Understanding Layer Patching in Model Size Interpolation

Sara Kangaslahti[1], **Jonathan Geuter**[1,2]*, **Nihal V. Nayak**[1]*, **Marco Fumero**[3]*, **Francesco Locatello**[3], **David Alvarez-Melis**[1,2]
[1]Harvard University [2]Kempner Institute [3]IST Austria
sarakangaslahti@g.harvard.edu

Abstract

Layer patching is a key technique for model size interpolation in large language models (LLMs), in which a subset of a distilled student model's layers is replaced with corresponding teacher-model layers to construct intermediate-sized models without additional training. Prior work largely ignores the order in which layers are patched, which we show can dramatically affect the performance and reliability of the resulting models. In this work, we study the role of layer patching in 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. The code and models are available at https://github.com/dcml-lab/boomerang-distillation.

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]. *Layer patching* offers a training-free way to fill the gap between two such models. Starting from the smaller model, individual layers are replaced by their counterparts from the larger model, yielding intermediate models whose size and performance lie between those of the two endpoints. In boomerang distillation [Kangaslahti et al., 2026, Zhou et al., 2026], layers of a student LLM are patched with contiguous blocks of teacher layers once the student has been distilled from the teacher model. A central design choice lies in *which* layers to patch for a given target size, a key user-facing decision with a combinatorial design space. As we show, this choice strongly shapes the resulting interpolation, yet it has remained 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 patching 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

*Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

unexplored. Moreover, in Section 5, 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 reliable interpolated models.

In this work, we provide a comprehensive understanding of layer patching for model interpolation. We define layer patching as an optimization problem over interpolations (Section 4.1), which applies to any pair of smaller and larger models with aligned layers, accounts for patches of different sizes, and includes interpolations that need not correspond to nested subsets of model layers, moving beyond the setup of Kangaslahti et al. [2026]. As its computational complexity makes it infeasible to solve, we introduce an alternative shortest-path problem (Section 4.2) which we empirically and theoretically show approximates the optimal model interpolation. In experiments, we first 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 5.1). 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 5.3). Finally, we propose *KLPatch* (Section 6), 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 patchable layers, and apply it both to boomerang distillation and to interpolation between a quantized LLM and its higher-precision counterpart, another setting in which model size interpolation has been studied [Dumitru et al., 2025].

Our experiments reveal valuable insights into how patching order drives model 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-best interpolations among tested orders across several language models in boomerang distillation. Furthermore, the same algorithm yields strong interpolations between quantized and full-precision LLMs. Overall, our findings establish patching order as a key design choice for building interpolated models.

Our work makes the following contributions:

- **Formalization of layer patching.** We formalize layer patching between a smaller and a larger model as an optimization problem over interpolation curves (Section 4.1), and derive an alternative shortest-path problem, which we empirically and theoretically justify (Sections 4.2, 5).
- **Phenomenology of patching order.** We provide the first thorough study of layer patching. In particular, we perform exhaustive sweeps over *all* interpolations in DistilBERT and DistilGPT (Section 5.1), and 200 random patching orders on Qwen3 4B, Qwen3 8B, and Pythia 6.9B (Section 5.3), 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 6.1), reducing the computational complexity from $O(2^N)$ to $O(N^2)$ and empirically yielding model interpolations close to the best observed performance for boomerang distillation (Section 6.2) and for interpolation between quantized and bf16 LLMs (Section 6.3).

2 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 and Jaggi, 2020, Ainsworth et al., 2023, Yang et al., 2026]. Several works [Entezari et al., 2021, Nagarajan and 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

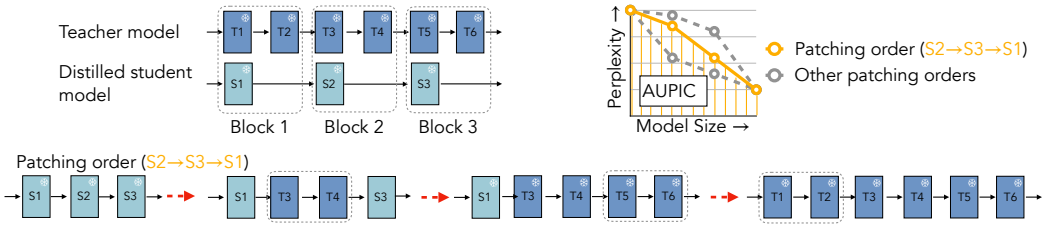


Figure 1: **Patching order in model size interpolation.** We show boomerang distillation with a six-layer teacher and a three-block distilled student, where each student block corresponds to a contiguous pair of two-layer teacher blocks. A patching order specifies the sequence in which student layers are replaced by their corresponding teacher blocks, inducing a trajectory of intermediate-size models from the student to the teacher. Here, the order $S2 \rightarrow S3 \rightarrow S1$ 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).

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 3). 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 6, 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.

3 Boomerang Distillation

We provide an empirical case study building 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). The procedure has three steps, which we describe below: (1) student initialization, (2) knowledge distillation, and (3) student patching.

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

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

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 layers 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.

4 Layer Patching as a Shortest-Path Problem

First, we formalize student patching as a combinatorial optimization problem (Section 4.1). We then recast the optimization as a shortest-path problem on a KL-weighted graph (Section 4.2), motivating the KLPatch algorithm of Section 6.

4.1 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 5 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 $f_{\boldsymbol{S},\boldsymbol{T}} : 2^{\{1,\ldots,N\}} \to \mathbb{R}$ that scores each subset A by the quality of $\boldsymbol{M}_A$, with higher meaning better.[2] The best k-subset then maximizes $f_{\boldsymbol{S},\boldsymbol{T}}(A)$ subject to $|A| = k$; collecting these maximizers across all k defines an *optimal interpolation* between $\boldsymbol{S}$ and $\boldsymbol{T}$.

Problem 1 (Optimal Interpolation)**.** Given a student $\boldsymbol{S}$, a teacher $\boldsymbol{T}$, and an objective function $f_{\boldsymbol{S},\boldsymbol{T}} : 2^{\{1,\ldots,N\}} \to \mathbb{R}$, 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

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

For simplicity, we define Problem 1 over cardinality instead of model size, but an equivalent definition taking size into account could be used as well. 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

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

permutation of layers naturally gives rise to an interpolation of models by patching layers sequentially according to the permutation.

An optimal permutation π^* should also take the incremental costs $d_{\pi^*(k)}$ of patching the kth layer into account, and maximize the **area under the interpolation curve (AUIC)** of $f_{\boldsymbol{S},\boldsymbol{T}}$, which we define as

$$\mathrm{AUIC}_f(\pi) \triangleq \sum_{k=1}^{N} d_{\pi(k)} \frac{f_{\boldsymbol{S},\boldsymbol{T}}(\boldsymbol{M}_{A_k^\pi}) + f_{\boldsymbol{S},\boldsymbol{T}}(\boldsymbol{M}_{A_{k-1}^\pi})}{2},$$

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

Problem 2 (Optimal Permutation)**.** Given a student $\boldsymbol{S}$, a teacher $\boldsymbol{T}$, and an objective function $f_{\boldsymbol{S},\boldsymbol{T}} : 2^{\{1,\ldots,N\}} \to \mathbb{R}$, the *optimal permutation* $\pi^* \in \mathfrak{S}_N$ of layers of $\boldsymbol{S}$ with respect to $f_{\boldsymbol{S},\boldsymbol{T}}$ is

$$\pi^* = \arg\max_{\pi \in \mathfrak{S}_N} \mathrm{AUIC}_f(\pi).$$

Computational complexity. The permutation search space is in fact larger than the interpolation search space ($|\mathfrak{S}_N| = N!$ versus $2^N - 2$). However, we note that the computational complexity of both problems is similar. This is because the computationally expensive part is typically evaluating $f_{\boldsymbol{S},\boldsymbol{T}}$, as for most evaluation functions this requires computing inference passes over an interpolated model. For both Problem 1 and 2, $f_{\boldsymbol{S},\boldsymbol{T}}$ needs to be evaluated for all of the possible $2^N - 2$ interpolated models. Once these $f_{\boldsymbol{S},\boldsymbol{T}}$ values are obtained, searching over interpolations or permutations is relatively cheap in comparison. Searching over interpolations corresponds to computing the maximum over $f_{\boldsymbol{S},\boldsymbol{T}}$ for each model size, or computing a maximum over $\binom{N}{k}$ real numbers for $k \in \{1, \ldots, N\}$. Similarly, performing a search over permutations requires computing the sum over $f_{\boldsymbol{S},\boldsymbol{T}}(A)$ for each permutation, or computing a sum over N real numbers for $N!$ permutations. Thus, finding the optimal permutation has effectively the same computational complexity as finding the optimal interpolation, as any $\mathrm{AUIC}_f(\pi)$ can be quickly evaluated once $f_{\boldsymbol{S},\boldsymbol{T}}(A)$ is known for all $A \subset \{1, ..., N\}$. The computational complexity of both problems is summarized in Table 1.

Problem	Evaluation Complexity	Search Space Size
Problem 1 (Optimal Interpolation)	$2^N - 2$ evaluations each costing $O(f_{\boldsymbol{S},\boldsymbol{T}})$	$2^N - 2$
Problem 2 (Optimal Permutation)	$2^N - 2$ evaluations each costing $O(f_{\boldsymbol{S},\boldsymbol{T}})$	$N!$

Table 1: **Computational complexity of Problems 1 and 2.** Both problems require the same number of evaluations of $f_{\boldsymbol{S},\boldsymbol{T}}$, but Problem 2 has a higher exhaustive search complexity over the evaluated models. However, evaluations of $f_{\boldsymbol{S},\boldsymbol{T}}$ are significantly more computationally expensive, since they require evaluating an interpolated model (often through inference), while each evaluation in the exhaustive search consists of comparing or summing $O(N)$ real numbers.

We will see in Section 4.2 that Problem 2 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 6 (KLPatch). Empirically, we also observe that solutions to Problem 2 closely track solutions to Problem 1 in terms of perplexity and downstream accuracy (Appendix H.2).

For $f_{\boldsymbol{S},\boldsymbol{T}} = \mathrm{PPL}_{\mathrm{data}}$, we will furthermore write $\mathrm{AUPIC}(\pi) \triangleq \mathrm{AUIC}_f(\pi)$ for the **area under the perplexity curve (AUPIC)** w.r.t. data-perplexity $\mathrm{PPL}_{\mathrm{data}}$. 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 4.2. Replacing data-perplexity with perplexity against samples drawn from $\boldsymbol{T}$ yields teacher-relative variants $\mathrm{AUPIC}_{\boldsymbol{T}}(\pi)$ and $\mathrm{AUPIC}^{\log}_{\boldsymbol{T}}(\pi)$; these underpin the shortest-path characterization of Section 4.2 (Proposition 1).

