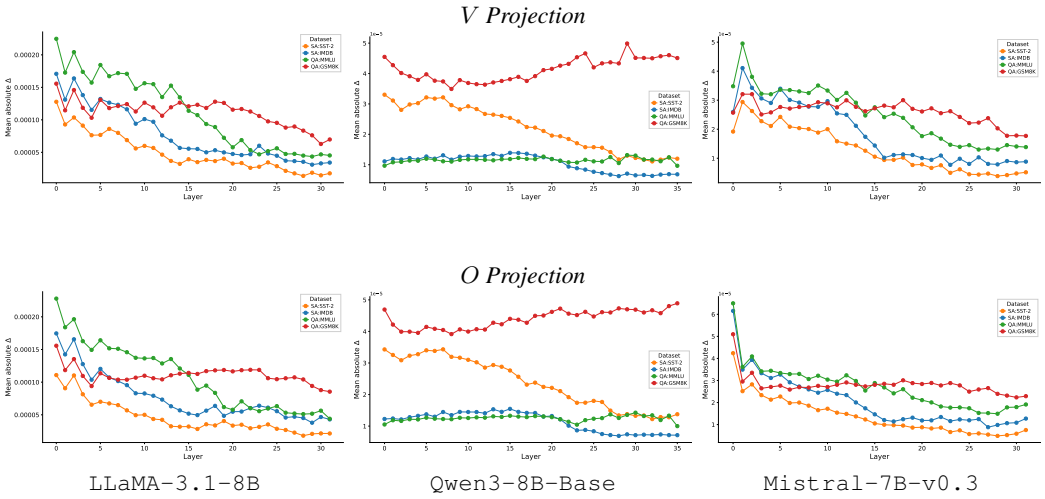


Figure 13: Epoch-6 Eltwise distances for the V and O projection matrices

M PROFILE REUSE ACCURACY CURVES

Figure 14 shows the epoch-wise accuracy trajectories for `QA:MMLU`, `SA:SST-2`, and `SA:IMDB` in the profile-reuse setting. The curves compare full fine-tuning, LoRA, and TOPOTUNER after reusing the freezing profile learned from `QA:GSM8K`. These trajectories provide a per-epoch view of model adaptation across the reused evaluation tasks.

