Understanding Layer Patching in Model Size Interpolation

Anonymous Authors[1]

Abstract

Zero-shot model size interpolation aims to create new models of intermediate target sizes by combining existing models without additional training. Recent work on boomerang distillation (Kangaslahti et al., 2026) shows that a student language model distilled from a larger teacher can be expanded by iteratively *patching* its layers, replacing student layers with contiguous blocks of teacher layers to obtain models whose size and performance interpolate between the student and the teacher. Selecting which layers to patch for a given intermediate model size is a key design choice of this procedure, yet it has remained largely underexplored. In this work, we provide the first systematic study of student-layer selection for model size interpolation. We cast finding the optimal layer subset for each model size as an optimization problem and prove it can be viewed as a shortest-path problem in a certain acyclic graph. In experiments, we show that patching strongly shapes interpolation behavior, with effects that vary substantially across model families. We find that simple sequential strategies—patching either from the first layer to the last or from the last to the first—often achieve surprisingly strong performance in practice. We further introduce KLPatch, a greedy patching algorithm based on KL divergence, which often improves over last-to-first patching and approximately solves the optimization problem. Together, our results provide a principled understanding of how layer patching affects model size interpolation and offer practical guidance for constructing near-optimal interpolated models.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

1. Introduction

Creating LLM families with different model sizes is one of the most practical ways to adapt to users' varying compute constraints (Huyen, 2022; Khandelwal et al., 2025; Team et al., 2026). However, pre-training LLMs of different sizes requires separate training runs, which often results in the release of only a small number of coarse-grained, fixed-size LLMs (Grattafiori et al., 2024; Yang et al., 2025; Team, 2026). Recent work on boomerang distillation (Kangaslahti et al., 2026) has explored zero-shot model-size interpolation, enabling the creation of LLM families with fine-grained sizes. Boomerang distillation is a procedure in knowledge distillation in which layers of a student LLM can be patched with contiguous blocks of teacher layers to create intermediate-size models whose performance smoothly interpolates between that of the student and the teacher LLM, without requiring additional training. Boomerang distillation involves several critical choices, such as the student initialization, training token budget, alignment losses, and student patching. Among these, the role of student patching, a key user-facing decision with a combinatorial design space, remains poorly understood.

Prior work on boomerang distillation provides limited guidance on how to patch student models for optimal interpolation performance. Kangaslahti et al. (2026) consider only two strategies: patching from the first layer to the last and patching from the last layer to the first. Although they show that some LLMs prefer one order over the other, the broader design space of patching remains largely unexplored. Moreover, in Section 3, we show that naively increasing the size of an interpolated model by arbitrarily patching the student can degrade performance, contradicting the conventional view that larger models perform better (Sutton, 2019; Kaplan et al., 2020; Hoffmann et al., 2022). These observations motivate a deeper understanding of patching to identify a generalizable recipe for constructing interpolated models.

In this work, we provide a comprehensive understanding of student patching for model size interpolation. We define student patching as an optimization problem over interpolations (Section 2.2), including those that need not correspond to nested subsets of model layers, moving beyond the setup of Kangaslahti et al. (2026). As its computational complex-

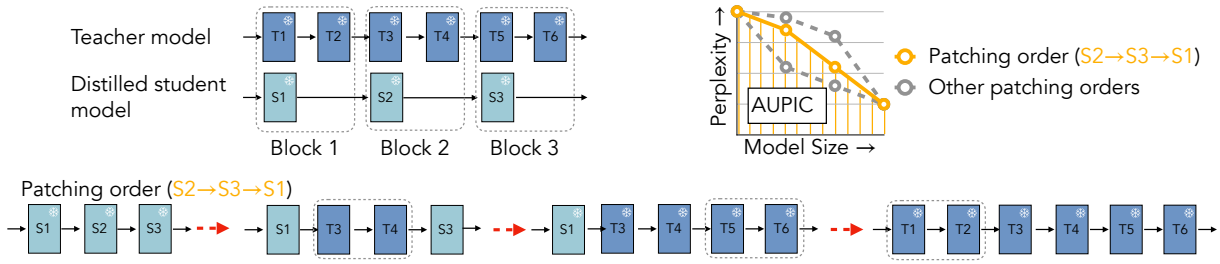

Figure 1. **Patching order in model size interpolation.** We show boomerang distillation with a six-layer teacher and a three-block distilled student, where each student block corresponds to a contiguous pair of two-layer teacher blocks. A patching order specifies the sequence in which student layers are replaced by their corresponding teacher blocks, inducing a trajectory of intermediate-size models from the student to the teacher. Here, the order $S2 \to S3 \to S1$ first replaces the middle student block, then the last block, and finally the first block. Different patching orders induce different interpolation curves, which we summarize using the area under the perplexity interpolation curve (AUPIC).

ity makes it infeasible to solve, we introduce an alternative shortest-path problem (Section 2.4) which we empirically and theoretically show approximates the optimal model interpolation. In experiments, we exhaustively search over all possible interpolations in small language models (Devlin et al., 2019; Radford et al., 2019) such as DistilBERT and DistilGPT to find the globally optimal model size interpolation (Section 3.1). Next, we extend our insights from small language models to larger models such as Qwen and Pythia (Biderman et al., 2023; Yang et al., 2025), and run an extensive study over 200 randomly sampled patching orders for each model (Section 3.2). Finally, we propose KLPatch (Section 4), a greedy KL divergence-based patching algorithm that approximately solves the shortest-path problem, reducing the computational complexity for finding performant interpolations from $O(2^N)$ to $O(N^2)$, where N is the number of layers in the student model.

Our experiments reveal valuable insights into how patching order drives model size interpolation. Our exhaustive search over all possible patching orders in DistilBERT and DistilGPT2 shows that, remarkably, patching from the last student layer to the first produces the globally optimal interpolation for DistilBERT and a top-5% interpolation for DistilGPT2. For LLMs, the patching recipe from Kangaslahti et al. (2026) is not always optimal or near-optimal but remains competitive, and shortest-path solutions tend to perform similarly to optimal interpolations. Finally, our experiments show that KLPatch often outperforms first-to-last and last-to-first patching and finds near-optimal interpolations across several language models, making it a compute-efficient method for creating highly performant model interpolations. Overall, our findings establish patching order as a key design choice for building interpolated models. **Our work makes the following contributions:**

- **Formalization of model interpolation.** We formalize student patching in boomerang distillation as an optimization problem over interpolation curves (Section 2.2), and derive an alternative shortest-path problem, which we empirically and theoretically justify (Sections 2.4, 3).
- **Phenomenology of patching order.** We provide the first thorough study of student patching. In particular, we perform exhaustive sweeps over *all* interpolations in DistilBERT and DistilGPT (Section 3.1), and 200 random patching orders on Qwen3 4B, Qwen3 8B, and Pythia 6.9B (Section 3.2), showing that simple last-to-first patching sometimes remains competitive, while optimal permutations obtained from solving the shortest-path problem closely approximate the optimal interpolation.
- **Practical patching algorithm.** We introduce KLPatch, an efficient greedy algorithm that approximately solves the interpolation optimization problem via an interpolation graph (Section 4.1), reducing the computational complexity from $O(2^N)$ to $O(N^2)$ and empirically yielding model interpolations close to the optimum (Section 4.2).

2. Layer Patching as a Shortest-Path Problem

Layer patching is an integral part of *boomerang distillation* (Kangaslahti et al., 2026), which we revisit in Sec. 2.1. We then formalize it as an optimization problem—first over subsets of student layers, then over permutations thereof—which converge algorithmically under greedy sequential construction (Sec. 2.2). We then introduce metrics for evaluating the resulting interpolations (Sec. 2.3) and recast the optimization as a shortest-path problem on a KL-weighted graph (Sec. 2.4), motivating the KLPatch algorithm (Sec. 4).

2.1. Layer Patching

Our work builds on *boomerang distillation* (Kangaslahti et al., 2026), a recent technique for model size interpolation. Starting from a teacher LLM and a smaller student distilled from it, boomerang distillation produces a chain of intermediate-size models that smoothly interpolate from the student to the teacher without any additional training (Figure 1). An integral design choice in boomerang distillation is *layer patching*, the process of choosing which student layers to keep and which to patch, using blocks of teacher layers. See Appendix F for more details.

2.2. A Combinatorial Optimization Perspective

Boomerang distillation requires choosing *which* student layers to patch with teacher blocks at each interpolated size, and we observe in Section 3 that this choice strongly determines downstream performance. We make this choice precise as a combinatorial optimization problem.

The choice of which layers to patch decomposes by interpolated size: for each cardinality $k \in \{1, ..., N\}$, there are $\binom{N}{k}$ candidate subsets $A \subseteq \{1, ..., N\}$, each inducing a partially patched model $\boldsymbol{M}_A$, and the best one depends on the chosen quality metric. Formally, fix an evaluation function $f_{\boldsymbol{S},\boldsymbol{T}} : 2^{\{1,...,N\}} \to \mathbb{R}$ that scores each subset A by the quality of $\boldsymbol{M}_A$, with higher meaning better.[1] The best k-subset then maximizes $f_{\boldsymbol{S},\boldsymbol{T}}(A)$ subject to $|A| = k$.

Problem 1 (Optimal Interpolation)**.** Given a student $\boldsymbol{S}$, a teacher $\boldsymbol{T}$, and an objective function $f_{\boldsymbol{S},\boldsymbol{T}} : 2^{\{1,...,N\}} \to \mathbb{R}$, the *optimal interpolation* between $\boldsymbol{S}$ and $\boldsymbol{T}$ with respect to $f_{\boldsymbol{S},\boldsymbol{T}}$ is the sequence of models $\boldsymbol{S} = \boldsymbol{M}_{A_0}, \boldsymbol{M}_{A_1}, ..., \boldsymbol{M}_{A_{N-1}}, \boldsymbol{M}_{A_N} = \boldsymbol{T}$ defined by

$$A_k = \operatorname*{arg\,max}_{A \subseteq \{1,...,N\},\, |A|=k} f_{\boldsymbol{S},\boldsymbol{T}}(A).$$

As $\binom{N}{k}$ subsets exist for each k, finding the optimal interpolation requires evaluating all $\sum_{k=1}^{N-1} \binom{N}{k} = 2^N - 2$ proper subsets of $\{1, \ldots, N\}$, which is infeasible in practice since each evaluation of $f_{\boldsymbol{S},\boldsymbol{T}}$ requires evaluating an interpolated model. An alternative formulation searches over permutations of layers rather than subsets, replacing the question "which k layers do we patch, for all k?" with "in what order do we patch all N layers?" A permutation of layers naturally gives rise to an interpolation of models by patching layers sequentially according to the permutation.

Problem 2 (Optimal Permutation)**.** With $\boldsymbol{S}$, $\boldsymbol{T}$, and $f_{\boldsymbol{S},\boldsymbol{T}}$ as before, the *optimal permutation* $\pi^* \in \mathfrak{S}_N$ of layers of $\boldsymbol{S}$ with respect to $f_{\boldsymbol{S},\boldsymbol{T}}$ is

$$\pi^* = \operatorname*{arg\,max}_{\pi \in \mathfrak{S}_N} \sum_{k=0}^{N} f_{\boldsymbol{S},\boldsymbol{T}}(A_k^\pi),$$

where $A_k^\pi \triangleq \{i : \pi(i) \leq k\}$.

The permutation search space is in fact larger ($|\mathfrak{S}_N| = N!$ versus $2^N - 2$), so Problem 2 is no easier *a priori* than Problem 1. But we will see in Section 2.4 that it admits a clean shortest-path interpretation with theoretical guarantees. More immediately, both formulations admit a natural greedy attack: grow the patched set one layer at a time, a strategy that underlies many tractable heuristics in combinatorial optimization, in some cases with formal approximation guarantees, e.g., for submodular maximization (Nemhauser et al., 1978). Applied to either Problem 1 or Problem 2, greedy addition produces the same object—a nested chain of subsets, equivalently a permutation—so the two formulations collapse algorithmically under this lens, though they differ in scoring: Problem 1 evaluates each A_k separately while Problem 2 aggregates across k. We instantiate this strategy in Section 4 (KLPatch). Empirically, we observe that solutions to Problem 2 closely track solutions to Problem 1 in terms of perplexity and accuracy (App. J.2).

[1] If lower is better, as in the case of perplexity, $f_{\boldsymbol{S},\boldsymbol{T}}$ is negated to recover a maximization problem.