4.2 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 $\{1, \dots, N\}$, 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 (Permutation Graph)**.** Let $\mathcal{G}$ be the Boolean lattice on $\{1, \dots, N\}$, viewed as a directed acyclic graph: vertices are subsets $A \subseteq \{1, \dots, N\}$, 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 $\boldsymbol{M}_{\{1,\dots,N\}} = \boldsymbol{T}$. Given a calibration dataset $\mathcal{D}_{\text{cal}}$ and writing

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

assign each edge the weight

$$w\big(\boldsymbol{M}_A \to \boldsymbol{M}_{A\cup\{i\}}\big) \triangleq d_i \frac{K(A) + K(A\cup\{i\})}{2}.$$

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

$$E(\pi) \triangleq \sum_{e\in\gamma_\pi} w(e) \;=\; \sum_{k=0}^{N-1} d_{\pi(k)} \frac{K(A_k^\pi) + K(A_{k+1}^\pi)}{2}.$$

Shortest paths in $\mathcal{G}$ are precisely the optimal permutations from Problem 2 with $f_{\boldsymbol{S},\boldsymbol{T}} = \log \mathrm{PPL}_{\boldsymbol{T}}$.

Proposition 1 (Shortest KL Paths are optimal)**.** *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 the combinatorial search Problem 2 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 E: 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 **??**. We note that in our experiments in Sections 5 and 6, 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 H.1, Figures 8 and 10). Section 6 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 H.1, Figures 7 and 9). In practice, KL performed better than cosine distance (Appendix L.1), which correlates poorly with downstream performance [Zhang et al., 2024, Hinostroza et al., 2026].

5 Empirical Characterization of Layer Patching

Section 4 cast the choice of patching order as a combinatorial optimization with a clean shortest-path characterization. We now characterize this optimization landscape empirically: Section 5.1 performs a full sweep over all possible interpolations on DistilBERT and DistilGPT2; Section 5.2 relates AUPIC to Spearman's footrule distance from the optimum; and Section 5.3 samples and evaluates a subset of patching orders for larger LLMs, where full enumeration is infeasible. Across all three 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 6.

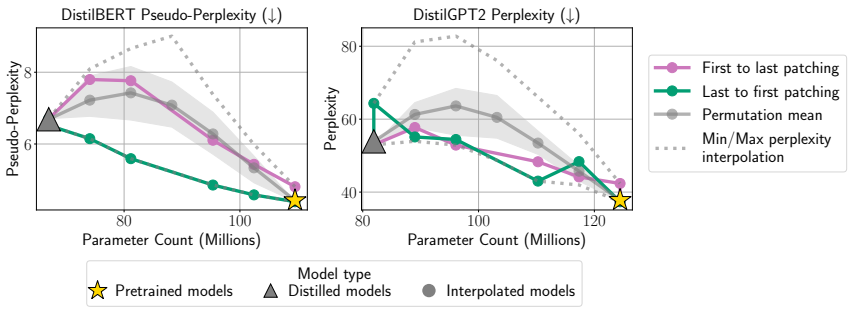


Figure 2: **All permutations of patching order for DistilBERT and DistilGPT2.** Patching order significantly affects interpolation performance, and the best ordering yields nearly linear interpolation in (pseudo-)perplexity between the students (DistilBERT, DistilGPT2) and teachers (BERT, GPT2). Shaded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

5.1 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 indeed all $2^6 = 64$ partially-patched models—directly enumerable, allowing us to characterize interpolation behavior exhaustively rather than through sampling.

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. We report the order with the lowest AUPIC as the empirical optimum.

Results. Patching order has a substantial impact on interpolation performance (Figure 2, Appendix Figure 15). 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 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 6.

5.2 Good Patching Orders Cluster Around the Optimum

Section 5.1 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 5.1 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 $d(\pi, \pi') = \sum_{i=1}^{N} |\pi(i) - \pi'(i)|$, 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 at the largest distances (Figure 3). Concretely, 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 4.2, near-optimal patching orders correspond to short paths in $\mathcal{G}$ that share a substantial number of edges with the shortest path; this local structure is what

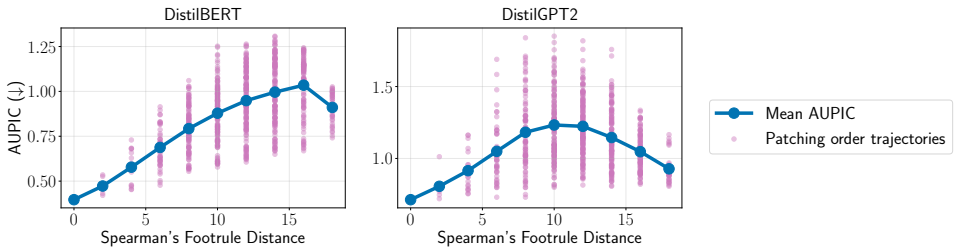


Figure 3: **AUPIC vs. Spearman footrule distance from the minimum-AUPIC permutation.** On both DistilBERT and DistilGPT2, mean AUPIC rises with distance from the optimum and then flattens, indicating a cluster of near-optimal orderings around the empirical minimum.

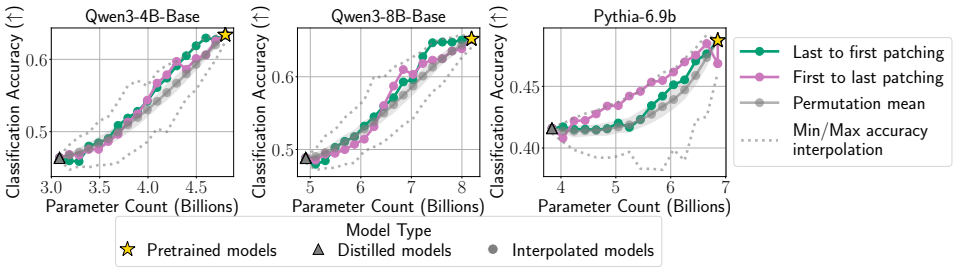


Figure 4: **Downstream interpolation curves across patching orders.** For each of Qwen3-4B, Qwen3-8B, and Pythia-6.9B, the mean downstream accuracy is plotted against patched model size, across the 200 sampled orderings together with the first-to-last and last-to-first baselines. Shaded bands show the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

makes greedy approaches such as KLPatch (Section 6) viable: small, locally-informed deviations from the shortest-path skeleton tend not to leave the near-optimal basin.

5.3 Effect of Layer Patching in Large Language Models

We now ask whether the small-model findings of Section 5.1 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 G 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 I.

Results. Across all three models, patching order has a substantial impact on downstream interpolation quality (Figure 4, Appendix Figure 16). The small-model regularity, however, does not transfer cleanly: while last-to-first attains the lowest *Wikitext* AUPIC (Appendix I), 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-best order—rather than averaging over random ones—is large in practice; Section 6 develops a procedure for doing so.

6 KL-Guided Patching Orders

Section 5 showed that patching order has a substantial impact on interpolation quality, and that fixed orderings (first-to-last, last-to-first) leave a large headroom relative to the empirical optimum. We now turn this characterization into an algorithm.

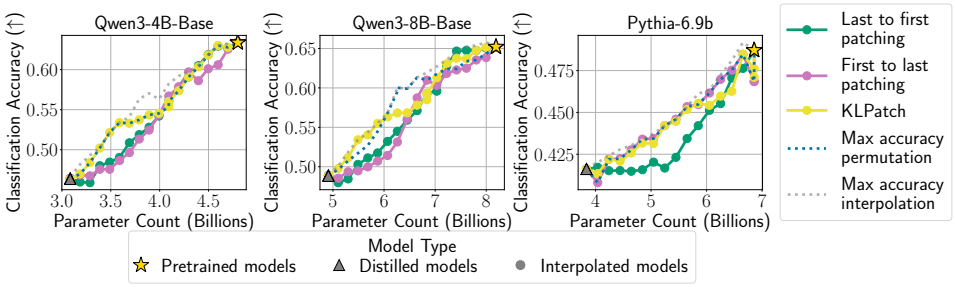


Figure 5: **KLPatch interpolation curves on Qwen3-4B, Qwen3-8B, and Pythia-6.9B.** For each model, mean downstream accuracy is plotted against the patched student size, comparing KLPatch (Algorithm 1) against first-to-last and last-to-first baselines. Across all three models, KLPatch tracks or outperforms last-to-first on both classification and generation tasks.

6.1 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 $N-k$ candidates, KLPatch reduces the search from $O(2^N)$ exhaustive evaluations 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 6.2), we show that it yields permutations close to the optima of both Problem 1 and Problem 2.

Algorithm 1 KLPatch: Iterative Greedy Construction of a Patching Order for Model Interpolation

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

$U \leftarrow \{1, \ldots, N\}$; $\quad \boldsymbol{M} \leftarrow \boldsymbol{S}$; $\quad \pi \leftarrow [\,]$
while $U \neq \emptyset$ **do** ▷ Iteratively select the next layer to patch.
  for each $i \in U$ **do** ▷ Patch the student model and measure the KLDivergence.
    $\tilde{\boldsymbol{M}}^{(i)} \leftarrow \textsc{Patch}(\boldsymbol{M}, i, \mathbf{b}^{(i)})$ ▷ replace ith layer in $\hat{\boldsymbol{S}}$ with $\mathbf{b}^{(i)}$
    $\ell_i \leftarrow \mathbb{E}_{x \sim \mathcal{D}_{\text{cal}}} \mathrm{KL}(p_{\boldsymbol{T}}(\cdot \mid x) \| p_{\tilde{\boldsymbol{M}}^{(i)}}(\cdot \mid x))$
  end for
  $i^\star \leftarrow \arg\min_i \ell_i$ ▷ Select the student layer with lowest KL to be patched.
  $\boldsymbol{M} \leftarrow \textsc{Patch}(\boldsymbol{M}, i^\star, \mathbf{b}^{(i^\star)})$
  $\pi \leftarrow \pi \,\|\, [i^\star]$ ▷ Add the student layer index to the patching order sequence π.
  $U \leftarrow U \setminus \{i^\star\}$
end while
return π

6.2 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 5.3 (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 K. In Appendix J.1, following Kangaslahti et al. [2026], we compare KLPatch to additional baselines such as layer pruning and randomly initializing the student model during distillation.