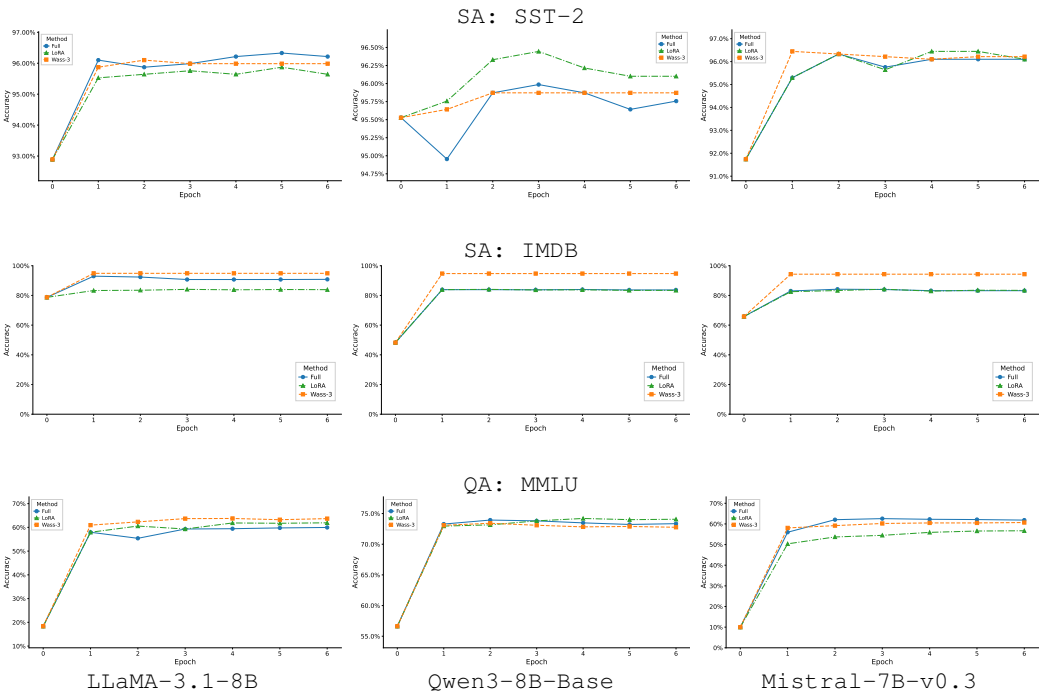


Figure 14: Accuracies on each epoch for all the models across datasets

N Fine-tuning with Dynamic Stopping

Figure 15 shows that larger η values generally lead to earlier stopping. The overall trend is consistent across models, while the exact stopping transitions vary by dataset and architecture.

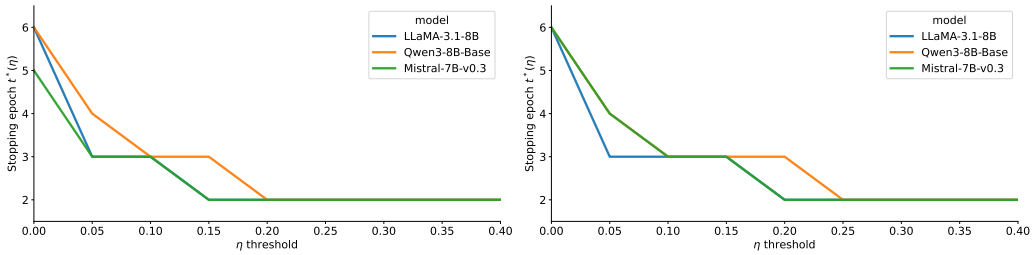


Figure 15: Topology-based stopping epochs under full fine-tuning. The plots show selected stopping epochs across η thresholds for SA:IMDB and QA:MMLU across the three model families.

O Catastrophic Forgetting Evaluation

The retention results show a second advantage of selective tuning. Tables 14– 16 evaluate the models on 11 different datasets after prior fine-tuning on QA:GSM8K. The clearest retention gains appear for LLaMA-3.1-8B and Mistral-7B-v0.3. For both models, TopoTuner improves over LoRA on QA:MMLU, SA:SST-2, SA:IMDB, IR:SQuAD, and IR:HotpotQA, showing stronger preservation on question-answering, sentiment-analysis, and information-retrieval evaluations.

Table 14: Catastrophic forgetting evaluation on QA:GSM8K fine-tuned LLaMA-3.1-8B. Metrics are avg. ± std over three fixed, non-overlapping evaluation subsets. Base and full finetuned models are shown as reference rows. **Bold** indicates the better result between LoRA and TopoTuner for each metric.

	Question Answering	*Sentiment Analysis*		*Information Retrieval*			
	QA:MMLU	SA:SST-2	SA:IMDB	IR:SQuAD v1.1		IR:HotpotQA	
Model	Acc.	Acc.	Acc.	EM	F1	EM	F1
Base model	54.5 ± 1.25	94.0 ± 0.72	89.8 ± 0.38	76.0 ± 0.32	86.9 ± 0.20	23.1 ± 1.25	31.9 ± 0.64
Full finetuned	52.9 ± 0.44	94.3 ± 0.80	89.0 ± 0.47	74.4 ± 0.39	85.0 ± 0.15	21.3 ± 1.40	29.4 ± 0.76
LoRA	51.8 ± 1.01	92.7 ± 0.52	89.3 ± 0.48	72.3 ± 0.49	83.6 ± 0.66	19.2 ± 1.03	27.2 ± 0.97
TopoTuner	**53.6 ± 0.51**	**94.4 ± 0.19**	**90.3 ± 0.36**	**76.9 ± 0.24**	**86.0 ± 0.48**	**21.8 ± 1.12**	**30.2 ± 0.71**