2.3. Evaluating Model Size Interpolation Trajectories

Different patching orders trace out very different interpolation curves of model performance versus model size (e.g., Figure 2); to compare orders we need a single number summarizing how well an interpolated family fills the gap between $\boldsymbol{S}$ and $\boldsymbol{T}$. We use the *area under the interpolation curve* (over model size): a discrete sum of the evaluation function $f_{\boldsymbol{S},\boldsymbol{T}}$ over the $N+1$ interpolation points along a patching order π (Fig. 1). Two variants follow from the natural sign of the underlying metric: the **area under the interpolation curve** (**AUIC**) for higher-is-better metrics (e.g., accuracy) and the **area under the perplexity interpolation curve** (**AUPIC**) for lower-is-better metrics (perplexity). For a permutation $\pi \in \mathfrak{S}_N$ with intermediate models $\boldsymbol{M}_{A_k^\pi}$ and model size $|\boldsymbol{M}_{A_k^\pi}|$,

$$\Delta_k^\pi \triangleq |\boldsymbol{M}_{A_k^\pi}| - |\boldsymbol{M}_{A_{k-1}^\pi}|,$$

$$\mathrm{AUIC}(\pi) \triangleq \sum_{k=1}^{N} \Delta_k^\pi \frac{g_{\mathrm{data}}(\boldsymbol{M}_{A_k^\pi}) + g_{\mathrm{data}}(\boldsymbol{M}_{A_{k-1}^\pi})}{2},$$

$$\mathrm{AUPIC}(\pi) \triangleq \sum_{k=1}^{N} \Delta_k^\pi \frac{\mathrm{PPL}_{\mathrm{data}}(\boldsymbol{M}_{A_k^\pi}) + \mathrm{PPL}_{\mathrm{data}}(\boldsymbol{M}_{A_{k-1}^\pi})}{2}.$$

where g_{data} denotes any higher-is-better task metric (e.g., accuracy). A log analogue $\mathrm{AUPIC}^{\log}(\pi)$ which replaces a sum over perplexities $\mathrm{PPL}_{\mathrm{data}}$ with a sum over log-perplexities $\log \mathrm{PPL}_{\mathrm{data}}$ will be useful in Section 2.4. Replacing data-perplexity with perplexity against samples drawn from $\boldsymbol{T}$ yields teacher-relative variants $\mathrm{AUPIC}_{\boldsymbol{T}}(\pi)$ and $\mathrm{AUPIC}_{\boldsymbol{T}}^{\log}(\pi)$; these underpin the shortest-path characterization of Section 2.4 (Proposition 1).

2.4. Patching Orders as Shortest Paths

Layer patching is typically done incrementally: each step adds one previously unpatched layer to the current patched set. The reachable configurations therefore form a Boolean lattice on $\{1, \ldots, N\}$, and patching orders correspond to directed paths through it from $\boldsymbol{S}$ (no layers patched) to $\boldsymbol{T}$ (all layers patched). AUPIC and its log/teacher variants are themselves sums of a per-state evaluation along such a path, so Problem 2 reduces to a shortest-path problem on a suitably weighted version of the lattice. The remainder of this subsection formalizes this view and identifies the objective for which the reduction is exact.

Definition 1 (Interpolation Graph)**.** Let $\mathcal{G}$ be the Boolean lattice on $\{1, \ldots, N\}$, viewed as a directed acyclic graph: vertices are subsets $A \subseteq \{1, \ldots, N\}$, identified with the partially-patched models $\boldsymbol{M}_A$; an edge connects $\boldsymbol{M}_A$ to $\boldsymbol{M}_{A\cup\{i\}}$ whenever $i \notin A$. The graph has $\binom{N}{k}$ vertices at depth k, with root $\boldsymbol{M}_\emptyset = \boldsymbol{S}$ and sink $\boldsymbol{M}_{\{1,\ldots,N\}} = \boldsymbol{T}$. Given a calibration dataset $\mathcal{D}_{\text{cal}}$, assign each edge the weight

$$w_{A,i} \triangleq \mathbb{E}_{x\sim\mathcal{D}_{\text{cal}}} \mathrm{KL}\big(p_{\boldsymbol{T}}(\cdot \mid x) \,\big\|\, p_{\boldsymbol{M}_{A\cup\{i\}}}(\cdot \mid x)\big)\,.$$

Each patching order $\pi \in \mathfrak{S}_N$ corresponds to a directed path γ_π of length N from $\boldsymbol{S}$ to $\boldsymbol{T}$ in $\mathcal{G}$, traversing the intermediate models $\{\boldsymbol{M}_{A_k^\pi}\}_{k=0}^N$. Its total weight is

$$E(\pi) \triangleq \sum_{e\in\gamma_\pi} w(e) \;=\; \sum_{k=0}^{N} \mathbb{E}_{x\sim\mathcal{D}_{\text{cal}}} \mathrm{KL}\big(p_{\boldsymbol{T}}(\cdot \mid x) \,\big\|\, p_k^\pi(\cdot \mid x)\big),$$

where p_k^π denotes the distribution induced by $\boldsymbol{M}_{A_k^\pi}$. When interpolation sizes are equidistant, shortest paths in $\mathcal{G}$ are precisely the optimal permutations from Problem 2.

Proposition 1 (Shortest KL Paths are optimal)**.** *Assume that for any permutation π, consecutive model interpolation sizes are equidistant. Then the shortest paths in $\mathcal{G}$ are exactly the optimal permutations w. r. t.* $\mathrm{AUPIC}_{\boldsymbol{T}}^{\log}(\pi)$ *(to be understood as a set-equality if there are multiple minimizers):*

$$\arg\min_{\pi\in\mathfrak{S}_N} E(\pi) \;=\; \arg\min_{\pi\in\mathfrak{S}_N} \mathrm{AUPIC}_{\boldsymbol{T}}^{\log}(\pi).$$

Proposition 1 thus reduces an $N!$-way combinatorial search to a shortest-path problem on $\mathcal{G}$. Two assumptions distinguishing the proposition's setting from the metrics reported in our experiments—perplexity referenced to $\boldsymbol{T}$ rather than to data, and on the log scale rather than the raw scale—are discussed in Appendix G: Proposition 2 bounds the data-vs-teacher gap, and Proposition 3 the log-vs-raw gap. We also discuss the assumption that model sizes are equidistant in Remark 2. We note that in our experiments in Sections 3 and 4, we report a normalized version of the AUPIC; this amounts to rescaling by a constant factor and does not affect the optimality guarantee in Proposition 1. Empirically, on the models we study, path length in $\mathcal{G}$ strongly correlates with data-AUPIC (Appendix J.1, Figures 7 and 9). Section 4 exploits this characterization with a tractable greedy algorithm which finds paths in $\mathcal{G}$ that are often near-optimal.

While $\mathcal{G}$ could be populated with different choices of edge weights, such as perplexity or downstream accuracy, using KL to the teacher model provides several advantages: Proposition 1 draws a direct connection to the teacher-relative log-AUPIC, while computing the KL, unlike computing $\mathrm{AUPIC}_{\boldsymbol{T}}^{\log}$, does not require sampling from $\boldsymbol{T}$. Unlike downstream accuracy, KL is task-agnostic and provides a dense signal over the full predictive distribution. Finally, KL measures whether a patched model remains aligned to the teacher, unlike perplexity (however, we do find it strongly correlates with perplexity; see Appendix J.1, Figures 6 and 8). In practice, KL performs better than cosine distance, which correlates poorly with downstream performance (Zhang et al., 2024; Hinostroza et al., 2026).

3. Empirical Characterization of Layer Patching

Section 2 casts the choice of patching order as a combinatorial optimization with a shortest-path characterization. We now characterize this optimization landscape empirically: Section 3.1 performs a full sweep over all possible interpolations on DistilBERT and DistilGPT2, and Section 3.2 samples and evaluates a subset of patching orders for larger LLMs, where full enumeration is infeasible. We also investigate AUPIC relative to Spearman's footrule distance on permutations in Appendix I. Across both regimes, last-to-first patching is a consistently strong but not always optimal recipe, and the gap between naive baselines and the empirical optimum motivates the algorithm of Section 4.

3.1. Full Sweep on DistilBERT and DistilGPT2

We start with DistilBERT and DistilGPT2 because their small size ($N = 6$ layers each) makes the entire space of 720 patching orders—and all $2^6 = 64$ partially-patched models—directly enumerable, allowing us to characterize interpolation behavior exhaustively.

Setup. We use DistilBERT and DistilGPT2 (Sanh et al., 2019) as students, with the corresponding teacher checkpoints, BERT (Devlin et al., 2019) and GPT2 (Radford et al., 2019), all from Hugging Face. Both students have $N = 6$ layers, yielding $6! = 720$ patching orders each. For every order, we iteratively patch one layer at a time, evaluating pseudo-perplexity (DistilBERT) or perplexity (DistilGPT2) on Wikitext (Merity et al., 2017) at each of the $N+1$ interpolation points; aggregating across the trajectory gives AUPIC.

Results. Patching order has a substantial impact on interpolation performance (Figure 2, Appendix Figure 14). The best ordering yields near-linear interpolation between the student and teacher in pseudo-perplexity (DistilBERT) or perplexity (DistilGPT2), while the worst ordering produces only two interpolated models that improve on the student. The last-to-first ordering of Kangaslahti et al. (2026) attains optimal AUPIC on DistilBERT and near-optimal AUPIC on DistilGPT2, whereas first-to-last is less consistent: near-optimal on DistilGPT2 but above the mean on DistilBERT. The dotted line envelope in Figure 2 bounds the best and worst interpolation curves over all $2^6 = 64$ partially-patched models, including non-nested ones not reachable by any single patching order; gaps between this envelope and the best

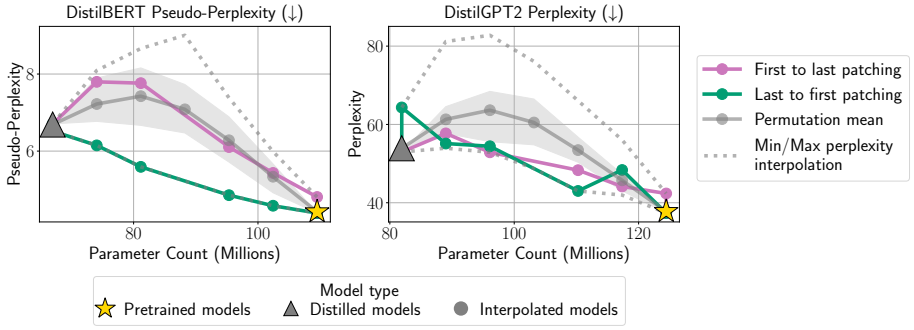

Figure 2. **All permutations of patching order for DistilBERT and DistilGPT2.** Patching order significantly affects interpolation performance, and the best ordering yields nearly linear interpolation in (pseudo-)perplexity between the students (DistilBERT, DistilGPT2) and teachers (BERT, GPT2). Shaded bands show the (25-75) inter-quartile range over all 720 orderings.

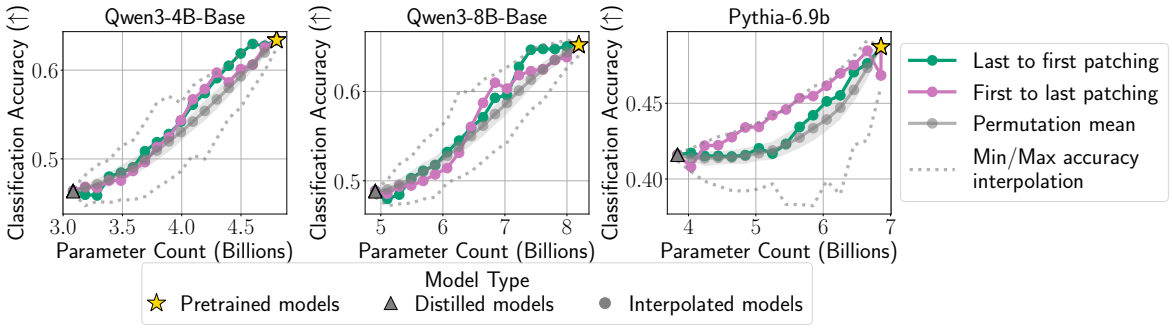

Figure 3. **Downstream interpolation curves across patching orders.** For each of Qwen3-4B, Qwen3-8B, and Pythia-6.9B, the mean downstream accuracy is plotted against patched model size, across the 200 sampled orderings together with the first-to-last and last-to-first baselines. Shaded bands show the (25-75) inter-quartile range for the 200 orderings. First-to-last remains competitive but does not always match the best sampled ordering.

ordering quantify the headroom available from relaxing the ordering constraint to general subset-based patching. The substantial gap between the optimum and the bulk of the AUPIC distribution, together with the inconsistency of fixed orderings across models, motivates a principled procedure for selecting patching orders, which we develop in Section 4.

3.2. Effect of Layer Patching in Large Language Models

We now ask whether the small-model findings of Section 3.1 carry over to larger language models. The full enumeration used there ($N! = 720$) is infeasible at this scale ($N! > 10^{12}$), so we instead sample 200 random patching orders per model and study the resulting AUIC distribution.

Setup. We use the distilled student models from Kangaslahti et al. (2026) for Qwen3-4B Base, Qwen3-8B Base, and Pythia-6.9B. For each of 200 randomly sampled patching orders, we iteratively patch one student layer at a time and evaluate downstream accuracy at every interpolation point, averaging across the task suite of Kangaslahti et al. (2026) to obtain an AUIC value per ordering (see Appendix H for details). We additionally include first-to-last and last-to-first as fixed-order baselines. Wikitext AUPIC results for the same models are reported in Appendix K.