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 5), generation (Appendix J.4), and Wikitext perplexity (Appendix J.5). 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 5), yielding near-best performance out of the sampled interpolation trajectories on these three models. In Appendix K, 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. We note that Llama was shown to be a special case and required different initialization and patching order for boomerang

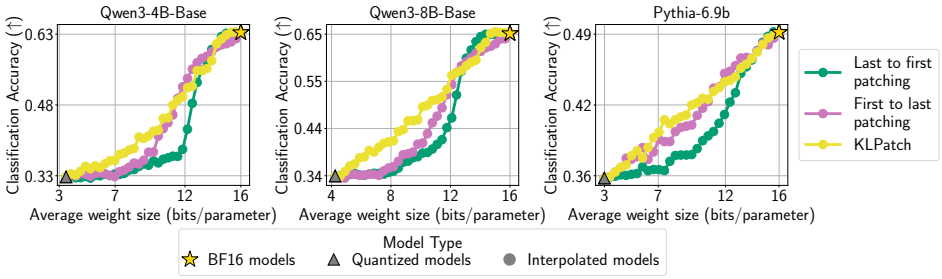


Figure 6: **KLPatch interpolation curves for variable quantization.** Mean downstream accuracy is plotted against bits per parameter, comparing KLPatch (Algorithm 1) against first-to-last and last-to-first baselines. Across all three models, KLPatch tracks or outperforms sequential patching orders.

distillation [Kangaslahti et al., 2026], indicating that the effect of patching order is influenced by the student model initialization. Overall, KLPatch provides an efficient approach for reliably creating high-performing model interpolations.

6.3 KLPatch for Variable Quantization

Setup. We apply KLPatch and sequential patching orders directly to the setting of variable quantization by interpolating between quantized and bf16 LLMs. We use Qwen3-4B Base, Qwen3-8B Base, and Pythia-6.9B at bf16 precision as $\boldsymbol{T}$ and construct $\boldsymbol{S}$ by compressing each model with 2-bit GPTQ quantization [Frantar et al., 2023]. Similarly to Dumitru et al. [2025], we use 2-bit quantization as a testbed for $\boldsymbol{S}$ since we do not observe a significant drop in performance for 4-bit or 8-bit precision models. We note that the embedding layer and LM head are kept full-precision for compatibility. Starting from the quantized model $\boldsymbol{S}$, we can increase the bits per parameter of the model by patching any quantized layer k in $\boldsymbol{S}$ with the kth bf16 layer in $\boldsymbol{T}$. We use the same calibration set $\mathcal{D}_{\text{cal}}$ from Section 6.2 of 64 samples drawn from the Pile [Gao et al., 2021] and evaluate the interpolated models on the classification task suite from Section 5.3.

Results. KLPatch produces interpolations that outperform or match first-to-last and last-to-first patching orders when constructing models with variable quantization (Figure 6). KLPatch especially compares favorably to sequential strategies in the region with small average bits per parameter, indicating that it is particularly beneficial for constructing highly quantized models. These results provide empirical evidence that our shortest-paths formulation and KLPatch algorithm, which are theoretically applicable to any per-layer intervention method, generalize beyond boomerang distillation to other model interpolation contexts. Applying our optimization problem and KLPatch to new settings, for example in model merging via layer swapping [Bandarkar et al., 2025, Shin and Hwang, 2026], is a promising direction for future research.

7 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. We also show that KLPatch can be applied to other model interpolation settings such as variable quantization.

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.

Acknowledgments

David Alvarez-Melis, Sara Kangaslahti, Jonathan Geuter, and Nihal V. Nayak acknowledge support from the National Science Foundation Graduate Research Fellowship (Grant No. DGE 2140743), the Kempner Institute, FAS Dean's Competitive Fund for Promising Scholarship, Aramont Fellowship Fund, and the NSF AI-SDM Institute (Grant No. IIS-2229881). Jonathan Geuter is supported by a Kempner Institute Graduate Fellowship. Francesco Locatello's contribution to this research was funded in part by the Austrian Science Fund (FWF) 10.55776/COE12.

References

Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha S. Srinivasa. 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.

Lucas Bandarkar, Benjamin Muller, Pritish Yuvraj, Rui Hou, Nayan Singhal, Hongjiang Lv, and Bing Liu. Layer swapping for zero-shot cross-lingual transfer in large language models, 2025. URL https://arxiv.org/abs/2410.01335.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. Pythia: A suite for analyzing large language models across training and scaling. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 2397–2430. PMLR, 2023. URL https://proceedings.mlr.press/v202/biderman23a.html.

Yonatan Bisk, Rowan Zellers, Ronan LeBras, Jianfeng Gao, and Yejin Choi. 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*, pages 7432–7439. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/6239.

Ruisi Cai, Saurav Muralidharan, Hongxu Yin, Zhangyang Wang, Jan Kautz, and Pavlo Molchanov. 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.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *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)*, pages 2924–2936, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

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, 2021. URL https://arxiv.org/abs/2110.14168.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *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), pages 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Razvan-Gabriel Dumitru, Vikas Yadav, Rishabh Maheshwary, Paul Ioan Clotan, Sathwik Tejaswi Madhusudhan, and Mihai Surdeanu. Variable layerwise quantization: A simple and effective approach to quantize LLMs. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 534–550, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.29. URL https://aclanthology.org/2025.findings-acl.29/.

Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. *arXiv preprint arXiv:2110.06296*, 2021.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. 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*, pages 3259–3269. PMLR, 2020. URL http://proceedings.mlr.press/v119/frankle20a.html.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tcbBPnfwxS.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling, 2021. URL https://arxiv.org/abs/2101.00027.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 2023. URL https://zenodo.org/records/10256836.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *ArXiv preprint*, abs/2407.21783, 2024. URL https://arxiv.org/abs/2407.21783.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural network. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1135–1143, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/ae0eb3eed39d2bcef4622b2499a05fe6-Abstract.html.

Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European conference on computer vision (ECCV)*, pages 784–800, 2018.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 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.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021b.

Cristian Hinostroza, Rodrigo Toro Icarte, Christ Devia, Andres Carvallo De Ferari, Eugenio Herrera-Berg, Denis Parra, and Jorge F Silva. 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.

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

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, DDL Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 10, 2022.

Bairu Hou, Qibin Chen, Jianyu Wang, Guoli Yin, Chong Wang, Nan Du, Ruoming Pang, Shiyu Chang, and Tao Lei. Instruction-following pruning for large language models. *arXiv preprint arXiv:2501.02086*, 2025.

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

Sara Kangaslahti, Nihal V. Nayak, Jonathan Geuter, Marco Fumero, Francesco Locatello, and David Alvarez-Melis. Boomerang distillation enables zero-shot model size interpolation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=4ZU8v4s3IR.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Apoorv Khandelwal, Tian Yun, Nihal V. Nayak, Jack Merullo, Stephen Bach, Chen Sun, and Ellie Pavlick. $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.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale ReAding comprehension dataset from examinations. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark, 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1082. URL https://aclanthology.org/D17-1082.

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

Hong Liu, Sang Michael Xie, Zhiyuan Li, and Tengyu Ma. Same pre-training loss, better downstream: Implicit bias matters for language models. 2023. doi: 10.48550/arXiv.2210.14199.

Amir M. Mansourian, Rozhan Ahmadi, Masoud Ghafouri, Amir Mohammad Babaei, Elaheh Badali Golezani, Zeynab yasamani ghamchi, Vida Ramezanian, Alireza Taherian, Kimia Dinashi, Amirali Miri, and Shohreh Kasaei. A comprehensive survey on knowledge distillation. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=3cbJzdR78B.

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 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.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 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.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260.

Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. Compact language models via pruning and knowledge distillation. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *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.

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

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

nostalgebraist. Interpreting gpt: The logit lens, 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI*, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 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*, pages 8732–8740. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/6399.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv preprint*, abs/1910.01108, 2019. URL https://arxiv.org/abs/1910.01108.

Hyunseo Shin and Wonseok Hwang. Layer-wise swapping for generalizable multilingual safety, 2026. URL https://arxiv.org/abs/2601.22620.

Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *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.

Sharath Turuvekere Sreenivas, Saurav Muralidharan, Raviraj Joshi, Marcin Chochowski, Ameya Sunil Mahabaleshwarkar, Gerald Shen, Jiaqi Zeng, Zijia Chen, Yoshi Suhara, Shizhe Diao, et al. Llm pruning and distillation in practice: The minitron approach. *ArXiv preprint*, abs/2408.11796, 2024. URL https://arxiv.org/abs/2408.11796.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. 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.

Rich Sutton. The bitter lesson. 2019.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Y Charles, HS Che, Cheng Chen, Guanduo Chen, et al. Kimi k2. 5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 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.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv preprint*, abs/1910.03771, 2019. URL https://arxiv.org/abs/1910.03771.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 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.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *ArXiv preprint*, abs/2505.09388, 2025. URL https://arxiv.org/abs/2505.09388.

Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. 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.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472.

Yang Zhang, Yawei Li, Xinpeng Wang, Qianli Shen, Barbara Plank, Bernd Bischl, Mina Rezaei, and Kenji Kawaguchi. Finercut: Finer-grained interpretable layer pruning for large language models. *arXiv preprint arXiv:2405.18218*, 2024.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

Yan Zhou, Sara Kangaslahti, Jonathan Geuter, Nihal V. Nayak, Marco Fumero, Francesco Locatello, and David Alvarez-Melis. Thinking at the right size: Amortized distillation across post-trained LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2026*. Association for Computational Linguistics, 2026. URL https://arxiv.org/abs/2608.22854.

Contents

1 Introduction **1**

2 Related Work **2**

3 Boomerang Distillation **3**

4 Layer Patching as a Shortest-Path Problem **4**

4.1 A Combinatorial Optimization Perspective 4

4.2 Patching Orders as Shortest Paths 6

5 Empirical Characterization of Layer Patching **6**

5.1 Full Sweep on DistilBERT and DistilGPT2 7

5.2 Good Patching Orders Cluster Around the Optimum 7

5.3 Effect of Layer Patching in Large Language Models 8

6 KL-Guided Patching Orders **8**

6.1 The KLPatch Algorithm 9

6.2 KLPatch Finds Near-Optimal Patching Orders 9

6.3 KLPatch for Variable Quantization 10

7 Conclusion **10**

A Limitations **18**

B Reproducibility Statement **18**

C Broader Impacts **18**

D Statement on Use of Large Language Models **18**

E Proofs **18**

F KLPatch Update Rule **20**

G Experimental Details **22**

H Empirically Validating Theoretical Assumptions **23**

H.1 KL Divergence Strongly Correlates with Perplexity 23

H.2 Optimal Patching Order Closely Approximates Optimal Interpolation 24

H.3 KLPatch Path is Close to Global Shortest Path 26

I Additional Permutation Results **27**

J Additional KLPatch Results **28**

J.1 Additional Baselines 28

J.2 KLPatch Trajectories are Best- or Near-Best-Performing 28

J.3 KLPatch Significantly Reduces Runtime 29

J.4 Generation Tasks 30
J.5 Wikitext Perplexity 30
J.6 Per-Task Performance 31

K Llama-3.2-3B KLPatch Results **31**

L KLPatch Ablations **32**
L.1 KLPatch Performs Best with KLDivergence 32
L.2 Iterative KLPatch Outperforms Non-Iterative Approaches 34
L.3 Additional search algorithms 35
L.4 Ablating the calibration set 35

A 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 K for more details.