	Summarization						*Code Generation*	
	SU:XSum			SU:CNN/DM			CG:HumanEval	CG:MBPP
Model	R1	R2	RL	R1	R2	RL	pass@1	pass@1
Base model	25.2 ± 0.05	6.7 ± 0.08	18.8 ± 0.10	31.7 ± 0.29	13.6 ± 0.18	20.6 ± 0.23	0.0 ± 0.00	2.7 ± 1.35
Full finetuned	14.8 ± 0.16	4.1 ± 0.11	11.7 ± 0.17	29.5 ± 0.34	12.4 ± 0.24	18.9 ± 0.28	1.8 ± 1.85	3.5 ± 3.07
LoRA	**27.2 ± 0.21**	**7.8 ± 0.22**	**21.6 ± 0.35**	**29.8 ± 0.27**	**12.6 ± 0.22**	**19.2 ± 0.19**	1.2 ± 1.05	**4.7 ± 3.48**
TopoTuner	**27.2 ± 0.17**	7.7 ± 0.08	21.0 ± 0.21	27.4 ± 0.45	11.8 ± 0.25	17.7 ± 0.34	**1.8 ± 3.21**	3.1 ± 1.33

	Instruction Following							
	IF:Dolly-15k				IF:Alpaca			
Model	R1	R2	RL	METEOR	R1	R2	RL	METEOR
Base model	38.0 ± 0.32	19.3 ± 0.25	29.9 ± 0.25	27.2	32.6 ± 0.27	15.0 ± 0.25	26.1 ± 0.18	25.7
Full finetuned	38.5 ± 0.17	19.4 ± 0.14	30.5 ± 0.09	27.1	39.4 ± 0.21	19.0 ± 0.28	32.3 ± 0.17	27.2
LoRA	**38.0 ± 0.31**	**19.5 ± 0.14**	**30.5 ± 0.20**	27.0	**37.0 ± 0.15**	**18.2 ± 0.16**	**30.7 ± 0.13**	**26.0**
TopoTuner	37.2 ± 0.09	18.9 ± 0.26	29.4 ± 0.18	**27.5**	31.9 ± 0.27	14.3 ± 0.27	25.7 ± 0.28	25.5

Table 15: Catastrophic forgetting evaluation on `QA:GSM8K` fine-tuned `Qwen3-8B-Base`. Metrics are avg. ± std over three fixed, non-overlapping evaluation subsets. Base and full finetuned models are shown as reference rows. **Bold** indicates the better result between LoRA and TOPOTUNER for each metric.

	Question Answering	*Sentiment Analysis*		*Information Retrieval*			
	QA:MMLU	SA:SST-2	SA:IMDB	IR:SQuAD v1.1		IR:HotpotQA	
Model	Acc.	Acc.	Acc.	EM	F1	EM	F1
Base model	64.2 ± 0.78	94.0 ± 0.79	90.0 ± 0.26	67.2 ± 0.57	82.9 ± 0.38	21.7 ± 0.94	30.4 ± 0.60
Full finetuned	64.4 ± 0.66	93.9 ± 0.99	89.6 ± 0.25	75.2 ± 0.19	87.3 ± 0.44	21.9 ± 0.40	30.4 ± 0.05
LoRA	$\mathbf{64.4 \pm 0.58}$	$\mathbf{94.4 \pm 0.79}$	89.9 ± 0.35	$\mathbf{77.8 \pm 0.19}$	$\mathbf{88.7 \pm 0.20}$	22.0 ± 0.39	30.6 ± 0.24
TOPOTUNER	64.3 ± 0.63	93.8 ± 0.91	$\mathbf{90.3 \pm 0.22}$	74.4 ± 0.25	86.9 ± 0.36	$\mathbf{22.2 \pm 0.64}$	$\mathbf{30.7 \pm 0.16}$

	Summarization						*Code Generation*	
	SU:XSum			SU:CNN/DM			CG:HumanEval	CG:MBPP
Model	R1	R2	RL	R1	R2	RL	pass@1	pass@1
Base model	25.9 ± 0.09	6.4 ± 0.04	18.2 ± 0.03	38.5 ± 0.13	16.2 ± 0.08	24.8 ± 0.09	6.1 ± 1.13	0.4 ± 0.67
Full finetuned	25.8 ± 0.08	6.4 ± 0.04	18.1 ± 0.07	38.6 ± 0.13	16.2 ± 0.10	24.8 ± 0.08	4.9 ± 2.18	0.8 ± 0.68
LoRA	$\mathbf{26.3 \pm 0.08}$	$\mathbf{6.6 \pm 0.07}$	$\mathbf{18.5 \pm 0.04}$	$\mathbf{38.6 \pm 0.06}$	$\mathbf{16.3 \pm 0.09}$	$\mathbf{24.9 \pm 0.08}$	$\mathbf{8.6 \pm 2.22}$	$\mathbf{1.2 \pm 1.18}$
TOPOTUNER	26.0 ± 0.04	6.5 ± 0.02	18.3 ± 0.05	$\mathbf{38.6 \pm 0.16}$	$\mathbf{16.3 \pm 0.12}$	$\mathbf{24.9 \pm 0.09}$	7.3 ± 3.74	0.8 ± 0.68