Results. Across all three models, patching order has a substantial impact on downstream interpolation quality (Figure 3, Appendix Figure 15). The small-model regularity, however, does not transfer cleanly: while last-to-first attains the lowest *Wikitext* AUPIC (Appendix K), it leaves a substantial gap in downstream AUIC relative to the best-sampled ordering. This is consistent with prior observations that pretraining perplexity is an imperfect proxy for downstream performance (Liu et al., 2023). Moreover, the IQR across the 200 sampled orderings is narrow relative to this gap, so the lift available from finding a near-optimal order—rather than averaging over random ones—is large in practice; Section 4 develops a procedure for doing so.

4. KL-Guided Patching Orders

Section 3 showed that patching has a substantial impact on interpolation quality, and that first-to-last / last-to-first strategies leave a large headroom to the empirical optimum. We now turn this characterization into an algorithm.

4.1. The KLPatch Algorithm

KLPatch (Algorithm 1) is an iterative greedy procedure that, at each step, patches the student layer whose patched version minimizes KL divergence to the teacher on a calibration set—i.e., it greedily picks the cheapest outgoing edge in the interpolation graph $\mathcal{G}$ of Definition 1. It repeats until all N layers are patched. Since iteration k evaluates $N - k$ candi-

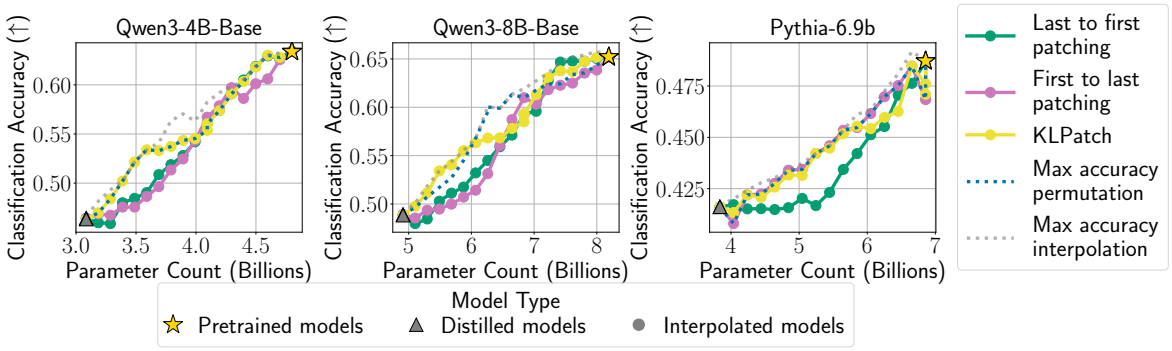

Figure 4. **KLPatch interpolation curves on Qwen3-4B, Qwen3-8B, and Pythia-6.9B.** For each model, mean downstream accuracy is plotted against the patched student size, comparing KLPatch (Algorithm 1) against first-to-last and last-to-first baselines. Across all three models, KLPatch tracks or outperforms last-to-first on both classification and generation tasks.

dates, KLPatch reduces the search from $O(N!)$ to $O(N^2)$ KL evaluations. Reading Problem 2 through Proposition 1, KLPatch is a greedy approximation to the shortest-path problem on $\mathcal{G}$. In empirical evaluations (Section 4.2), we show that it yields permutations close to the optima of both Problem 1 and Problem 2.

Algorithm 1 KLPatch

Require: Teacher model $\boldsymbol{T}$ with layer blocks $(\mathbf{b}^{(1)}, \ldots, \mathbf{b}^{(N)})$; trained student $\boldsymbol{S}$; calibration set $\mathcal{D}_{\text{cal}}$

$U \leftarrow \{1, \ldots, N\}$
$\boldsymbol{M} \leftarrow \boldsymbol{S}$
$\pi \leftarrow [\,]$
while $U \neq \emptyset$ **do**
  for each $i \in U$ **do**
    $\widetilde{\boldsymbol{M}}^{(i)} \leftarrow \textsc{Patch}(\boldsymbol{M}, i, \mathbf{b}^{(i)})$ `# Patch layer i`
    $\ell_i \leftarrow \mathbb{E}_{x \sim \mathcal{D}_{\text{cal}}} \left[\mathrm{KL}\big(p_{\boldsymbol{T}}(\cdot \mid x) \,\|\, p_{\widetilde{\boldsymbol{M}}^{(i)}}(\cdot \mid x)\big) \right]$
  end for
  $i^\star \leftarrow \arg\min_{i \in U} \ell_i$ `# Select lowest KL patch`
  $\boldsymbol{M} \leftarrow \textsc{Patch}(\boldsymbol{M}, i^\star, \mathbf{b}^{(i^\star)})$
  $\pi \leftarrow \pi \,\|\, [i^\star]$ `# Append i* to patching order`
  $U \leftarrow U \setminus \{i^\star\}$
end while
return π

4.2. KLPatch Finds Near-Optimal Patching Orders

Setup. We compare KLPatch's patching order against first-to-last / last-to-first baselines as well as the best interpolation and permutation obtained from 200 random orderings on the same models, students, and downstream task suite as in Section 3.2 (Qwen3-4B, Qwen3-8B, Pythia-6.9B), and against the distribution of 200 random orderings from that section. For KLPatch's calibration set $\mathcal{D}_{\text{cal}}$, we use 64 samples drawn from the Pile (Gao et al., 2021); the same calibration set is used across all student layers and all iterations of Algorithm 1. We additionally evaluate KLPatch on Llama-3.2-3B in Appendix M.

Results. KLPatch yields strong model interpolations across various families and tasks. On Qwen3-4B, Qwen3-8B, and Pythia-6.9B, KLPatch outperforms or matches both first-to-last and last-to-first on classification (Figure 4), generation (Appendix L.2), and Wikitext perplexity (Appendix L.3). It also beats all of the 200 random orderings for Qwen3-4B-Base and Pythia-6.9b and all but one of the random orderings for Qwen3-8B-Base (Table 4), yielding near-optimal interpolation trajectories on these three models. In Appendix M, we also evaluate KLPatch on Llama models, and show that while it does not work out-of-the-box, slightly modifying the algorithm by always patching layer 1 first recovers its strong performance. Overall, KLPatch provides an efficient approach for reliably creating high-performing model interpolations.

5. Conclusion

We presented a systematic study of layer patching in boomerang distillation. An exhaustive sweep over all interpolations on DistilBERT and DistilGPT2 shows that last-to-first is globally optimal or near-optimal, and that good orderings cluster together in permutation space. On Qwen3-4B, Qwen3-8B, and Pythia-6.9B, last-to-first remains competitive, but the gap to the empirical optimum is significant. To close this gap, we cast finding the optimal patching order as a shortest-path problem on a KL-weighted lattice (Proposition 1), and propose KLPatch, a greedy $O(N^2)$ algorithm that commits to the cheapest outgoing edge at each step; KLPatch produces the best or near-best patching orders and consistently outperforms first-to-last and last-to-first on Qwen and Pythia.

Yet, several questions remain open. The magnitude of KLPatch's gains varies across model families, and characterizing the conditions under which patching-order optimization helps most is a natural next step. Other directions include sub-greedy alternatives that exploit Proposition 1 more globally, amortized procedures that avoid recomputing teacher-conditioned KL at every step, and reinforcement-learning approaches in the spirit of He et al. (2018), which adaptively select interpolation trajectories rather than committing greedily.

References

Ainsworth, S. K., Hayase, J., and Srinivasa, S. S. Git re-basin: Merging models modulo permutation symmetries. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=CQsmMYmlP5T.

Biderman, S., Schoelkopf, H., Anthony, Q. G., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., Skowron, A., Sutawika, L., and van der Wal, O. Pythia: A suite for analyzing large language models across training and scaling. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 2397–2430. PMLR, 2023. URL https://proceedings.mlr.press/v202/biderman23a.html.

Bisk, Y., Zellers, R., LeBras, R., Gao, J., and Choi, Y. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 7432–7439. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/6239.

Cai, R., Muralidharan, S., Yin, H., Wang, Z., Kautz, J., and Molchanov, P. LLamaflex: Many-in-one LLMs via generalized pruning and weight sharing. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=AyC4uxx2HW.

Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., and Toutanova, K. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Burstein, J., Doran, C., and Solorio, T. (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., and Solorio, T. (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Entezari, R., Sedghi, H., Saukh, O., and Neyshabur, B. The role of permutation invariance in linear mode connectivity of neural networks. *arXiv preprint arXiv:2110.06296*, 2021.

Frankle, J., Dziugaite, G. K., Roy, D., and Carbin, M. Linear mode connectivity and the lottery ticket hypothesis. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3259–3269. PMLR, 2020. URL http://proceedings.mlr.press/v119/frankle20a.html.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., Presser, S., and Leahy, C. The pile: An 800gb dataset of diverse text for language modeling, 2021. URL https://arxiv.org/abs/2101.00027.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. A framework for few-shot language model evaluation, 2023. URL https://zenodo.org/records/10256836.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. The llama 3 herd of models. *ArXiv preprint*, abs/2407.21783, 2024. URL https://arxiv.org/abs/2407.21783.

Han, S., Pool, J., Tran, J., and Dally, W. J. Learning both weights and connections for efficient neural network. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28: Annual Conference on Neural*

Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada, pp. 1135–1143, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/ae0eb3eed39d2bcef4622b2499a05fe6-Abstract.html.

He, Y., Lin, J., Liu, Z., Wang, H., Li, L.-J., and Han, S. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 784–800, 2018.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021a. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021b.

Hinostroza, C., Icarte, R. T., Devia, C., Ferari, A. C. D., Herrera-Berg, E., Parra, D., and Silva, J. F. Rethinking layer relevance in large language models beyond cosine similarity. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=mRLnS8jQWt.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *ArXiv preprint*, abs/1503.02531, 2015. URL https://arxiv.org/abs/1503.02531.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D., Hendricks, L. A., Welbl, J., Clark, A., et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 10, 2022.

Hou, B., Chen, Q., Wang, J., Yin, G., Wang, C., Du, N., Pang, R., Chang, S., and Lei, T. Instruction-following pruning for large language models. *arXiv preprint arXiv:2501.02086*, 2025.

Huyen, C. *Designing machine learning systems*. " O'Reilly Media, Inc.", 2022.

Kangaslahti, S., Nayak, N. V., Geuter, J., Fumero, M., Locatello, F., and Alvarez-Melis, D. Boomerang distillation enables zero-shot model size interpolation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=4ZU8v4s3IR.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Khandelwal, A., Yun, T., Nayak, N. V., Merullo, J., Bach, S., Sun, C., and Pavlick, E. $100k or 100 days: Trade-offs when pre-training with academic resources. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=EFxC34XbDh.

Lai, G., Xie, Q., Liu, H., Yang, Y., and Hovy, E. RACE: Large-scale ReAding comprehension dataset from examinations. In Palmer, M., Hwa, R., and Riedel, S. (eds.), *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 785–794, Copenhagen, Denmark, 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1082. URL https://aclanthology.org/D17-1082.

LeCun, Y., Denker, J., and Solla, S. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

Liu, H., Xie, S. M., Li, Z., and Ma, T. Same pre-training loss, better downstream: Implicit bias matters for language models. 2023. doi: 10.48550/arXiv.2210.14199.

Mansourian, A. M., Ahmadi, R., Ghafouri, M., Babaei, A. M., Golezani, E. B., yasamani ghamchi, Z., Ramezanian, V., Taherian, A., Dinashi, K., Miri, A., and Kasaei, S. A comprehensive survey on knowledge distillation. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=3cbJzdR78B.

Men, X., Xu, M., Zhang, Q., Wang, B., Lin, H., Lu, Y., Han, X., and Chen, W. Shortgpt: Layers in large language models are more redundant than you expect. *ArXiv preprint*, abs/2403.03853, 2024. URL https://arxiv.org/abs/2403.03853.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Mihaylov, T., Clark, P., Khot, T., and Sabharwal, A. Can a suit of armor conduct electricity? a new dataset for open book question answering. In Riloff, E., Chiang, D., Hockenmaier, J., and Tsujii, J. (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260.

Muralidharan, S., Sreenivas, S. T., Joshi, R., Chochowski, M., Patwary, M., Shoeybi, M., Catanzaro, B., Kautz, J., and Molchanov, P. Compact language models via pruning and knowledge distillation. In Globersons, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J. M., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems 38: 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/4822991365c962105b1b95b1107d30e5-Abstract-Conference.html.

Nagarajan, V. and Kolter, J. Z. Uniform convergence may be unable to explain generalization in deep learning. *Advances in neural information processing systems*, 32, 2019.

Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L. An analysis of approximations for maximizing submodular set functions. 14:265–294, 1978. ISSN 0025-5610,1436-4646. doi: 10.1007/bf01588971.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 8024–8035, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. *OpenAI*, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 8732–8740. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/6399.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv preprint*, abs/1910.01108, 2019. URL https://arxiv.org/abs/1910.01108.

Singh, S. P. and Jaggi, M. Model fusion via optimal transport. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/fb2697869f56484404c8ceee2985b01d-Abstract.html.