B 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.

C 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.

D 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.

E Proofs

Proposition 1 (Shortest KL Paths are optimal)**.** *The shortest paths in $\mathcal{G}$ are exactly the optimal permutations with respect to* $\mathrm{AUPIC}_{\boldsymbol{T}}^{\log}(\pi)$ *(to be understood as a set-equality if there are multiple minimizers):*

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

Proof. For any intermediate model $\boldsymbol{M}_A$, $A \subseteq \{1,\dots,N\}$, the cross-entropy decomposition gives

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

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

$$K(A) = \log \mathrm{PPL}_{\boldsymbol{T}}(\boldsymbol{M}_A) - \log \mathrm{PPL}_{\boldsymbol{T}}(\boldsymbol{T}).$$

Applying this identity to each intermediate model $\boldsymbol{M}_k^\pi$, i.e. to $A = A_k^\pi$, and summing over the edges of γ_π,

$$\begin{aligned}
E(\pi) &= \sum_{k=1}^{N} d_{\pi_k} \frac{K(A_{k-1}^\pi) + K(A_k^\pi)}{2} \\
&= \sum_{k=1}^{N} d_{\pi_k} \frac{\log \mathrm{PPL}_{\boldsymbol{T}}(\boldsymbol{M}_{k-1}^\pi) + \log \mathrm{PPL}_{\boldsymbol{T}}(\boldsymbol{M}_k^\pi)}{2} - \log \mathrm{PPL}_{\boldsymbol{T}}(\boldsymbol{T}) \sum_{k=1}^{N} d_{\pi_k} \\
&= \mathrm{AUPIC}_{\boldsymbol{T}}^{\log}(\pi) - (|\boldsymbol{T}| - |\boldsymbol{S}|) \log \mathrm{PPL}_{\boldsymbol{T}}(\boldsymbol{T}),
\end{aligned}$$

using the fact that $\sum_{k=1}^{N} d_{\pi_k} = |\boldsymbol{M}_N^\pi| - |\boldsymbol{M}_0^\pi| = |\boldsymbol{T}| - |\boldsymbol{S}|$, since $\boldsymbol{M}_0^\pi = \boldsymbol{S}$ and $\boldsymbol{M}_N^\pi = \boldsymbol{T}$. The final term is independent of π, so $E(\pi)$ and $\mathrm{AUPIC}_{\boldsymbol{T}}^{\log}(\pi)$ have the same minimizers. □

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

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

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

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

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 $p_{\boldsymbol{T}} = p_{\mathrm{data}}$ on $\mathcal{D}_{\mathrm{cal}}$.

Proof. For each k, define

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

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

$$\begin{aligned}
\left|\mathrm{AUPIC}^{\log}(\pi) - \mathrm{AUPIC}_{\boldsymbol{T}}^{\log}(\pi)\right| &= \left|\sum_{k=1}^{N} d_{\pi_k} \frac{(a_{k-1} - b_{k-1}) + (a_k - b_k)}{2}\right| \\
&\le \sum_{k=1}^{N} d_{\pi_k} \frac{|a_{k-1} - b_{k-1}| + |a_k - b_k|}{2} \\
&\le \eta \sum_{k=1}^{N} d_{\pi_k} \\
&= (|\boldsymbol{T}| - |\boldsymbol{S}|)\,\eta.
\end{aligned}$$

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

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

multiplying the bounds for $k-1$ and k by $d_{\pi_k}/2 \ge 0$ and summing over $k = 1, \dots, N$ yields

$$\begin{aligned}
e^{-\eta} \sum_{k=1}^{N} d_{\pi_k} \frac{\mathrm{PPL}_{\boldsymbol{T}}(\boldsymbol{M}_{k-1}^\pi) + \mathrm{PPL}_{\boldsymbol{T}}(\boldsymbol{M}_k^\pi)}{2} &\le \sum_{k=1}^{N} d_{\pi_k} \frac{\mathrm{PPL}_{\mathrm{data}}(\boldsymbol{M}_{k-1}^\pi) + \mathrm{PPL}_{\mathrm{data}}(\boldsymbol{M}_k^\pi)}{2} \\
&\le e^{\eta} \sum_{k=1}^{N} d_{\pi_k} \frac{\mathrm{PPL}_{\boldsymbol{T}}(\boldsymbol{M}_{k-1}^\pi) + \mathrm{PPL}_{\boldsymbol{T}}(\boldsymbol{M}_k^\pi)}{2}.
\end{aligned}$$

Equivalently,

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

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

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

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

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

$$(|\boldsymbol{T}| - |\boldsymbol{S}|) \exp\left(\frac{1}{|\boldsymbol{T}| - |\boldsymbol{S}|}\mathrm{AUPIC}^{\log}(\pi)\right) \le \mathrm{AUPIC}(\pi),$$

with equality if and only if $\log \mathrm{PPL}_{\mathrm{data}}(\boldsymbol{M}_k^\pi)$ is constant in k (assuming $d_i > 0$ for all i). 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

$$\mathrm{AUPIC}^{\log}(\pi) = \sum_{k=1}^{N} d_{\pi_k} \frac{z_{k-1} + z_k}{2}, \qquad \mathrm{AUPIC}(\pi) = \sum_{k=1}^{N} d_{\pi_k} \frac{e^{z_{k-1}} + e^{z_k}}{2}.$$

Both are weighted sums over the $2N$ terms z_{k-1}, z_k ($k = 1, \dots, N$) with nonnegative weights $d_{\pi_k}/2$ that add up to $\sum_{k=1}^{N} d_{\pi_k} = |\boldsymbol{T}| - |\boldsymbol{S}|$. Since the exponential function is convex, Jensen's inequality gives

$$\exp\left(\frac{1}{|\boldsymbol{T}| - |\boldsymbol{S}|} \sum_{k=1}^{N} d_{\pi_k} \frac{z_{k-1} + z_k}{2}\right) \le \frac{1}{|\boldsymbol{T}| - |\boldsymbol{S}|} \sum_{k=1}^{N} d_{\pi_k} \frac{e^{z_{k-1}} + e^{z_k}}{2}.$$

Multiplying both sides by $|\boldsymbol{T}| - |\boldsymbol{S}|$, we obtain

$$(|\boldsymbol{T}| - |\boldsymbol{S}|) \exp\left(\frac{1}{|\boldsymbol{T}| - |\boldsymbol{S}|} \mathrm{AUPIC}^{\log}(\pi)\right) \le \mathrm{AUPIC}(\pi).$$

By strict convexity of the exponential function, equality holds if and only if all z_{k-1}, z_k with $d_{\pi_k} > 0$ coincide. If $d_i > 0$ for all i, this means

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

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.

F KLPatch Update Rule

KLPatch (Algorithm 1) builds a path greedily, one edge at a time. This section compares two natural greedy rules, shows that one of them is optimal when the KL gains of the layers do not interact, and gives an example where the other rule fails. We also show that in the equidistant setting, both rules reduce to the KL rule used in Algorithm 1.

Two greedy rules. Let $A \subseteq \{1, \dots, N\}$ be the set of layers patched so far. The *cheapest-edge rule* follows the outgoing edge of smallest weight,

$$i^\star \in \arg\min_{i \notin A} \; w(\boldsymbol{M}_A \to \boldsymbol{M}_{A\cup\{i\}}) = \arg\min_{i \notin A} \; d_i \, \frac{K(A) + K(A \cup \{i\})}{2},$$

that is, it minimizes the next term of $\mathrm{AUPIC}_{\boldsymbol{T}}^{\log}$. This rule is probably the first one that comes to mind, as it directly optimizes for the cheapest outgoing edge. The *steepest-descent rule* maximizes the KL drop per added parameter instead,

$$i^\star \in \arg\max_{i \notin A} \; \frac{K(A) - K(A \cup \{i\})}{d_i},$$

that is, it follows the edge of steepest slope in the (size, KL) plane. If $d_i = 0$, we set the ratio to $+\infty$ if $K(A) - K(A \cup \{i\}) > 0$ and to $-\infty$ otherwise. Note that the total width $|\boldsymbol{T}| - |\boldsymbol{S}| = \sum_i d_i$ of the interpolation curve is fixed, since every layer is eventually patched. Choosing a narrow step therefore saves nothing; it only postpones the wide steps. What matters is how fast the curve descends, and a lower curve now lowers every remaining trapezoid. The cheapest-edge rule only sees the next trapezoid, whose weight is dominated by the width d_i whenever $K(A)$ is large compared to the drop. We make this precise under a simplifying assumption, under which we show that the steepest-descent role is optimal.

Definition 2 (Additive gains)**.** We say that K has *additive gains* if there exist $g_1, \dots, g_N \in \mathbb{R}$ such that

$$K(A) = K(\emptyset) - \sum_{i \in A} g_i \qquad \text{for all } A \subseteq \{1, \dots, N\}.$$

In words, patching layer i lowers the KL to the teacher by g_i regardless of which other layers are patched. Necessarily $g_i = K(\emptyset) - K(\{i\})$ and $\sum_i g_i = K(\emptyset)$, since $K(\{1, \dots, N\}) = 0$.