	Instruction Following							
	IF:Dolly-15k				IF:Alpaca			
Model	R1	R2	RL	METEOR	R1	R2	RL	METEOR
Base model	36.8 ± 0.21	17.6 ± 0.16	27.5 ± 0.16	27.0	37.6 ± 0.22	17.9 ± 0.20	28.8 ± 0.13	29.2
Full finetuned	37.2 ± 0.29	18.0 ± 0.17	27.9 ± 0.20	27.5	37.3 ± 0.18	17.5 ± 0.15	28.5 ± 0.06	28.9
LoRA	$\mathbf{38.0 \pm 0.33}$	$\mathbf{18.7 \pm 0.14}$	$\mathbf{28.7 \pm 0.17}$	**28.0**	$\mathbf{38.3 \pm 0.22}$	$\mathbf{18.3 \pm 0.26}$	$\mathbf{29.4 \pm 0.17}$	**29.5**
TOPOTUNER	37.3 ± 0.24	18.1 ± 0.16	28.0 ± 0.14	27.3	37.8 ± 0.26	18.0 ± 0.31	29.0 ± 0.21	29.3

Table 16: Catastrophic forgetting evaluation on `QA:GSM8K` fine-tuned `Mistral-7B-v0.3`. Metrics are avg. ± std over three fixed, non-overlapping evaluation subsets. Base and full finetuned models are shown as reference rows. **Bold** indicates the better result between LoRA and TOPOTUNER for each metric.

	Question Answering	*Sentiment Analysis*		*Information Retrieval*			
	QA:MMLU	SA:SST-2	SA:IMDB	IR:SQuAD v1.1		IR:HotpotQA	
Model	Acc.	Acc.	Acc.	EM	F1	EM	F1
Base model	54.5 ± 0.41	89.0 ± 0.61	61.4 ± 0.48	71.3 ± 1.02	83.9 ± 0.72	23.6 ± 1.08	31.8 ± 0.80
Full finetuned	52.9 ± 0.33	89.0 ± 1.84	60.2 ± 0.50	73.2 ± 0.64	84.4 ± 0.54	21.6 ± 1.28	29.0 ± 0.83
LoRA	50.2 ± 0.79	70.6 ± 2.81	59.2 ± 0.56	69.8 ± 0.98	81.6 ± 0.80	18.7 ± 0.88	26.5 ± 0.66
TOPOTUNER	$\mathbf{53.2 \pm 0.33}$	$\mathbf{89.9 \pm 0.86}$	$\mathbf{60.6 \pm 0.56}$	$\mathbf{70.2 \pm 0.74}$	$\mathbf{83.0 \pm 0.63}$	$\mathbf{22.0 \pm 1.34}$	$\mathbf{29.9 \pm 0.86}$

	Summarization						*Code Generation*	
	SU:XSum			SU:CNN/DM			CG:HumanEval	CG:MBPP
Model	R1	R2	RL	R1	R2	RL	pass@1	pass@1
Base model	22.2 ± 0.10	6.2 ± 0.12	16.4 ± 0.17	39.5 ± 0.16	17.4 ± 0.14	25.0 ± 0.14	1.8 ± 1.82	3.5 ± 1.15
Full finetuned	22.2 ± 0.16	7.2 ± 0.11	16.9 ± 0.21	39.6 ± 0.18	17.5 ± 0.20	25.0 ± 0.14	1.8 ± 1.85	1.6 ± 1.77
LoRA	$\mathbf{29.2 \pm 0.23}$	$\mathbf{8.9 \pm 0.08}$	$\mathbf{22.4 \pm 0.24}$	36.5 ± 0.18	15.5 ± 0.22	23.6 ± 0.14	0.0 ± 0.00	0.4 ± 0.67
TOPOTUNER	27.7 ± 0.14	8.5 ± 0.15	20.6 ± 0.26	$\mathbf{39.8 \pm 0.21}$	$\mathbf{17.6 \pm 0.22}$	$\mathbf{25.1 \pm 0.16}$	$\mathbf{2.4 \pm 1.04}$	$\mathbf{1.9 \pm 1.34}$