Sreenivas, S. T., Muralidharan, S., Joshi, R., Chochowski, M., Mahabaleshwarkar, A. S., Shen, G., Zeng, J., Chen, Z., Suhara, Y., Diao, S., et al. Llm pruning and distillation in practice: The minitron approach. *ArXiv preprint*, abs/2408.11796, 2024. URL https://arxiv.org/abs/2408.11796.

Sun, M., Liu, Z., Bair, A., and Kolter, J. Z. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=PxoFut3dWW.

Sutton, R. The bitter lesson. 2019.

Team, K., Bai, T., Bai, Y., Bao, Y., Cai, S., Cao, Y., Charles, Y., Che, H., Chen, C., Chen, G., et al. Kimi k2. 5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

Team, Q. Qwen3.5: Accelerating productivity with native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. GLUE: A multi-task benchmark and analysis platform for natural language understanding. 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=rJ4km2R5t7.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., and Brew, J. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv preprint*, abs/1910.03771, 2019. URL https://arxiv.org/abs/1910.03771.

Xia, M., Gao, T., Zeng, Z., and Chen, D. Sheared llama: Accelerating language model pre-training via structured pruning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=09iOdaeOzp.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. Qwen3 technical report. *ArXiv preprint*, abs/2505.09388, 2025. URL https://arxiv.org/abs/2505.09388.

Yang, E., Shen, L., Guo, G., Wang, X., Cao, X., Zhang, J., and Tao, D. Model merging in llms, mllms, and beyond: Methods, theories, applications, and opportunities. *ACM Comput. Surv.*, 58(8), February 2026. ISSN 0360-0300. doi: 10.1145/3787849. URL https://doi.org/10.1145/3787849.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. HellaSwag: Can a machine really finish your sentence? In Korhonen, A., Traum, D., and Màrquez, L. (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472.

Zhang, Y., Li, Y., Wang, X., Shen, Q., Plank, B., Bischl, B., Rezaei, M., and Kawaguchi, K. Finercut: Finer-grained interpretable layer pruning for large language models. *arXiv preprint arXiv:2405.18218*, 2024.

Zhou, J., Lu, T., Mishra, S., Brahma, S., Basu, S., Luan, Y., Zhou, D., and Hou, L. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

A. Related Work

Model Interpolation Model interpolation is a technique where the weights of multiple models are combined to produce new capabilities without any additional training (Frankle et al., 2020; Singh & Jaggi, 2020; Ainsworth et al., 2023; Yang et al., 2026). Several works (Entezari et al., 2021; Nagarajan & Kolter, 2019) show that when two neural networks are initialized identically and share a part of the optimization trajectories, their *modes* are linearly connected, enabling linear interpolation between them. However, these explanations are limited to model interpolation of the same size. In contrast, we aim to understand interpolation between models of different sizes. Recent work has explored zero-shot model size interpolation, which aims to interpolate between models of different sizes (Cai et al., 2025; Kangaslahti et al., 2026). Cai et al. (2025) show that explicitly training a Gumbel Softmax-based router allows them to interpolate between models of different sizes. More recently, Kangaslahti et al. (2026) find that we can achieve zero-shot model size interpolation without training additional routers via knowledge distillation (Section **??**). We extend their work by comprehensively examining the user-facing choice of patching and its effects on model size interpolation.

Pruning Model pruning aims to compress model parameters by removing redundant parameters to improve inference efficiency while maintaining the full model performance (LeCun et al., 1989; Han et al., 2015; Sun et al., 2024; Xia et al., 2024; Sreenivas et al., 2024; Hou et al., 2025). Analogous to student patching order is a model pruning technique called layer dropping, where the goal is to drop layers based on a computationally efficient criterion, such as activation similarity (Men et al., 2024; Hinostroza et al., 2026) and statistical distance (Zhang et al., 2024), so that the drop in performance is minimal. Our greedy KL patching algorithm (see Algorithm 1) closely relates to the iterative layer pruning algorithm in Zhang et al. (2024). Although both rely on KL divergence (or a variant) between the larger and smaller models' token distributions to assess layer importance, Zhang et al. (2024) uses this information to remove layers, whereas we use it to patch layers and thereby construct interpolated models. Furthermore, in Section 4, we show that the greedy KL divergence algorithm for patching often recovers a near-optimal patching trajectory.

Knowledge Distillation. Knowledge distillation is a well-known technique to create smaller models that mimic the behavior of larger models (Hinton et al., 2015; Sanh et al., 2019; Muralidharan et al., 2024; Mansourian et al., 2025). Prior work has used knowledge distillation to create compact language models such as DistilBERT (Sanh et al., 2019) by training them on a corpus of text using the knowledge distillation objective (Hinton et al., 2015). Kangaslahti et al. (2026) showed that these distilled models exhibit zero-shot model size interpolation without any additional training. We build on their initial experiment by creating interpolated models using DistilBERT and DistilGPT and patching the student in all possible ways to identify the optimal patching order.

B. Limitations

In this work, we use student models distilled from Kangaslahti et al. (2026). Although this saves a significant amount of compute, we inherit the limitations of the distilled models, which could potentially influence the patching order. For example, we observe that KLPatch, out of the box, does not work well for Llama, as it significantly underperforms sequential patching order baselines. We suspect this may be due to differences in the initialization and training of the distilled student model relative to the other distilled models. See Appendix M for more details.

C. Reproducibility Statement

We conduct all experiments using PyTorch (Paszke et al., 2019) and the Hugging Face Transformers library (Wolf et al., 2019). We also use the publicly available codebase and distilled student models from (Kangaslahti et al., 2026) and use pretrained models from the Hugging Face Hub. The code is available at `https://anonymous.4open.science/r/understanding_patching_order-23C6`.

D. Broader Impacts

We use the distilled student models from Kangaslahti et al. (2026) and may therefore inherit biases or limitations present in these models. Before deployment, we recommend carefully evaluating the models on the intended target tasks and conducting appropriate post-training and safety evaluations.

E. Statement on Use of Large Language Models

We utilized generative AI tools for code completion, debugging, and minor grammatical corrections in the manuscript. The authors carried out all the substantive research contributions, analyses, and interpretations.

F. Boomerang Distillation

Here we provide more details on boomerang distillation (Kangaslahti et al., 2026).

Notation. Let the teacher $\boldsymbol{T}$ have M transformer layers and the student $\boldsymbol{S}$ from the same family have $N < M$ layers. Both models consist of an embedding layer, a sequence of transformer blocks, and an LM head, and all blocks produce hidden states of the same dimension.

Step 1: Student Initialization. The student is initialized by partitioning the teacher's M transformer layers into N contiguous blocks; each student block is then initialized from the first teacher layer in the corresponding block (Kangaslahti et al., 2026). The student's embedding layer and LM head are copied from the teacher.

Step 2: Knowledge Distillation and Alignment. The initialized student is trained on a corpus such as the Pile (Gao et al., 2021) using a combination of knowledge distillation (Hinton et al., 2015) and a block-level alignment loss such as the cosine distance loss (Sanh et al., 2019), which encourages each student block to mimic the hidden state at the boundary of its corresponding teacher block. Concretely, given a token sequence $x = (x_1, \ldots, x_L)$ and a position j, let $p_{\boldsymbol{T}}(\cdot \mid x_{<j})$ and $p_{\boldsymbol{S}}(\cdot \mid x_{<j})$ denote the teacher's and student's next-token distributions. The per-token loss is

$$\mathcal{L}_{\boldsymbol{\theta_S}}(x_j) = \mathrm{CE}(x_j \mid x_{<j}; \boldsymbol{\theta_S}) + \lambda_{\mathrm{KL}}\,\mathrm{KL}(p_{\boldsymbol{T}}(\cdot \mid x_{<j}) \,\|\, p_{\boldsymbol{S}}(\cdot \mid x_{<j})) + \lambda_{\cos} \sum_{i=1}^{N} \mathcal{L}^{(i)}_{\cos}(x_{<j}; \boldsymbol{\theta_T}, \boldsymbol{\theta_S}).$$

where CE is the cross-entropy loss against the ground-truth next token, $\mathcal{L}^{(i)}_{\cos}$ is the cosine distance between the student's hidden state at the boundary of block i and the corresponding teacher hidden state, and $\lambda_{\mathrm{KL}}, \lambda_{\cos} > 0$ weigh the distillation and alignment terms.

Step 3: Student Patching. After training, intermediate-size models are constructed by *patching*—replacing student blocks with the corresponding teacher blocks. For each subset $A \subseteq \{1, \ldots, N\}$, we write $\boldsymbol{M}_A$ for the model obtained by patching in the teacher blocks aligned with the indices in A, leaving the remaining $N - |A|$ student blocks intact. The endpoints recover the student and teacher, $\boldsymbol{S} = \boldsymbol{M}_\emptyset$ and $\boldsymbol{T} = \boldsymbol{M}_{\{1,\ldots,N\}}$, and growing A one block at a time produces a chain of intermediate models linking the two without any additional training (Figure 1). The order in which student blocks are patched determines the trajectory through this chain, and is the central object of study in the remainder of the paper.

G. Proofs

Proposition 1 (Shortest KL Paths are optimal)**.** *Assume that for any permutation π, consecutive model interpolation sizes are equidistant; that is, there is a constant $c \in \mathbb{N}$ such that*

$$|\boldsymbol{M}^{\pi}_{k}| - |\boldsymbol{M}^{\pi}_{k-1}| = c$$

for any $\pi \in \mathfrak{S}_N$ and $k = 1, ..., N$, where $|M|$ denotes the number of parameters of M. Then the shortest paths in $\mathcal{G}$ are exactly the optimal permutations with respect to $\mathrm{AUPIC}^{\log}_{\boldsymbol{T}}(\pi)$ (to be understood as a set-equality if there are multiple minimizers):

$$\arg\min_{\pi \in \mathfrak{S}_N} E(\pi) = \arg\min_{\pi \in \mathfrak{S}_N} \mathrm{AUPIC}^{\log}_{\boldsymbol{T}}(\pi).$$

Proof. For any intermediate model $\boldsymbol{M}$, the cross-entropy decomposition gives

$$H(p_{\boldsymbol{T}}(\cdot \mid x), p_{\boldsymbol{M}}(\cdot \mid x)) = H(p_{\boldsymbol{T}}(\cdot \mid x)) + \mathrm{KL}\left(p_{\boldsymbol{T}}(\cdot \mid x) \,\|\, p_{\boldsymbol{M}}(\cdot \mid x)\right).$$

Averaging over $x \sim \mathcal{D}_{\mathrm{cal}}$ and rearranging, we obtain

$$\mathbb{E}_{x \sim \mathcal{D}_{\mathrm{cal}}} \mathrm{KL}\left(p_{\boldsymbol{T}}(\cdot \mid x) \,\|\, p_{\boldsymbol{M}}(\cdot \mid x)\right) = \log \mathrm{PPL}_{\boldsymbol{T}}(\boldsymbol{M}) - \log \mathrm{PPL}_{\boldsymbol{T}}(\boldsymbol{T}).$$

Applying this identity to each intermediate model $\boldsymbol{M}_k^\pi$ and summing over $k = 0, \ldots, N$,

$$\begin{aligned} E(\pi) &= \sum_{k=0}^{N} \mathbb{E}_{x\sim\mathcal{D}_{\text{cal}}} \mathrm{KL}\left(p_{\boldsymbol{T}}(\cdot \mid x) \,\middle\|\, p_{\boldsymbol{M}_k^\pi}(\cdot \mid x)\right) \\ &= \sum_{k=0}^{N} \log \mathrm{PPL}_{\boldsymbol{T}}(\boldsymbol{M}_k^\pi) - (N+1) \log \mathrm{PPL}_{\boldsymbol{T}}(\boldsymbol{T}) \\ &= \frac{1}{c}\mathrm{AUPIC}_{\boldsymbol{T}}^{\log}(\pi) + \frac{1}{2}\left(\log \mathrm{PPL}_{\boldsymbol{T}}(\boldsymbol{S}) + \log \mathrm{PPL}_{\boldsymbol{T}}(\boldsymbol{T})\right) - (N+1) \log \mathrm{PPL}_{\boldsymbol{T}}(\boldsymbol{T}), \end{aligned}$$

using the fact that $\boldsymbol{M}_0^\pi = \boldsymbol{S}$ and $\boldsymbol{M}_N^\pi = \boldsymbol{T}$. The final two terms are independent of π, so $E(\pi)$ and $\mathrm{AUPIC}_T^{\log}(\pi)$ have the same minimizers. □

Proposition 2 (Reference Distribution)**.** *If*

$$|\log \mathrm{PPL}_{\text{data}}(\boldsymbol{M}_k^\pi) - \log \mathrm{PPL}_{\boldsymbol{T}}(\boldsymbol{M}_k^\pi)| \le \eta$$

for all evaluated $k = 0, \ldots, N$, then

$$\left|\mathrm{AUPIC}_{\boldsymbol{T}}^{\log}(\pi) - \mathrm{AUPIC}^{\log}(\pi)\right| \le (N+1)\eta,$$

and

$$e^{-\eta}\mathrm{AUPIC}_{\boldsymbol{T}}(\pi) \le \mathrm{AUPIC}(\pi) \le e^{\eta}\mathrm{AUPIC}_{\boldsymbol{T}}(\pi).$$

Equality holds in all three inequalities when $p_{\boldsymbol{T}} = p_{\text{data}}$ on $\mathcal{D}_{\text{cal}}$.