Proposition 4 (Steepest descent is optimal under additive gains)**.** *Assume $d_i > 0$ for all i and that K has additive gains. Then:*

(i) $E(\pi) = \sum_{k=1}^{N} g_{\pi_k} \, |\boldsymbol{M}_k^\pi| + \mathrm{const}$*, where the constant does not depend on π.*

(ii) *π minimizes E over $\mathfrak{S}_N$ if and only if $g_{\pi_1}/d_{\pi_1} \geq g_{\pi_2}/d_{\pi_2} \geq \cdots \geq g_{\pi_N}/d_{\pi_N}$.*

(iii) *The steepest-descent rule returns a minimizer of E, and hence, by Proposition 1, a minimizer of $\mathrm{AUPIC}_{\boldsymbol{T}}^{\log}$, for every tie-breaking rule.*

Proof. *(i)* Under additive gains, $K(A_k^\pi) = K(\emptyset) - \sum_{m=1}^{k} g_{\pi_m}$ for $k = 0, \dots, N$. Hence

$$E(\pi) = \sum_{k=1}^{N} d_{\pi_k} \, \frac{K(A_{k-1}^\pi) + K(A_k^\pi)}{2} = (|\boldsymbol{T}| - |\boldsymbol{S}|) \, K(\emptyset) - \frac{1}{2} \sum_{k=1}^{N} d_{\pi_k} \Big(\sum_{m=1}^{k-1} g_{\pi_m} + \sum_{m=1}^{k} g_{\pi_m} \Big).$$

Exchanging the order of summation and using $\sum_{k=m+1}^{N} d_{\pi_k} = |\boldsymbol{T}| - |\boldsymbol{M}_m^\pi|$ and $\sum_{k=m}^{N} d_{\pi_k} = |\boldsymbol{T}| - |\boldsymbol{M}_m^\pi| + d_{\pi_m}$,

$$\sum_{k=1}^{N} d_{\pi_k} \Big(\sum_{m=1}^{k-1} g_{\pi_m} + \sum_{m=1}^{k} g_{\pi_m} \Big) = \sum_{m=1}^{N} g_{\pi_m} \, (2|\boldsymbol{T}| - 2|\boldsymbol{M}_m^\pi| + d_{\pi_m}) \, .$$

Therefore

$$E(\pi) = \sum_{m=1}^{N} g_{\pi_m} \, |\boldsymbol{M}_m^\pi| + \Big[\, (|\boldsymbol{T}| - |\boldsymbol{S}|) \, K(\emptyset) - |\boldsymbol{T}| \sum_{i=1}^{N} g_i - \frac{1}{2} \sum_{i=1}^{N} g_i d_i \Big],$$

and the bracket does not depend on π.

(ii) Let π satisfy $\pi_k = i$ and $\pi_{k+1} = j$, and let π' be obtained from π by swapping i and j. The models at all steps other than k and $k+1$ are the same under π and π', so by (i)

$$E(\pi) - E(\pi') = g_i |\boldsymbol{M}_k^\pi| + g_j |\boldsymbol{M}_{k+1}^\pi| - g_j |\boldsymbol{M}_k^{\pi'}| - g_i |\boldsymbol{M}_{k+1}^{\pi'}|.$$

Since $|\boldsymbol{M}_k^\pi| = |\boldsymbol{M}_{k-1}^\pi| + d_i$, $|\boldsymbol{M}_k^{\pi'}| = |\boldsymbol{M}_{k-1}^\pi| + d_j$ and $|\boldsymbol{M}_{k+1}^\pi| = |\boldsymbol{M}_{k+1}^{\pi'}| = |\boldsymbol{M}_{k-1}^\pi| + d_i + d_j$, this gives

$$E(\pi) - E(\pi') = g_j d_i - g_i d_j.$$

Hence $E(\pi) > E(\pi')$ if and only if $g_i/d_i < g_j/d_j$. Now let π^* be a minimizer of E, which exists since $\mathfrak{S}_N$ is finite. If two consecutive layers $\pi_k^* = i$, $\pi_{k+1}^* = j$ satisfied $g_i/d_i < g_j/d_j$, swapping them would strictly decrease E, a contradiction. So π^* is sorted by nonincreasing ratio g/d. Conversely, let π be any permutation sorted by nonincreasing ratio. Then π and π^* have the same ratio at every step, and π can be obtained from π^* by successively swapping consecutive layers with equal ratio. Each such swap changes E by $g_j d_i - g_i d_j = 0$, so $E(\pi) = E(\pi^*)$ and π is a minimizer.

(iii) Under additive gains, $K(A) - K(A \cup \{i\}) = g_i$ for every A and $i \notin A$. The steepest-descent criterion at any step is therefore g_i/d_i, independently of A, and the rule patches at each step a remaining layer with the largest ratio. Its output is sorted by nonincreasing ratio, whatever the tie-breaking, and is a minimizer of E by (ii). By Proposition 1 it is also a minimizer of $\mathrm{AUPIC}_{\boldsymbol{T}}^{\log}$. □

Example. Let $N = 2$ with block sizes $d_1 = 1$, $d_2 = 2$ and additive gains $g_1 = 1$, $g_2 = 4$, so that $K(\emptyset) = 5$, $K(\{1\}) = 4$, $K(\{2\}) = 1$ and $K(\{1,2\}) = 0$. The cheapest-edge rule compares $w(\boldsymbol{M}_\emptyset \to \boldsymbol{M}_{\{1\}}) = 1 \cdot \frac{5+4}{2} = 4.5$ with $w(\boldsymbol{M}_\emptyset \to \boldsymbol{M}_{\{2\}}) = 2 \cdot \frac{5+1}{2} = 6$ and patches layer 1 first, giving $E = 4.5 + 2 \cdot \frac{4+0}{2} = 8.5$. The steepest-descent rule compares $g_1/d_1 = 1$ with $g_2/d_2 = 2$ and patches layer 2 first, giving $E = 6 + 1 \cdot \frac{1+0}{2} = 6.5$, which is optimal by Proposition 4. The first edge of the steepest-descent path is more expensive, but it lowers the curve for the entire remaining size range. For these block sizes, the cheapest-edge rule patches layer 1 first whenever $g_1 > 0$, however small g_1 and however large g_2, whereas this is optimal only if $g_1 \geq g_2/2$.

Remark 2 (Equidistant block sizes). If all blocks have the same size, $d_i = c > 0$ for all i, then c and $K(A)$ are common to all candidates $i \notin A$, and both rules reduce to

$$i^\star \in \arg\min_{i \notin A} K(A \cup \{i\}),$$

that is, patch the layer whose patched model has the smallest KL divergence to the teacher. In the equidistant setting the two rules therefore reduce to the KL rule used in Algorithm 1.

Remark 3 (Non-additive gains in practice). Under additive gains the steepest-descent criterion does not depend on A, so the non-iterative variant KLInitial of Appendix L.2, which ranks the layers once by $K(\{i\})$, returns the same order as KLPatch whenever the block sizes are equal. The gap between KLPatch and KLInitial observed in Appendix L.2 thus shows that the gains are not additive for the models we study, and that the iterative recomputation in Algorithm 1 is needed. Without additivity, no rule that looks only one step ahead is optimal in general, since the best first patch can depend on interactions that are only visible after further patching.

G 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.

H Empirically Validating Theoretical Assumptions

In this section, we empirically validate the assumptions made in Section 4 and Appendix E.

H.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 5.1 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 7). This indicates that distributions of interpolated models that are similar to the distribution of the teacher are also similar to the data distribution. Figure 8 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 9 demonstrates that KL divergence on the Pile calibration set correlates with Wikitext perplexity, and Figure 10 shows that KL path length computed on the Pile calibration set is correlated with AUPIC.

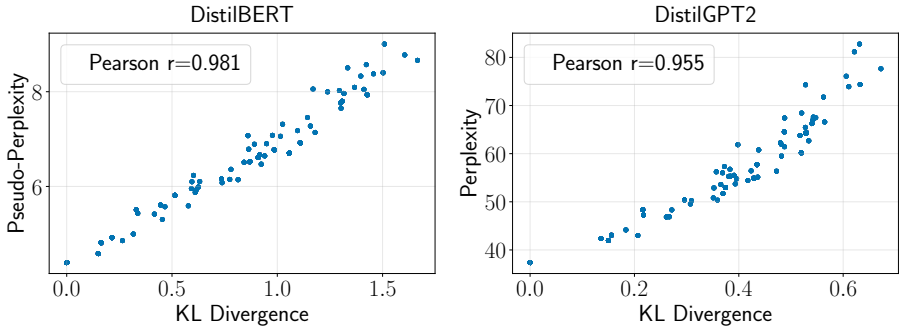


Figure 7: **KL divergence on Wikitext versus Wikitext perplexity.** When KL divergence is computed on Wikitext, there is a strong correlation between the KL divergence between interpolated models and the teacher and Wikitext perplexity for DistilBERT and DistilGPT2 models.

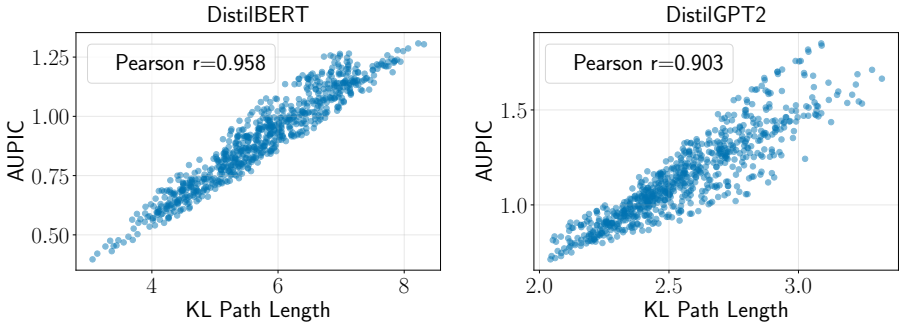


Figure 8: **KL Path Length on Wikitext versus Wikitext AUPIC.** For all possible patching orders, when KL divergence and perplexity are both computed on Wikitext, the KL path length in the interpolation graph correlates strongly with AUPIC.

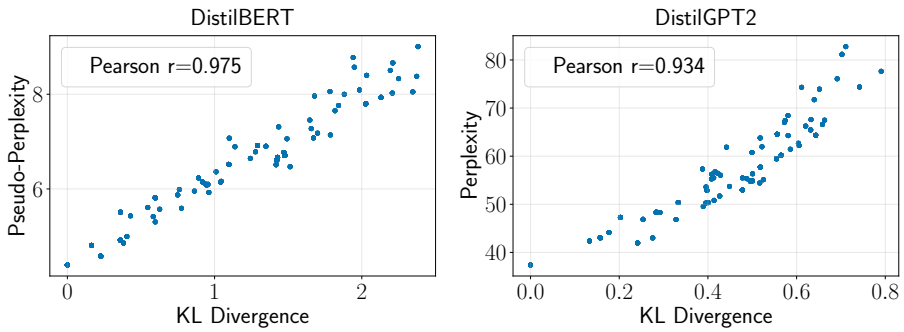


Figure 9: **KL divergence on the Pile versus Wikitext perplexity.** When KL divergence is computed on a 64 example calibration set from the Pile, there is a strong correlation between the KL divergence between interpolated models and the teacher and Wikitext perplexity for DistilBERT and DistilGPT2 models.

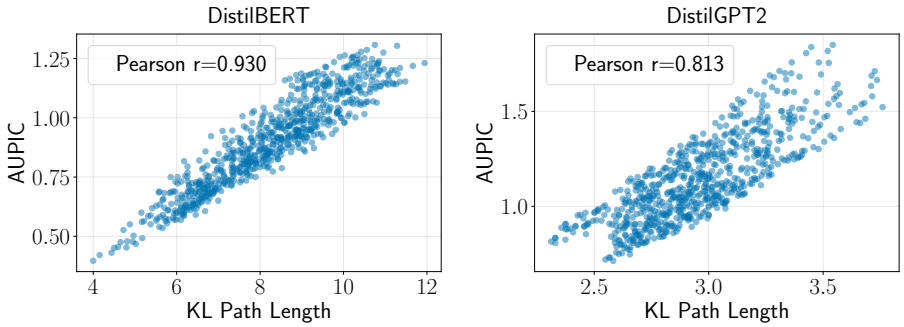


Figure 10: **KL Path Length on the Pile versus Wikitext AUPIC.** For all possible patching orders, when KL divergence is computed on a 64 example calibration set of the Pile and perplexity is computed on Wikitext, the KL path length in the interpolation graph correlates strongly with AUPIC.