	Instruction Following							
	IF:Dolly-15k				IF:Alpaca			
Model	R1	R2	RL	METEOR	R1	R2	RL	METEOR
Base model	28.7 ± 0.21	12.8 ± 0.05	21.7 ± 0.13	24.4	26.8 ± 0.23	11.3 ± 0.18	20.0 ± 0.15	24.2
Full finetuned	35.5 ± 0.13	17.3 ± 0.02	27.9 ± 0.07	26.7	32.8 ± 0.08	14.8 ± 0.04	25.4 ± 0.02	26.5
LoRA	$\mathbf{34.7 \pm 0.29}$	$\mathbf{17.1 \pm 0.15}$	$\mathbf{28.2 \pm 0.12}$	24.7	$\mathbf{35.0 \pm 0.14}$	$\mathbf{16.3 \pm 0.19}$	$\mathbf{28.3 \pm 0.10}$	**25.4**
TOPOTUNER	33.4 ± 0.35	16.2 ± 0.09	26.4 ± 0.17	**26.4**	29.1 ± 0.25	12.8 ± 0.15	22.3 ± 0.18	25.2

Qwen3-8B-Base shows a more mixed pattern. LoRA remains stronger on several generation-style metrics, especially summarization, instruction following, and code generation. However, TOPOTUNER stays very close to LoRA on most metrics and still improves over LoRA on SA:IMDB and IR:HotpotQA. Overall, these results suggest that selective fine-tuning reduces forgetting most clearly on question-answering, sentiment-analysis, and information-retrieval tasks, while its effect on generation-style evaluations depends on architecture.

Taken together, the experiments support three claims. First, fine-tuning changes are small in magnitude but structured in location, which makes naive layer-order freezing ineffective. Second, Wasserstein ranking and norm ranking each recover useful selective-tuning configurations, but neither signal alone induces a universally dominant ordering. Third, selective attention-projection tuning can preserve substantial target-task performance while reducing trainable parameters and retaining more out-of-domain capability than full fine-tuning. This combination of efficiency, retention, and task-geometry analysis is the central empirical message of the paper.

P OVERFITTING ANALYSIS

We analyze overfitting behavior across full fine-tuning, LoRA, TDA-High3, TDA-Low3, Eltwise-High3, and Eltwise-Low3 for LLaMA-3.1-8B, Table 17 reports the corresponding generalization gaps. For LLaMA-3.1-8B, full fine-tuning obtains the highest final test accuracy, but it also shows the largest train-test gap. In contrast, TDA-High3 provides the best selective test accuracy while yielding the smallest train-test gap. Overall, these results suggest that TOPOTUNER can act as a structural regularizer, preserving most of the useful adaptation while avoiding the stronger overfitting behavior observed in full fine-tuning.

Table 17: Overfitting analysis on LLaMA-3.1-8B across fine-tuning methods. Train-validation and train-test gaps are computed from epoch-6 accuracy values. **Bold** indicates the best result among LoRA, Eltwise, and TDA variants. Smaller gaps indicate better generalization stability.

Method	Train	Val	Test	Train-Val Gap	Train-Test Gap
LLaMA-3.1-8B					
Full FT	71.61	58.74	59.59	12.87	12.02
LoRA	62.50	**58.51**	57.32	**3.99**	5.18
Eltwise-High3	61.62	56.08	**58.07**	5.54	3.55
Eltwise-Low3	62.03	55.62	56.79	6.41	5.24
TDA-High3	61.15	56.53	**58.07**	4.62	**3.08**
TDA-Low3	**62.53**	57.14	57.32	5.39	5.21