Proof. For each k, define

$$a_k := \log \mathrm{PPL}_{\text{data}}(\boldsymbol{M}_k^\pi), \qquad b_k := \log \mathrm{PPL}_{\boldsymbol{T}}(\boldsymbol{M}_k^\pi).$$

By assumption, $|a_k - b_k| \le \eta$ for all $k = 0, \ldots, N$. Therefore, by the triangle inequality,

$$\left|\mathrm{AUPIC}^{\log}(\pi) - \mathrm{AUPIC}_{\boldsymbol{T}}^{\log}(\pi)\right| = \left|\sum_{k=0}^{N} a_k - \sum_{k=0}^{N} b_k\right| \le \sum_{k=0}^{N} |a_k - b_k| \le (N+1)\eta.$$

For the raw-scale statement, the same assumption implies

$$b_k - \eta \le a_k \le b_k + \eta.$$

Exponentiating gives

$$e^{-\eta} e^{b_k} \le e^{a_k} \le e^{\eta} e^{b_k}.$$

Since

$$e^{a_k} = \mathrm{PPL}_{\text{data}}(\boldsymbol{M}_k^\pi), \qquad e^{b_k} = \mathrm{PPL}_{\boldsymbol{T}}(\boldsymbol{M}_k^\pi),$$

summing over $k = 0, \ldots, N$ yields

$$e^{-\eta} \sum_{k=0}^{N} \mathrm{PPL}_{\boldsymbol{T}}(\boldsymbol{M}_k^\pi) \le \sum_{k=0}^{N} \mathrm{PPL}_{\text{data}}(\boldsymbol{M}_k^\pi) \le e^{\eta} \sum_{k=0}^{N} \mathrm{PPL}_{\boldsymbol{T}}(\boldsymbol{M}_k^\pi).$$

Equivalently,

$$e^{-\eta}\mathrm{AUPIC}_{\boldsymbol{T}}(\pi) \le \mathrm{AUPIC}(\pi) \le e^{\eta}\mathrm{AUPIC}_{\boldsymbol{T}}(\pi).$$

If $p_{\boldsymbol{T}} = p_{\text{data}}$ on $\mathcal{D}_{\text{cal}}$, then

$$\log \mathrm{PPL}_{\text{data}}(\boldsymbol{M}_k^\pi) = \log \mathrm{PPL}_{\boldsymbol{T}}(\boldsymbol{M}_k^\pi)$$

for every k, so the above bounds hold with $\eta = 0$, giving equality. □

Proposition 3 (Log Scale versus raw Scale)**.** *We have*

$$(N+1)\exp\left(\frac{1}{N+1}\mathrm{AUPIC}^{\log}(\pi)\right) \le \mathrm{AUPIC}(\pi),$$

with equality if and only if $\log \mathrm{PPL}_{\mathrm{data}}(\boldsymbol{M}_k^{\pi})$ *is constant in* k*. The same statement is true for* $\mathrm{AUPIC}_{\boldsymbol{T}}^{\log}$ *and* $\mathrm{AUPIC}_{\boldsymbol{T}}$ *in place of* $\mathrm{AUPIC}^{\log}$ *and* AUPIC*.*

Proof. Let

$$z_k := \log \mathrm{PPL}_{\mathrm{data}}(\boldsymbol{M}_k^{\pi}), \qquad k = 0, \dots, N.$$

Then

$$\mathrm{AUPIC}^{\log}(\pi) = \sum_{k=0}^{N} z_k, \qquad \mathrm{AUPIC}(\pi) = \sum_{k=0}^{N} e^{z_k}.$$

Since the exponential function is convex, Jensen's inequality gives

$$\exp\left(\frac{1}{N+1}\sum_{k=0}^{N} z_k\right) \le \frac{1}{N+1}\sum_{k=0}^{N} e^{z_k}.$$

Multiplying both sides by $N+1$, we obtain

$$(N+1)\exp\left(\frac{1}{N+1}\mathrm{AUPIC}^{\log}(\pi)\right) \le \mathrm{AUPIC}(\pi).$$

By strict convexity of the exponential function, equality holds if and only if

$$z_0 = z_1 = \cdots = z_N,$$

that is, if and only if $\log \mathrm{PPL}_{\mathrm{data}}(\boldsymbol{M}_k^{\pi})$ is constant in k.

The same Jensen inequality and equality condition also hold with $\mathrm{AUPIC}_{\boldsymbol{T}}^{\log}$ and $\mathrm{AUPIC}_{\boldsymbol{T}}$ in place of $\mathrm{AUPIC}^{\log}$ and AUPIC. □

Remark 1 (Log vs raw AUPIC). Log and raw AUPIC have the same monotone direction but generally distinct argmins, and can disagree when one ordering has a single anomalously bad intermediate model whose raw perplexity dominates the sum. The map $z \mapsto e^z$ is strictly increasing, so if one ordering has no larger log-perplexity than another ordering at every interpolation point, then it also has no larger raw perplexity at every interpolation point. In this pointwise sense, the log and raw objectives have the same monotone direction.

However, the two summed objectives need not have the same minimizer. The log objective sums the z_k, while the raw objective sums e^{z_k}, so the raw objective penalizes isolated large values much more heavily. For example, consider two candidate interpolation curves with the same endpoints and intermediate log-perplexities

$$(z_0, z_1, z_2, z_3) = (0, 0, 6, 0), \qquad (z_0', z_1', z_2', z_3') = (0, 3.1, 3.1, 0).$$

Then

$$\sum_{k=0}^{3} z_k = 6 < 6.2 = \sum_{k=0}^{3} z_k',$$

so the first curve is preferred on the log scale. But

$$\sum_{k=0}^{3} e^{z_k} = 3 + e^6 > 2 + 2e^{3.1} = \sum_{k=0}^{3} e^{z_k'},$$

so the second curve is preferred on the raw scale. Thus log-scale and raw-scale AUPIC can have distinct argmins.

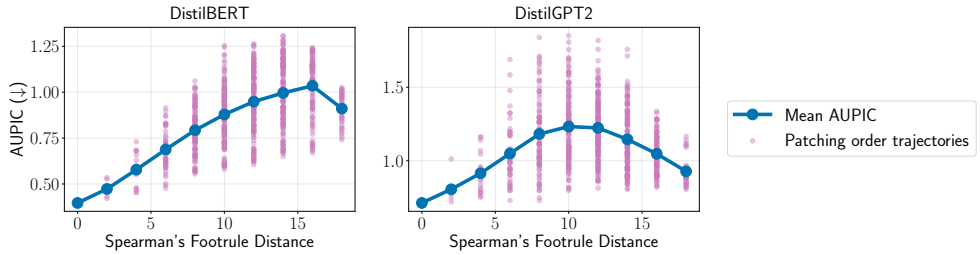

Figure 5. **AUPIC vs. Spearman footrule distance from the minimum-AUPIC permutation.** On both DistilBERT and DistilGPT2, mean AUPIC rises with distance from the optimum and then flattens, indicating a cluster of near-optimal orderings around the empirical minimum.

Remark 2 (Equidistant Model Sizes Assumption). Proposition 1 hinges on the assumption that for any permutation, consecutive model sizes are equidistant. This assumption holds in settings where all teacher blocks have the same size, which can be achieved by initializing the student from every kth teacher layer. This is almost exactly what (Kangaslahti et al., 2026) do in practice; however, as is standard practice since models such as DistilBERT and DistilGPT, they also ensure the first and last layer are kept, which results in one additional layer over the simple "every other layer" approach.[2]

H. Experimental Details

Models. We use the openly available distilled models released by Kangaslahti et al. (2026) for Qwen, Pythia, and Llama, and the DistilBERT and DistilGPT models released by Sanh et al. (2019) on Hugging Face. All the interpolated models are created without any additional training.

Evaluation Datasets. We use the same evaluation datasets as Kangaslahti et al. (2026) throughout the paper. We report the classification accuracy on these ten tasks: ARC-easy and ARC-challenge (Clark et al., 2018), BoolQ (Clark et al., 2019), HellaSwag (Zellers et al., 2019), OpenBookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), WinoGrande (Sakaguchi et al., 2020), RACE (Lai et al., 2017), MMLU (Hendrycks et al., 2021a), and RTE (Wang et al., 2019). For generation tasks, we report the exact match on the following datasets: GSM8K (Cobbe et al., 2021), IFEval (Zhou et al., 2023), and MATH (Hendrycks et al., 2021b). We report perplexity and pseudo-perplexity on the Wikitext dataset (Merity et al., 2017). We use `lm-evaluation-harness` (Gao et al., 2023) to evaluate all of the models used in our experiments.

Hardware and Walltime. For the DistilBERT and DistilGPT interpolation experiments, we use a single NVIDIA A100 40GB GPU, whereas for the LLM interpolation experiments, we use a single NVIDIA H100 80GB GPU to create and evaluate the intermediate models. Evaluating a single intermediate model in a trajectory is completed in under an hour.

I. Good Patching Orders Cluster Around the Optimum

Section 3.1 characterized the spread of AUPIC across all orderings. Here we ask a finer question: are good orderings *near* the optimum in permutation space, or are they scattered? A clean structural answer would say that small perturbations of the optimum remain near-optimal, suggesting a smooth landscape amenable to local search.

Setup. We use the 720 trajectories from Section 3.1 for both DistilBERT and DistilGPT2 and measure each ordering's Spearman footrule distance to the empirical optimum. For permutations $\pi, \pi' \in \mathfrak{S}_N$, the footrule distance is computed as $d(\pi, \pi') = \sum_{i=1}^{N} \left|\pi(i) - \pi'(i)\right|$, i.e., the ℓ_1 distance between the position vectors. We then bin orderings by their distance to the optimum and report the mean AUPIC per bin.

Results. Mean AUPIC increases with footrule distance from the optimum and then plateaus (Figure 5). Orderings within small footrule distance to the optimum form a cluster of near-optimal AUPIC, while at larger distances the relationship becomes noisy and AUPIC saturates. In the language of Section 2.4, near-optimal patching orders correspond to short paths in $\mathcal{G}$ close to the shortest path; this local structure is what makes greedy approaches such as KLPatch (Section 4) viable: small, locally-informed deviations from the shortest-path skeleton tend not to leave the near-optimal basin.

[2]Which additional layer is kept varies by model family; for Qwen and Pythia, they keep layers $1, 3, 5, \ldots, N-1, N$, while for Llama, they keep $1, 2, 4, 6, \ldots, N$; for every model used, the number of layers N is even.

J. Empirically Validating Theoretical Assumptions

In this section, we empirically validate the assumptions made in Section 2 and Appendix G.

J.1. KL Divergence Strongly Correlates with Perplexity

In Proposition 1, we optimize for the perplexity with respect to the teacher distribution rather than the data distribution. In this section, we show that optimizing w.r.t. the teacher distribution is reasonable by demonstrating a strong correlation between the KL divergence to the teacher model and downstream perplexity.

Setup. We use the exhaustive set of DistilBERT and DistilGPT2 patching orders from Section 3.1 to study the relationship between KL divergence and data perplexity. We evaluate KL divergence on the Wikitext test set (Merity et al., 2017) and on a calibration set of 64 samples from the Pile (Gao et al., 2021), and evaluate perplexity on the Wikitext test set.

Results. When KL divergence and perplexity are evaluated on the same dataset (Wikitext), there is a strong correlation (Pearson r =0.981 for DistilBERT and Pearson r=0.955 for DistilGPT2) between KL divergence and perplexity (Figure 6). This indicates that distributions of interpolated models that are similar to the distribution of the teacher are also similar to the data distribution. Figure 7 further shows that KL path length is strongly correlated to AUPIC (Pearson r=0.958 for DistilBERT and Pearson r=0.903 for DistilGPT2), so shorter interpolation paths in $\mathcal{G}$ tend to have better performance in terms of AUPIC. These results also hold when the KL divergence is computed on a different dataset than the perplexity. Figure 8 demonstrates that KL divergence on the Pile calibration set correlates with Wikitext perplexity, and Figure 9 shows that KL path length computed on the Pile calibration set is correlated with AUPIC.