H.2 Optimal Patching Order Closely Approximates Optimal Interpolation

In Section 4.1, 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 5.1 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 5.3.

Results. Figure 11 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 2). For DistilGPT2, the optimal permutation is slightly higher in perplexity than the optimal interpolation for small model sizes (Figure 11), but is very close in AUPIC to the optimal interpolation (Table 2).

For larger models, while it is computationally infeasible to compute the exact optimal permutation and interpolation, Figure 12 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 13. Similarly, the maximum AUIC across permutations is close to the maximum AUIC interpolation (Table 3). 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 2: **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.

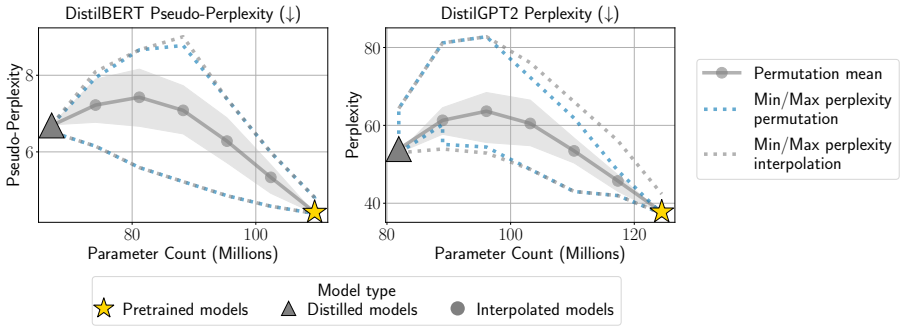


Figure 11: **Comparison of minimum AUPIC permutation vs interpolation for DistilBERT and DistilGPT2.** The minimum permutation produces the same perplexity as the minimum interpolation for DistilBERT and perplexity close to the minimum interpolation for DistilGPT2. Shaded bands show the (25-75) inter-quartile 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 3: **AUIC results for best performing interpolations vs permutations.** The maximum AUIC permutation has AUIC close to that of the maximum AUIC interpolation.

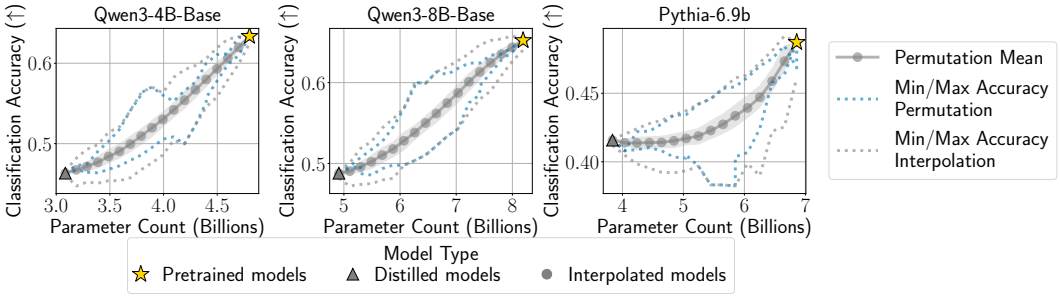


Figure 12: **Comparison of maximum AUIC permutation vs interpolation for Qwen3-4B-Base, Qwen3-8B-Base, and Pythia-6.9b over 200 random orderings.** Across models, the maximum AUIC permutation has classification accuracy very close to that of the maximum interpolation. Shaded bands show the (25-75) inter-quartile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

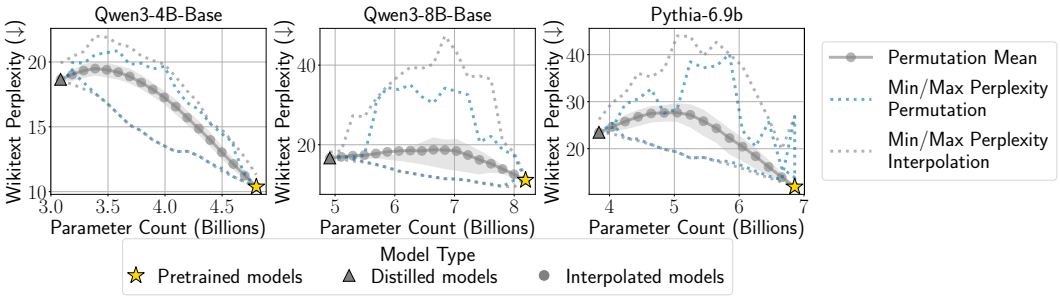


Figure 13: **Comparison of minimum AUPIC permutation vs interpolation for Qwen3-4B-Base, Qwen3-8B-Base, and Pythia-6.9b over 200 random orderings.** Across models, the minimum AUPIC permutation has perplexity very close to that of the minimum interpolation. Shaded bands show the (25-75) inter-quartile range over all 200 orderings.

H.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 6 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 5.1. 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 4 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 4: **AUPIC results for different patching paths.** The KLPatch path has an AUPIC close to that of the global shortest KL path.

In Figure 14, 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.

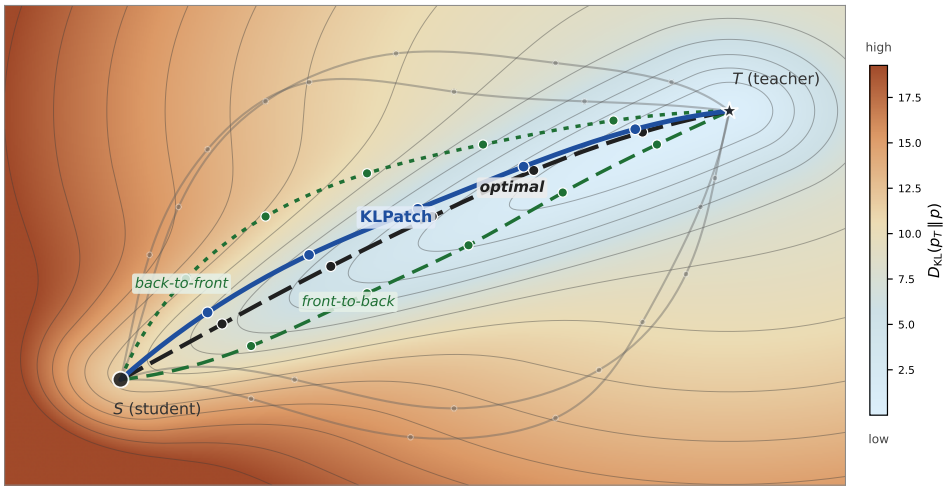


Figure 14: **KLPatch finds patching orders that are close to optimal:** Visual representation of the optimal patching order between the student S and teacher T along the KL divergence to the teacher model, compared to the KLPatch order, front-to-back and back-to-front, and random permutations.

I Additional Permutation Results

We present extended results from Sections 5.1 and 5.3 on permutations of patching orders for model size interpolation. Figures 15-18 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.

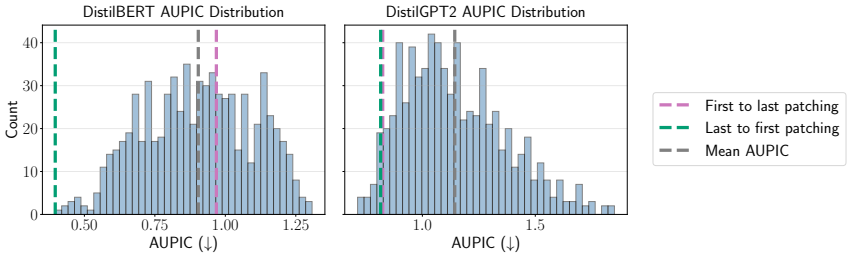


Figure 15: **AUPIC distribution across patching order permutations for DistilBERT and DistilGPT2.** Across the 720 orderings, last-to-first patching is at or near the optimum on both models, suggesting it is a simple and reliable default recipe for model size interpolation.

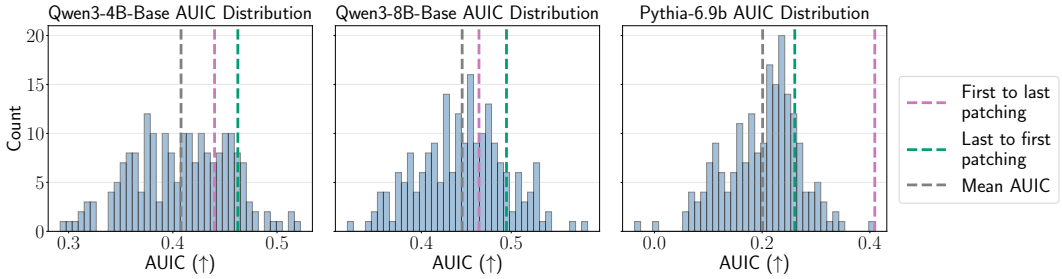


Figure 16: **Distribution of AUIC for different patching orders.** Across 200 orderings, first to last and last to first patching have relatively high AUIC, but last-to-first patching does not have the highest AUIC for Qwen models despite having the lowest AUPIC.

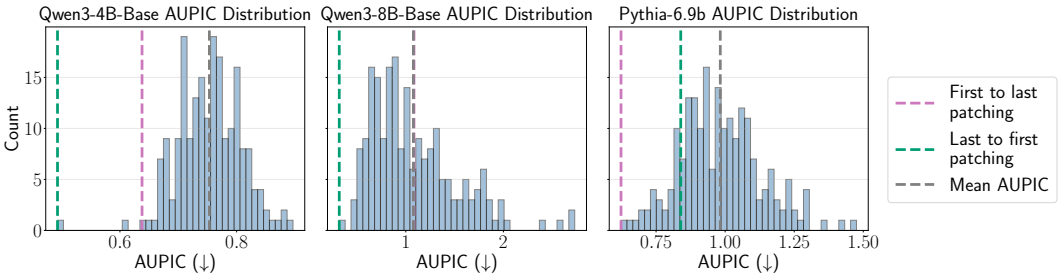


Figure 17: **Distribution of AUPIC for different patching orders.** Across the 200 orderings, either last to first or first to last patching often yields the highest AUPIC, suggesting it is a simple and reliable default recipe for model size interpolation.

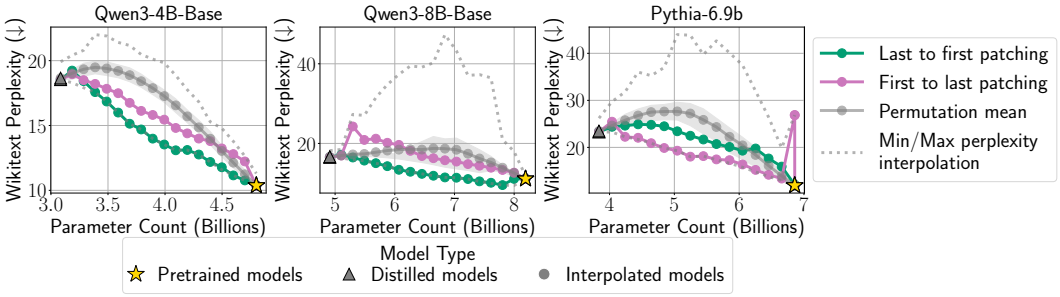


Figure 18: **Wikitext perplexity for different patching orders.** Across 200 orderings, either last to first patching and first to last patching achieve smooth interpolation performance, suggesting that they are generalizable recipes that work well across models. On the other hand, we observe that a naive patching order can sometimes result in poor interpolation performance. Shaded bands show the (25-75) inter-quartile range over all 200 orderings.

J Additional KLPatch Results

We present additional results for KLPatch compared to baselines, estimated runtimes, and performance on generation tasks and Wikitext.

J.1 Additional Baselines

Setup. We reproduce results from Kangaslahti et al. [2026] to provide additional baselines for boomerang distillation and better contextualize the performance of KLPatch. We compare our results to naive layer pruning, where we iteratively remove every other layer from the teacher model, starting from layer 2. This corresponds to intermediate-sized models with the same initialization as those from last to first patching, but without distilling the student model. We also provide a random initialization baseline, where the student is randomly initialized, then distilled with the same training setup and budget as in Kangaslahti et al. [2026]. This baseline shows the performance of interpolation from a standard distilled student model.

Results. Figure 19 shows that all of the models constructed with boomerang distillation significantly outperform the two baselines, confirming that model size interpolation yields strong gains over standard pruning and distillation approaches. As KLPatch improves performance further compared to naive sequential patching strategies, it provides further gains in the model size interpolation setting.

J.2 KLPatch Trajectories are Best- or Near-Best-Performing

Setup. We compute the AUIC and AUPIC for the 200 random orderings from Section 5.3, 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

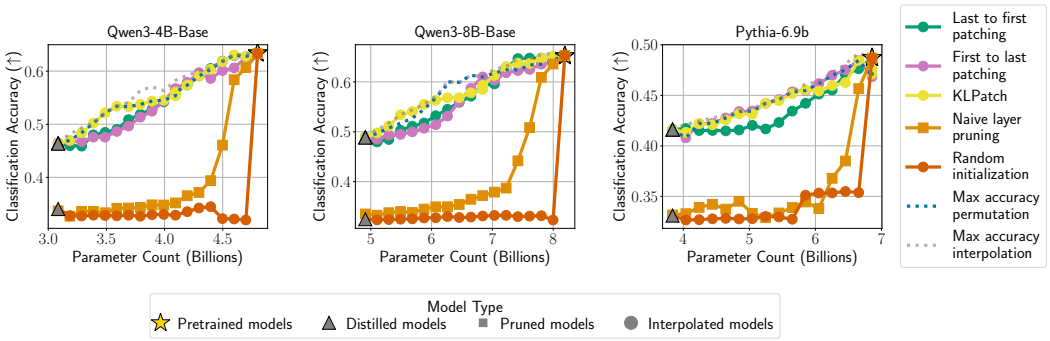


Figure 19: **Model size interpolation significantly outperforms layer pruning and interpolation from a standard distilled model.** KLPatch produces further gains over naive last to first and first to last patching in boomerang distillation.

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 H.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 5-6). 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 5: **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.

J.3 KLPatch Significantly Reduces Runtime

We compare the runtime required for KLPatch ($O(N^2)$) to that of exhaustive search ($O(2^N)$) for each of the three models tested. We estimate exhaustive search runtime by recording the time it takes to compute KL divergence to the teacher on the calibration set for each intermediate model size and multiplying it by the number of evaluations needed at that model size. As shown in Table 7, KLPatch achieves over $548\times$ speedup over exhaustive search for all three models, and the speedup increases as the number of student layers increases (from 17 student layers in the Pythia model to 19 in both Qwen models). Furthermore, exhaustive search takes a computationally infeasible amount of time as the number of student layers increases.

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 6: **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.

Model	Exhaustive Search Runtime (minutes)	KLPatch Runtime (minutes)	Speedup From KLPatch
Qwen3-4B-Base	64302.079	37.337	1722.209×
Qwen3-8B-Base	65327.869	38.097	1714.785×
Pythia-6.9b	14134.826	25.760	548.704×

Table 7: **KLPatch produces over $500 - 1700\times$ speedup over exhaustive search.** We estimate exhaustive search runtime by computing a single KL divergence evaluation pass on each intermediate model size and multiplying it by the number of evaluations needed at that model size.

J.4 Generation Tasks

Figure 20 shows that KLPatch closely tracks the best performing permutation and 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.

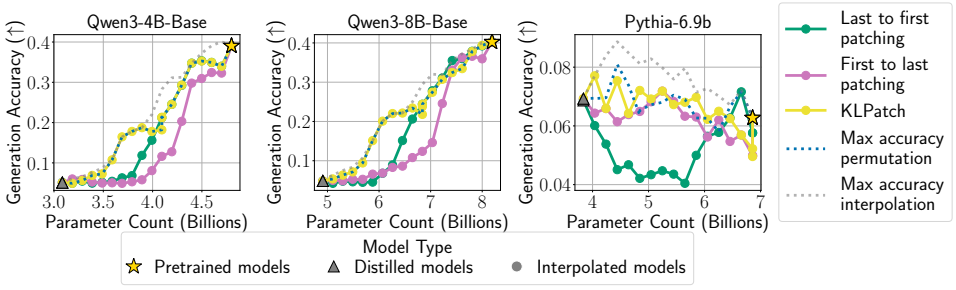


Figure 20: **KLPatch for Qwen3-4B-Base, Qwen3-8B-Base, and Pythia-6.9b** Across models, KLPatch tracks the max accuracy permutation and outperforms or tracks the last to first patching on generation tasks.

J.5 Wikitext Perplexity

Figure 21 shows that KLPatch often tracks closely to the best performing patching order among last to first and first to last patching orders on Wikitext.

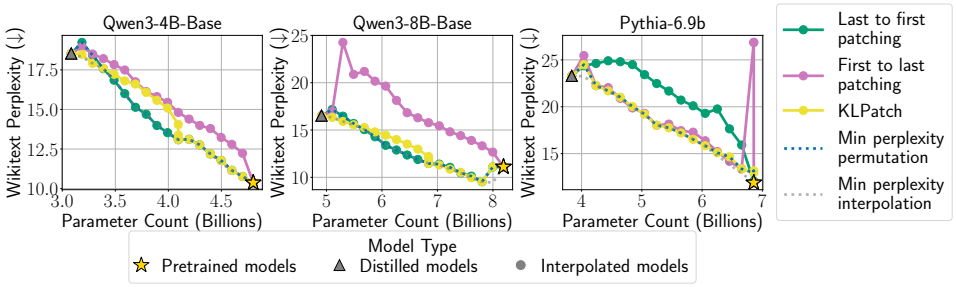


Figure 21: **KLPatch for Qwen3-4B-Base, Qwen3-8B-Base, and Pythia-6.9b** Across models, KLPatch tracks the last to first patching on perplexity.

J.6 Per-Task Performance

We report a per-task breakdown of the performance of KLPatch compared to last-to-first and first-to-last patching in Tables 8 and 9. We find that KLPatch is the most reliable across classification and generation results and achieves best or second-best performance on all tasks for all models other than MMLU for Pythia-6.9b.

Model	Method	ARC-challenge	ARC-easy	BoolQ	HellaSwag	MMLU	OpenBookQA	PIQA	RACE	RTE	WinoGrande
Qwen3-4B-Base	Last to first patching	0.577	0.618	0.441	0.457	0.444	0.381	0.472	0.303	0.469	0.351
	First to last patching	0.388	0.338	0.446	0.433	0.456	0.276	0.400	0.273	0.593	0.526
	KLPatch	0.599	0.543	0.618	0.436	0.544	0.308	0.439	0.478	0.601	0.466
Qwen3-8B-Base	Last to first patching	0.636	0.650	0.454	0.476	0.418	0.592	0.570	0.487	0.425	0.451
	First to last patching	0.384	0.405	0.447	0.435	0.525	0.244	0.483	0.277	0.572	0.501
	KLPatch	0.661	0.613	0.690	0.455	0.549	0.422	0.484	0.520	0.601	0.584
Pythia-6.9b	Last to first patching	0.310	0.279	-0.880	0.340	0.602	0.271	0.281	0.308	0.279	0.159
	First to last patching	0.379	0.317	1.031	0.369	0.616	0.242	0.357	0.386	0.542	0.596
	KLPatch	0.356	0.324	1.105	0.348	0.276	0.288	0.346	0.310	0.358	0.540

Table 8: **Per-task classification AUIC.** KLPatch has the most consistent performance across tasks in terms of classification AUIC.

Model	Method	GSM8K	MATH	IFEval
Qwen3-4B-Base	Last to first patching	0.437	0.408	0.191
	First to last patching	0.259	0.173	0.316
	KLPatch	0.520	0.386	0.213
Qwen3-8B-Base	Last to first patching	0.464	1.080	0.213
	First to last patching	0.306	0.218	0.386
	KLPatch	0.572	1.191	0.251
Pythia-6.9b	Last to first patching	0.183	0.001	0.143
	First to last patching	0.309	0.001	0.173
	KLPatch	0.360	0.001	0.184

Table 9: **Per-task generation AUIC.** KLPatch has the most consistent generation AUIC across tasks.

K 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 22 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 also see in Table 10 that Llama-3.2-3B has very low cosine similarity between the outputs of the first two layers compared to the other

models. This supports model-specific layer-importance differences that we suspect 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.

Model	Layer 1 & 2 Cosine Similarity
Llama-3.2-3B	0.07
Qwen3-4B-Base	0.81
Qwen3-8B-Base	0.90
Pythia-6.9b	0.96

Table 10: **Cosine Similarity Comparison Across Models.** We compute the average cosine similarity between the output activations of layers 1 and 2 of each model on our calibration set. Llama-3.2-3B has much lower cosine similarity between layer 1 and 2 than the other models tested, suggesting model-specific layer importance differences.