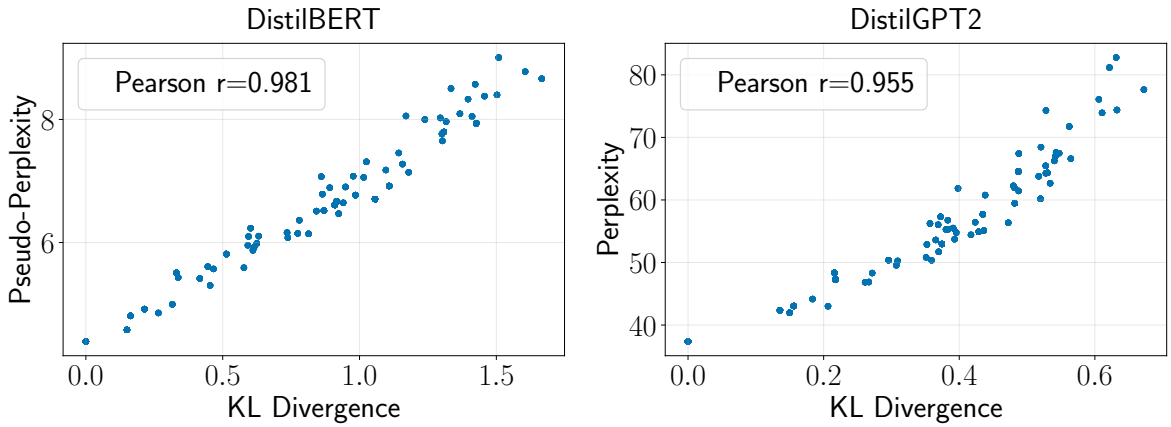

Figure 6. **KL divergence on Wikitext versus Wikitext perplexity.** When KL divergence is computed on Wikitext, there is a strong correlation between the KL divergence between interpolated models and the teacher and Wikitext perplexity for DistilBERT and DistilGPT2 models.

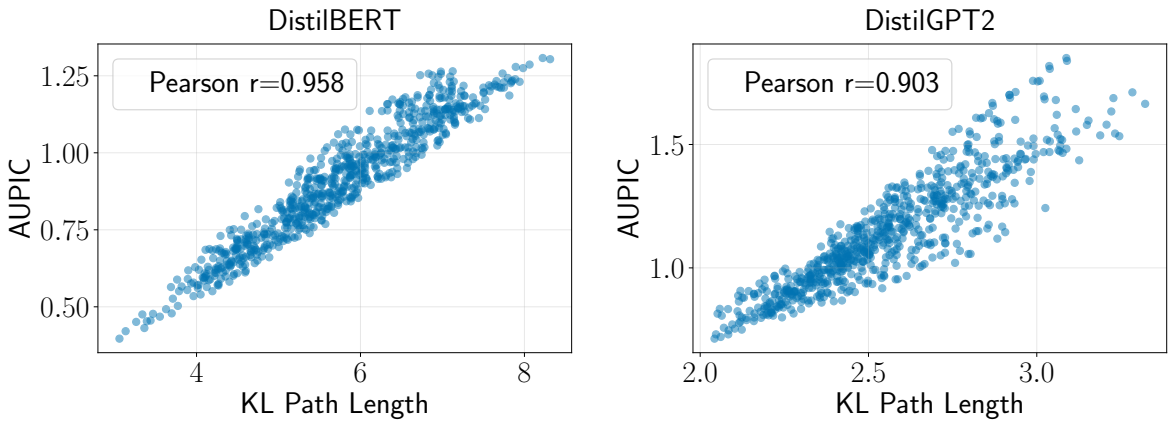

Figure 7. **KL Path Length on Wikitext versus Wikitext AUPIC.** For all possible patching orders, when KL divergence and perplexity are both computed on Wikitext, the KL path length in the interpolation graph correlates strongly with AUPIC.

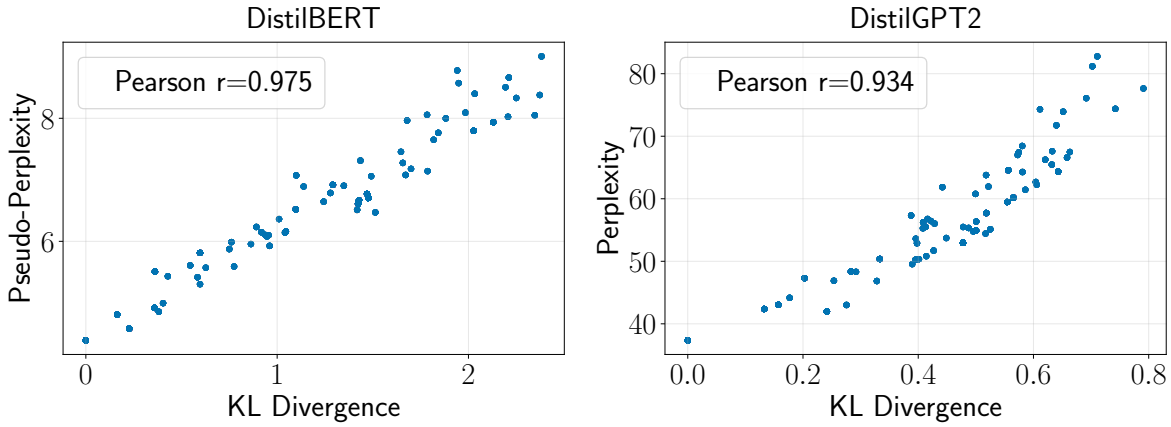

Figure 8. **KL divergence on the Pile versus Wikitext perplexity.** When KL divergence is computed on a 64 example calibration set from the Pile, there is a strong correlation between the KL divergence between interpolated models and the teacher and Wikitext perplexity for DistilBERT and DistilGPT2 models.

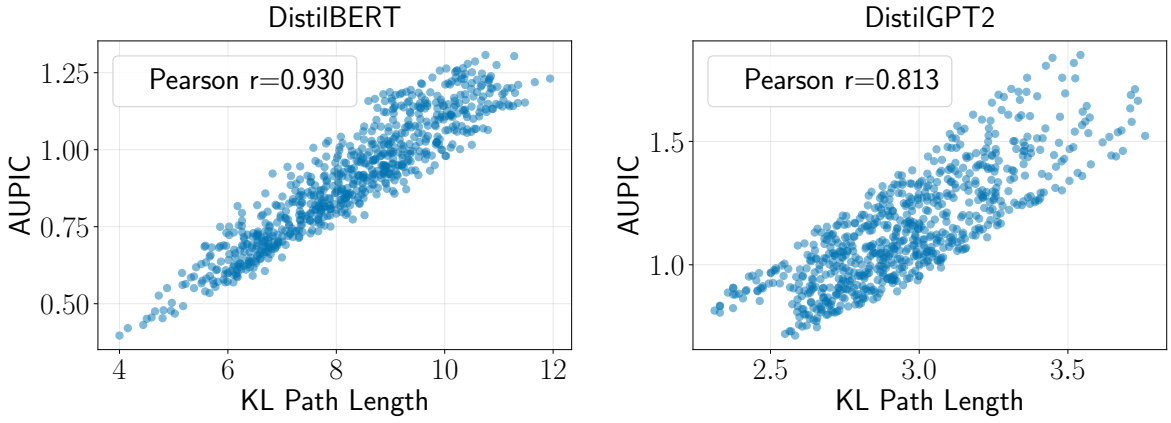

Figure 9. **KL Path Length on the Pile versus Wikitext AUPIC.** For all possible patching orders, when KL divergence is computed on a 64 example calibration set of the Pile and perplexity is computed on Wikitext, the KL path length in the interpolation graph correlates strongly with AUPIC.

J.2. Optimal Patching Order Closely Approximates Optimal Interpolation

In Section 2.2, we introduce the optimal interpolation problem (Problem 1) and the optimal permutation problem (Problem 2) for subset selection and argue that greedy algorithms naturally solve both formulations. In this section, we empirically validate that the optimal solution to Problem 2 is close in performance to the optimal solution to Problem 1. Thus, restricting the search space to the set of permutations does not impose a significant cap on performance as compared to the optimal interpolation problem.

Setup. We evaluate the perplexity of the exhaustive set of DistilBERT and DistilGPT2 patching orders from Section 3.1 and evaluate the classification accuracy of the set of 200 patching orders for Qwen3-4B-Base, Qwen3-8B-Base, and Pythia-6.9b from Section 3.2.

Results. Figure 10 shows that for DistilBERT models, the optimal permutation (lower dotted blue line) is exactly equal to the optimal interpolation (lower dotted gray line) for each model size. As a result, the minimum AUPIC for the optimal interpolation and patching order are identical (Table 1). For DistilGPT2, the optimal permutation is slightly higher in perplexity than the optimal interpolation for small model sizes (Figure 10), but is very close in AUPIC to the optimal interpolation (Table 1).

For larger models, while it is computationally infeasible to compute the exact optimal permutation and interpolation, Figure 11 demonstrates that on the subset of 200 patching orders that we evaluated, the maximum AUIC permutation (higher blue dotted line) has a classification accuracy close to that of the maximum interpolation of corresponding size (higher gray dotted line). We observe analogous results for perplexity and AUPIC in Figure 12. Similarly, the maximum AUIC across permutations is close to the maximum AUIC interpolation (Table 2). Overall, these results indicate that across models and tasks, the optimal permutation (Problem 2) is close in performance to the optimal interpolation (Problem 1).

Model	Interpolation	AUPIC (↓)
DistilBERT	Mean AUPIC	0.909
	Minimum AUPIC Interpolation	0.397
	Minimum AUPIC Permutation	0.397
DistilGPT2	Mean AUPIC	1.072
	Minimum AUPIC Interpolation	0.657
	Minimum AUPIC Permutation	0.713

Table 1. **AUPIC results for best performing interpolations vs permutations.** The minimum AUPIC permutation has AUPIC equal to or close to that of the minimum AUPIC interpolation.

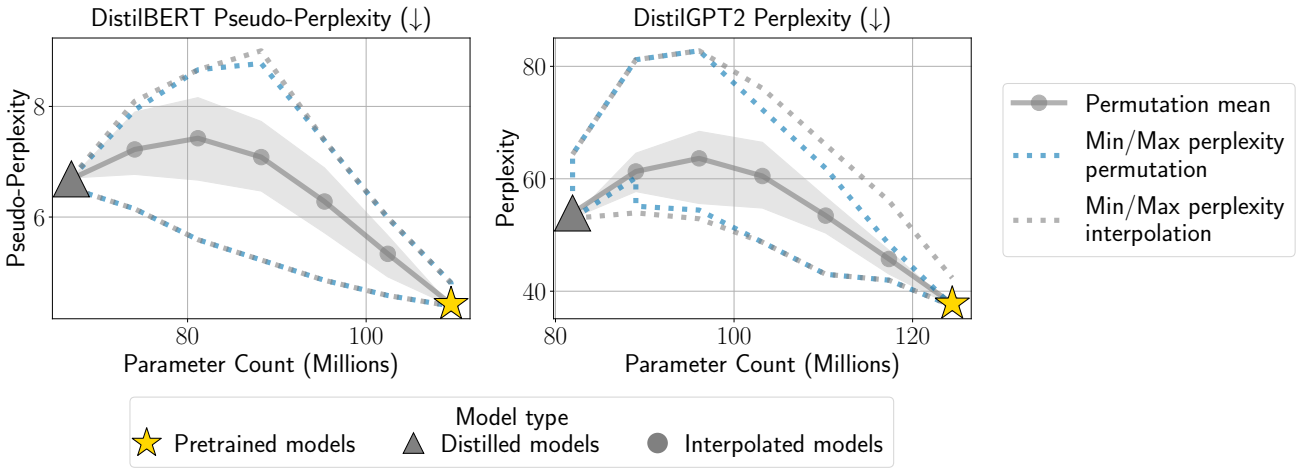

Figure 10. **Comparison of minimum AUPIC permutation vs interpolation for DistilBERT and DistilGPT2.** The minimum permutation produces the same perplexity as the minimum interpolation for DistilBERT and perplexity close to the minimum interpolation for DistilGPT2. Shaded bands show the (25-75) inter-quartile range over all 720 orderings.

Model	Interpolation	AUIC (↑)
Qwen3-4B-Base	Mean AUIC	0.409
	Maximum AUIC Interpolation	0.571
	Maximum AUIC Permutation	0.522
Qwen3-8B-Base	Mean AUIC	0.447
	Maximum AUIC Interpolation	0.634
	Maximum AUIC Permutation	0.585
Pythia-6.9b	Mean AUIC	0.209
	Maximum AUIC Interpolation	0.459
	Maximum AUIC Permutation	0.409

Table 2. **AUIC results for best performing interpolations vs permutations.** The maximum AUIC permutation has AUIC close to that of the maximum AUIC interpolation.

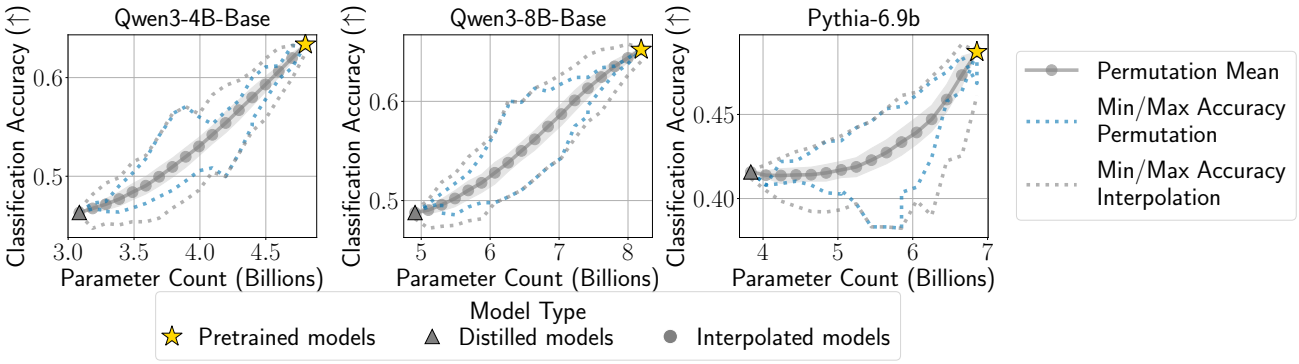

Figure 11. **Comparison of maximum AUIC permutation vs interpolation for Qwen3-4B-Base, Qwen3-8B-Base, and Pythia-6.9b over 200 random orderings.** Across models, the maximum AUIC permutation has classification accuracy very close to that of the maximum interpolation. Shaded bands show the (25-75) inter-quartile range over all 200 orderings.

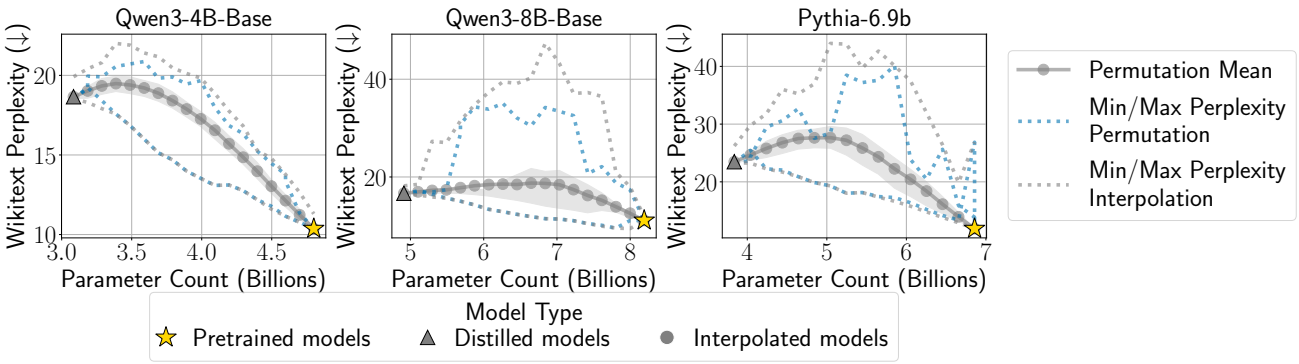

Figure 12. **Comparison of minimum AUPIC permutation vs interpolation for Qwen3-4B-Base, Qwen3-8B-Base, and Pythia-6.9b over 200 random orderings.** Across models, the minimum AUPIC permutation has perplexity very close to that of the minimum interpolation. Shaded bands show the (25-75) inter-quartile range over all 200 orderings.

J.3. KLPatch Path is Close to Global Shortest Path

We prove in Proposition 1 that shortest paths in $\mathcal{G}$ are optimal permutations and introduce KLPatch in Section 4 to find the approximate shortest path in a greedy manner. Here, we demonstrate that KLPatch produces paths that are close to the global shortest path in terms of AUPIC.

Setup. We use the set of exhaustive patching orders for DistilBERT and DistilGPT2 models from Section 3.1. We compute the KL divergence for every path using a calibration set of 64 examples from the Pile and report AUPIC for the global

shortest path and the path found using our KLPatch algorithm. We provide the first to last and last to first patching orders as baselines. For each patching order, we report the AUPIC and the *AUPIC Percentile*, or the percent of total patching orders that have AUPIC greater than or equal to that of the patching order being tested:

$$\text{AUPIC Percentile}(\pi) = \frac{1}{|\mathfrak{S}_N|} \sum_{\pi_i \in \mathfrak{S}_N} \mathbf{1}\{\text{AUPIC}(\pi_i) \geq \text{AUPIC}(\pi)\}$$

Results. Table 3 shows that KLPatch produces paths with interpolation performance equal to or very close to that of the global shortest KL path. Furthermore, last to first patching provides a very strong baseline for DistilBERT and DistilGPT2 models, but KLPatch recovers an AUPIC equal to (DistilBERT) or very close to (DistilGPT2) that of the last-to-first patching. These results validate that KLPatch finds a path equal to or close in performance to that of the global shortest KL path.

Model	Path	Patching Order π	AUPIC (↓)	AUPIC Percentile (↑)
DistilBERT	Minimum AUPIC Path	$[5, 4, 3, 2, 1, 0]$	0.397	100%
	First to Last Patching	$[0, 1, 2, 3, 4, 5]$	0.968	38.75%
	Last to First Patching	$[5, 4, 3, 2, 1, 0]$	0.397	100%
	Global Shortest Path	$[5, 4, 3, 2, 1, 0]$	0.397	100%
	KLPatch Path	$[5, 4, 3, 2, 1, 0]$	0.397	100%
DistilGPT2	Minimum AUPIC Path	$[4, 5, 3, 2, 0, 1]$	0.713	100%
	First to Last Patching	$[0, 1, 2, 3, 4, 5]$	0.824	95.56%
	Last to First Patching	$[5, 4, 3, 2, 1, 0]$	0.815	96.39%
	Global Shortest KL Path	$[0, 1, 2, 3, 4, 5]$	0.824	95.56%
	KLPatch Path	$[2, 3, 0, 1, 4, 5]$	0.831	95.28%

Table 3. **AUPIC results for different patching paths.** The KLPatch path has an AUPIC close to that of the global shortest KL path.

In Figure 13, we provide a visual representation of different patching orders, including the one found by KLPatch, to illustrate that KLPatch finds a permutation close to the optimal one.

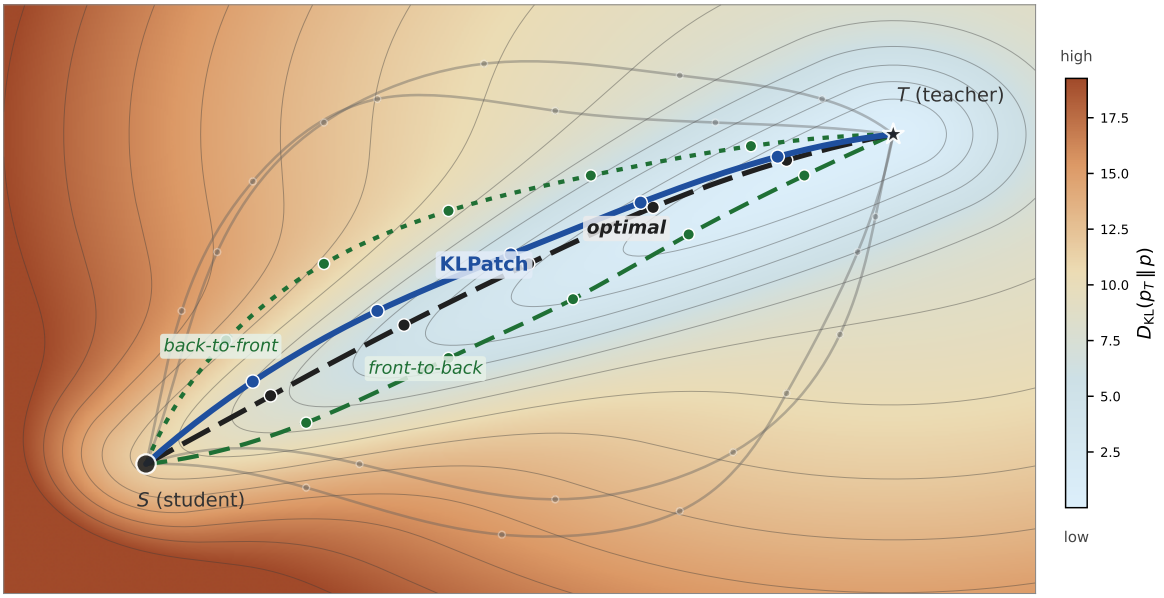

Figure 13. **KLPatch finds patching orders that are close to optimal:** Visual representation of the optimal patching order between the student $\boldsymbol{S}$ and teacher $\boldsymbol{T}$ along the KL divergence to the teacher model, compared to the KLPatch order, front-to-back and back-to-front, and random permutations.

K. Additional Permutation Results

We present extended results from Sections 3.1 and 3.2 on permutations of patching orders for model size interpolation. Figures 14-17 show that first to last and last to first patching orders often achieve the best interpolation trajectories compared to the random patching orders. These results show that simple sequential patching orders are a reliable recipe for achieving smooth model size interpolation.

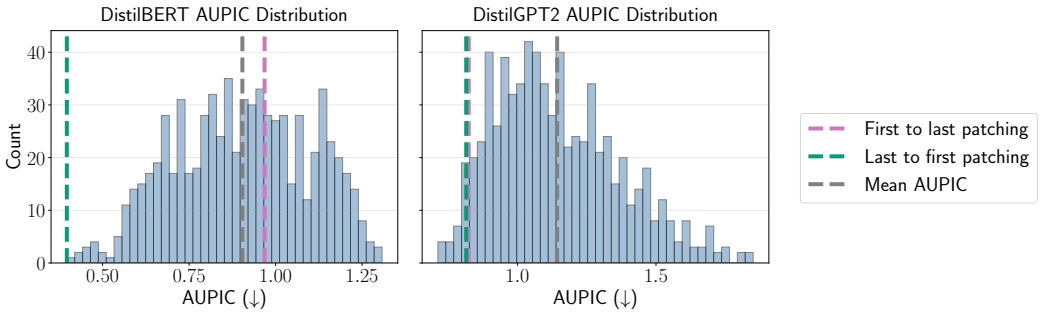

Figure 14. **AUPIC distribution across patching order permutations for DistilBERT and DistilGPT2.** Across the 720 orderings, last-to-first patching is at or near the optimum on both models, suggesting it is a simple and reliable default recipe for model size interpolation.

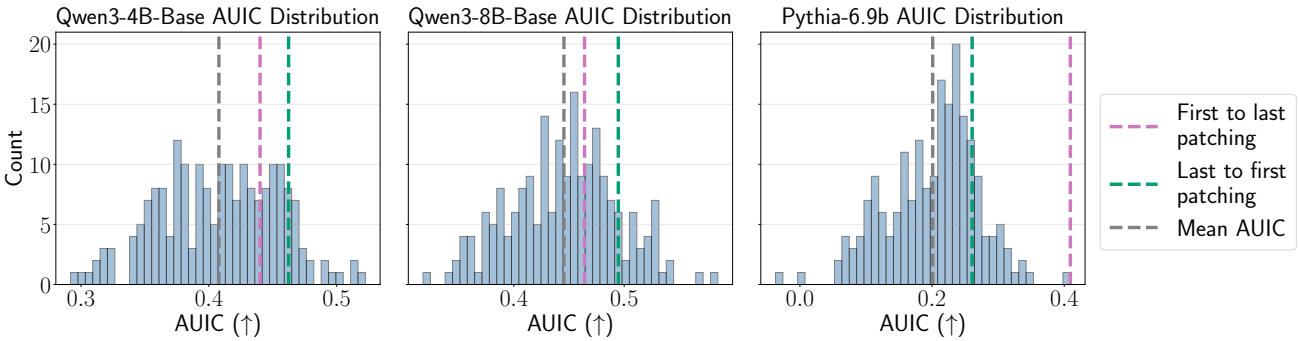

Figure 15. **Distribution of AUIC for different patching orders.** Across 200 orderings, first to last and last to first patching have relatively high AUIC, but last-to-first patching does not have the highest AUIC for Qwen models despite having the lowest AUPIC.

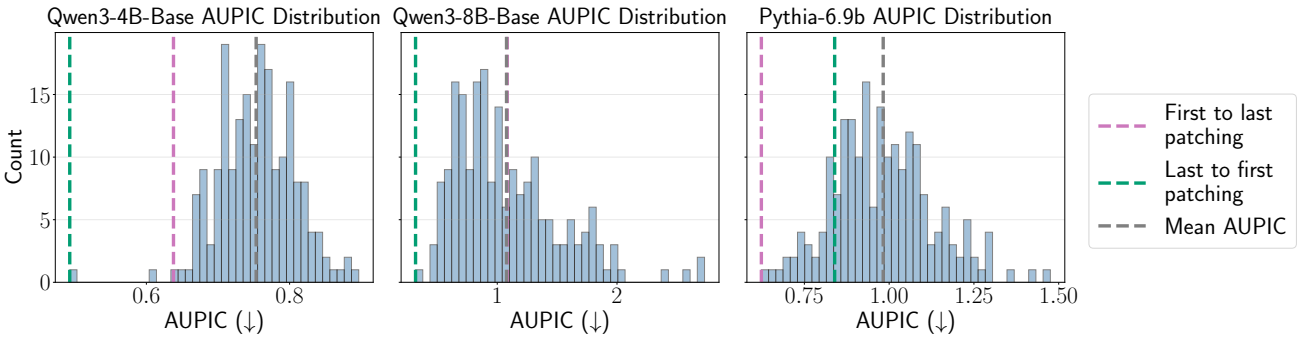

Figure 16. **Distribution of AUPIC for different patching orders.** Across the 200 orderings, either last to first or first to last patching often yields the highest AUPIC, suggesting it is a simple and reliable default recipe for model size interpolation.

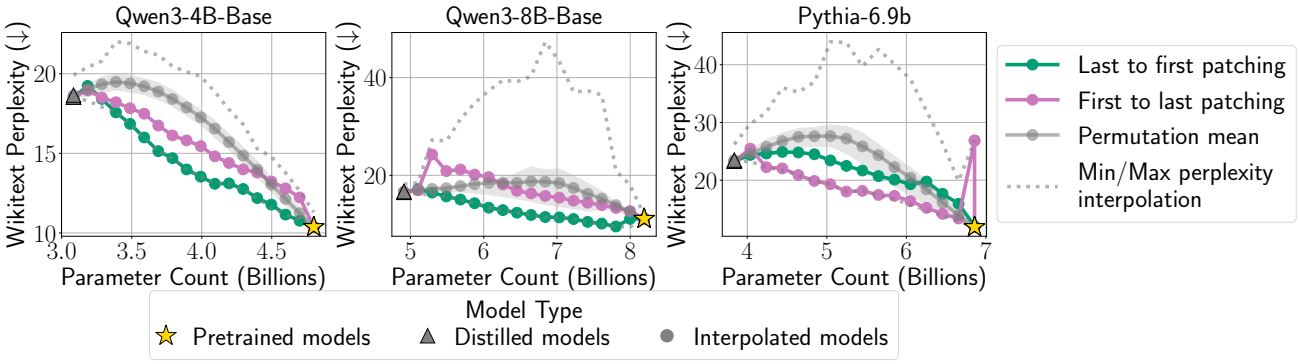

Figure 17. **Wikitext perplexity for different patching orders.** Across 200 orderings, either last to first patching and first to last patching achieve smooth interpolation performance, suggesting that they are generalizable recipes that work well across models. On the other hand, we observe that a naive patching order can sometimes result in poor interpolation performance. Shaded bands show the (25-75) inter-quartile range over all 200 orderings.

L. Additional KLPatch Results

We present additional results for KLPatch compared to baselines and on generation tasks and Wikitext.

L.1. KLPatch Trajectories are Best- or Near-Best-Performing

Setup. We compute the AUIC and AUPIC for the 200 random orderings from Section 3.2, as well as first to last, last to first, and KLPatch patching orders. We report the *AUIC percentile*, or the percent of total patching orders in the subset $S \subseteq \mathfrak{S}_N$ of orderings tested that have AUIC less than or equal to that of a given patching order π:

$$\text{AUIC Percentile}(\pi) = \frac{1}{|S_N|} \sum_{\pi_i \in S_N} \mathbf{1}\{\text{AUIC}(\pi_i) \leq \text{AUIC}(\pi)\}$$

We also compute the AUPIC percentile (defined in Section J.3) over S for each of the trajectories π being considered. We note that the percentile is with respect to 203 patching total patching orders (the set of 200 random orderings, first to last, last to first, and KLPatch).

Results. KLPatch produces the best or near-best trajectories in terms of AUIC and AUPIC across models (Tables 4-5). This shows that KLPatch is a reliable method for finding patching orders with high downstream performance.

Model	Interpolation	AUIC (↑)	AUIC Percentile (↑)
Qwen3-4B-Base	Maximum AUIC Interpolation	0.580	100%
	Maximum AUIC Permutation	0.533	100%
	First to Last Patching	0.441	69.46%
	Last to First Patching	0.462	89.16%
	KLPatch Path	0.533	100%
Qwen3-8B-Base	Maximum AUIC Interpolation	0.637	100%
	Maximum AUIC Permutation	0.585	100%
	First to Last Patching	0.464	63.55%
	Last to First Patching	0.495	85.71%
	KLPatch Path	0.571	99.51%
Pythia-6.9b	Maximum AUIC Interpolation	0.459	100%
	Maximum AUIC Permutation	0.410	100%
	First to Last Patching	0.410	100%
	Last to First Patching	0.261	82.27%
	KLPatch Path	0.379	99.51%

Table 4. **AUIC results for different interpolations.** KLPatch recovers a patching order with best (Qwen3-4B-Base) or near-best (Qwen3-8B-Base, Pythia-6.9b) performance in terms of AUIC.

Model	Interpolation	AUPIC (↓)	AUPIC Percentile (↑)
Qwen3-4B-Base	Minimum AUPIC Interpolation	0.481	100%
	Minimum AUPIC Permutation	0.493	100%
	First to Last Patching	0.637	69.46%
	Last to First Patching	0.493	100%
	KLPatch Path	0.539	99.51%
Qwen3-8B-Base	Minimum AUPIC Interpolation	0.277	100%
	Minimum AUPIC Permutation	0.318	100%
	First to Last Patching	1.084	41.38%
	Last to First Patching	0.318	100%
	KLPatch Path	0.357	99.51%
Pythia-6.9b	Minimum AUPIC Interpolation	0.545	100%
	Minimum AUPIC Permutation	0.567	100%
	First to Last Patching	0.623	99.51%
	Last to First Patching	0.839	83.74%
	KLPatch Path	0.567	100%

Table 5. **AUPIC results for different interpolations.** KLPatch recovers a patching order with the best (Pythia-6.9b) or near-best (Qwen3-4B-Base, Qwen3-8B-Base) performance out of the permutations tested in terms of AUPIC.

L.2. Generation Tasks

Figure 18 shows that KLPatch achieves better model size interpolation compared to last to first and first to last patching when we patch fewer student layers, but eventually converges to the best performing patching order trajectory on the generation tasks.

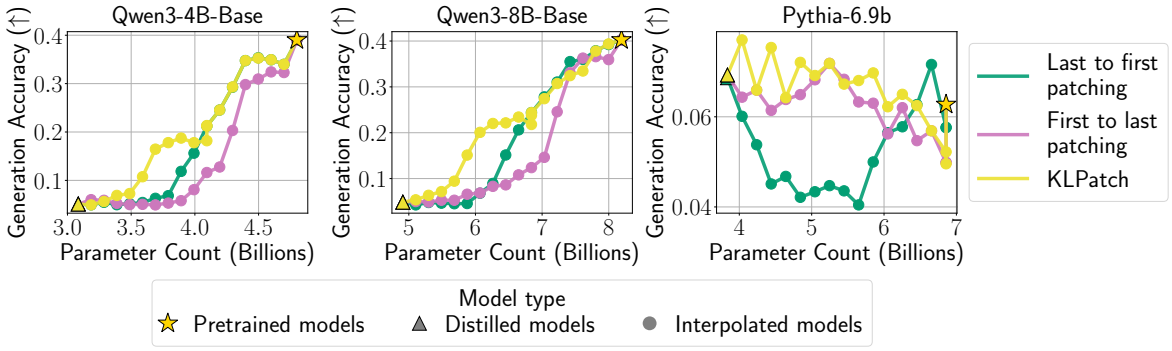

Figure 18. **KLPatch for Qwen3-4B-Base, Qwen3-8B-Base, and Pythia-6.9b** Across models, KLPatch often outperforms or tracks the last to first patching on generation tasks.

L.3. Wikitext Perplexity

Figure 19 shows that KLPatch often tracks closely to the best performing patching order among last to first and first to last patching orders on Wikitext.

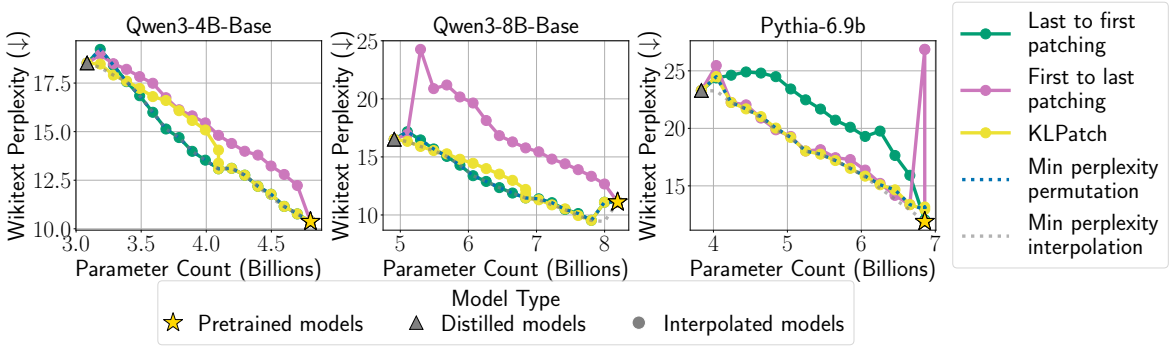

Figure 19. **KLPatch for Qwen3-4B-Base, Qwen3-8B-Base, and Pythia-6.9b** Across models, KLPatch tracks the last to first patching on perplexity.

M. Llama-3.2-3B KLPatch Results

In this section, we discuss the KLPatch of Llama 3.2 3B, which requires rather special considerations to achieve smooth interpolation.

Figure 20 shows that KLPatch severely underperforms first to last patching, suggesting that KLPatch might not universally improve over naive patching orders. However, after enforcing a heuristic constraint that the first student layer (layer 1) be patched first, then the KLPatch algorithm recovers the first to last patching.

One possible reason for this discrepancy could be the initialization of the student model. Kangaslahti et al. (2026) initialize the student model by keeping the first two layers from the teacher model. In contrast, when distilling other student models, namely Qwen 4B Base, Qwen 8B Base, and Pythia 6.9b, the student is initialized by keeping the last two layers. We suspect these differences also contribute to the differences in patching order. In fact, this difference in student model initialization also led Kangaslahti et al. (2026) to adopt the first to last patching strategy rather than the last to first patching strategy to achieve smooth interpolation. While these results show that Llama models require special treatment of KLPatch to achieve good model size interpolation performance, they also highlight that the effect of patching order is strongly influenced by the student model initialization.

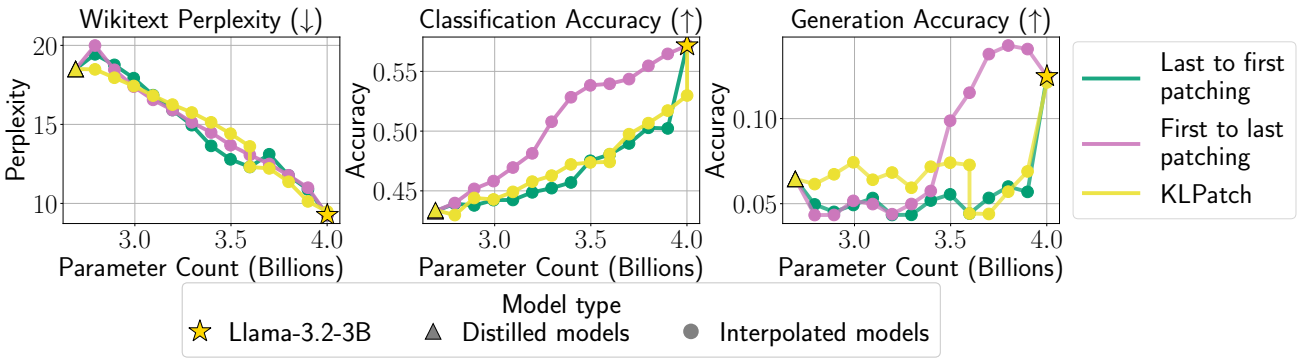

Figure 20. **Standard KLPatch for Llama-3.2-3B.** The standard KLPatch algorithm does not recover the optimal patching order.

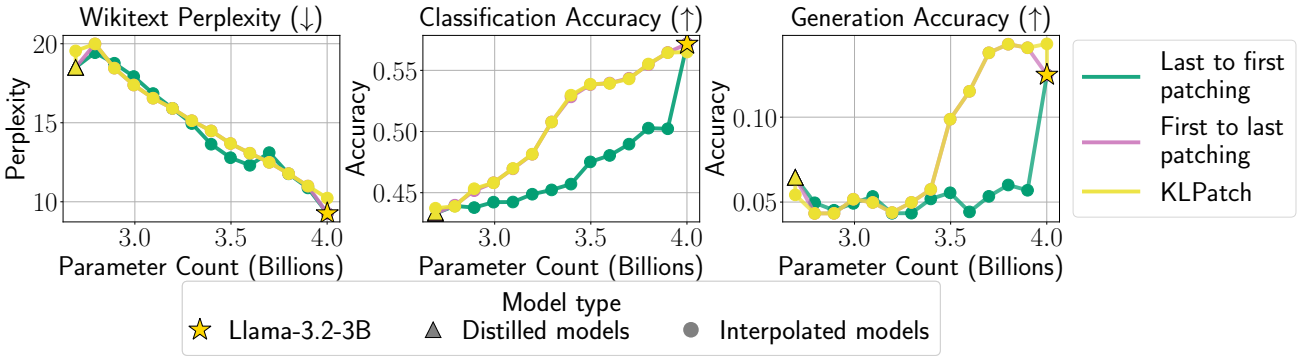

Figure 21. **KLPatch for Llama-3.2-3B by first patching Layer** 0**.** Enforcing layer 0 as the first patched layer allows KLPatch to recover the optimal patching order and improves performance by avoiding an initially suboptimal patching choice.