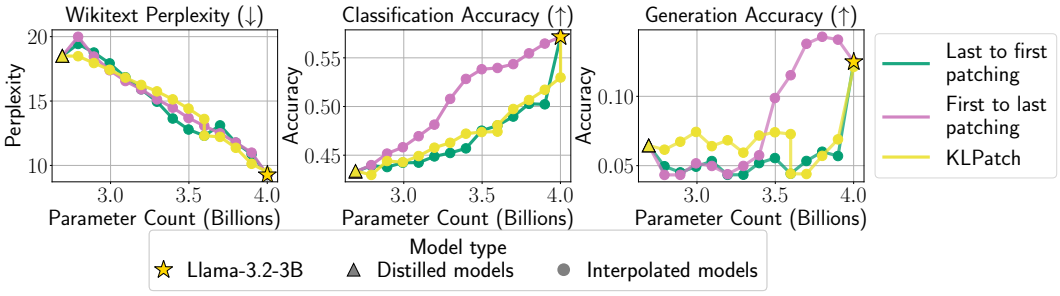


Figure 22: **Standard KLPatch for Llama-3.2-3B.** The standard KLPatch algorithm does not recover the optimal patching order.

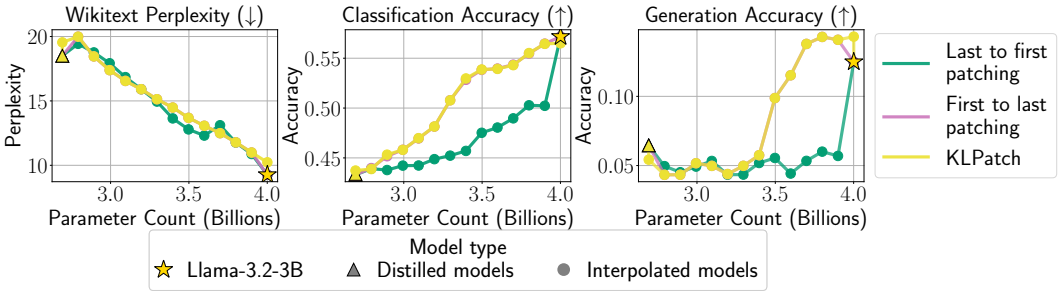


Figure 23: **KLPatch for Llama-3.2-3B by first patching Layer 1.** Enforcing layer 1 as the first patched layer allows KLPatch to recover the optimal patching order and improves performance by avoiding an initially suboptimal patching choice.

L KLPatch Ablations

We show additional ablations on the KLPatch algorithm, including using metrics other than KL Divergence (L.1), a non-iterative greedy approach (L.2), additional search algorithms (L.3), and calibration set ablations (L.4).

L.1 KLPatch Performs Best with KLDivergence

Setup. We compare the standard KLPatch setup with other metrics used to compute ℓ_i in Algorithm 1 in place of KL divergence. Specifically, we test last-layer cosine similarity between the student

and teacher, $\mathcal{L}_{\cos}^{(N)}(x_{<j}; \boldsymbol{\theta_T}, \boldsymbol{\theta_S})$, which we denote as cosine patching, as well as perplexity, which we call perplexity patching. We compute both of these metrics on the same validation set used to calculate KL Divergence in the standard KLPatch setup.

Results. KLPatch produces the strongest interpolation behavior across models and classification and generation tasks (Figures 24-26). Both KLPatch and perplexity patching signficantly outperform cosine patching. Perplexity patching has somewhat similar performance to KLPatch, especially on classification tasks, as for strong teachers, we expect that KL divergence to the teacher provides very similar signal to perplexity. However, while perplexity patching is a strong baseline, KLPatch provides the most consistent results across models, especially in the generation setting.

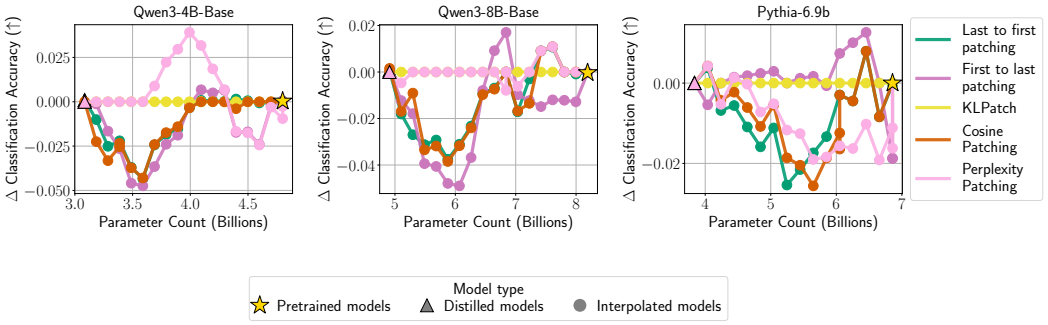


Figure 24: **Change in classification accuracy with respect to KLPatch for ablated metrics.** Ablating the metrics used to compute ℓ_i in Algorithm 1 produces interpolation curves worse than or on par with KLPatch on classification tasks.

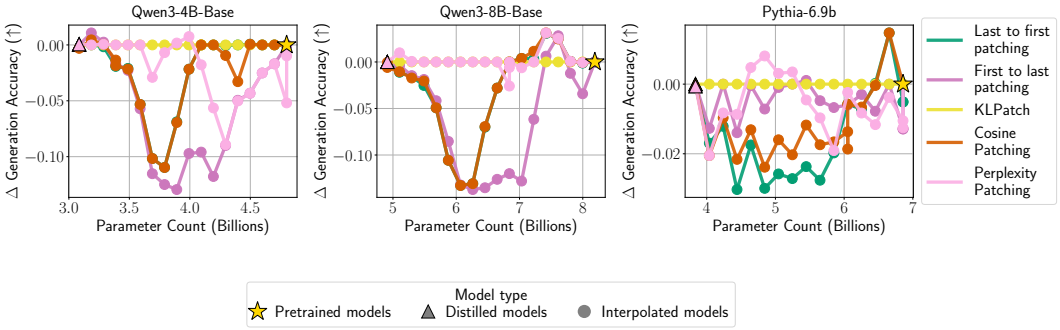


Figure 25: **Change in generation accuracy for KLPatch with ablated metrics.** For generation tasks, cosine patching and perplexity patching baselines have worse or similar performance to KLPatch across models.

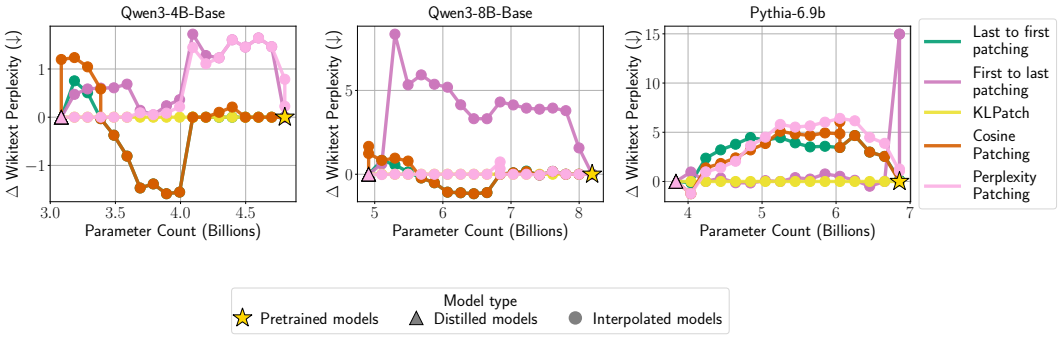


Figure 26: **Change in wikitext perplexity with ablated metrics for KLPatch.** KLPatch has similar or better wikitext perplexity to other metrics.

L.2 Iterative KLPatch Outperforms Non-Iterative Approaches

Setup. We test the standard KLPatch setup against non-iterative baselines. Specifically, we compare against a version of KLPatch, which we call KLInitial, that performs a single iteration of KLPatch and orders all of the layers in terms of their KL ℓ_i when only that single layer is patched. We also include baselines that evaluate layer importance without patching any layers. Specifically, we rank student layer importance using the use the block influence metric proposed by [Men et al., 2024], which calculates the cosine similarity between the inputs and outputs of a given layer, as layers with higher input/output cosine similarity make smaller changes to the residual stream. We then patch layers in order of highest to lowest importance. We also include the logit lens approach [nostalgebraist, 2020], which is a popular method that maps activations to the vocabulary space by multiplying them with the unembedding matrix, as an additional baseline. For each student layer, we use the logit lens technique to compute the KL divergence between the mapped output activations of the student layer and its corresponding teacher block. We then patch student layers in order of highest to lowest KL divergence to their teacher blocks.

Results. We report the results comparing KLPatch to non-iterative approachs in Figures 27-29. For classification tasks and perplexity, KLPatch has performance similar to that of KLInitial for smaller interpolated model sizes and outperforms KLInitial for larger interpolated model sizes. For generation tasks, KLPatch outperforms KLInitial across model sizes. This indicates that information about the compounding effect of patching intermediate layers (which we obtain from KLPatch but not KLInitial) is especially useful for generation tasks and for intermediate models with more layers patched. Block influence and logit lens underperform KLPatch and KLInitial, especially for smaller interpolated model sizes, indicating that it is challenging to obtain patching orders with comparable performance to KLPatch from initial importance or alignment information alone.

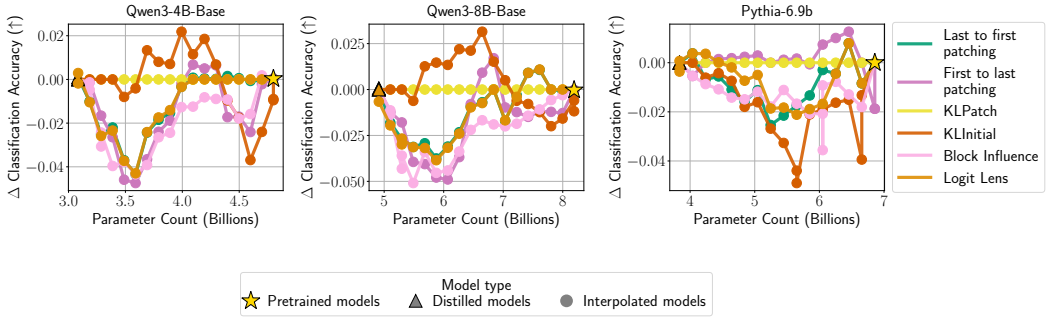


Figure 27: **Difference in classification accuracy between non-iterative approaches and KLPatch.** KLPatch has comparable or better performance than non-iterative approaches on classification tasks.

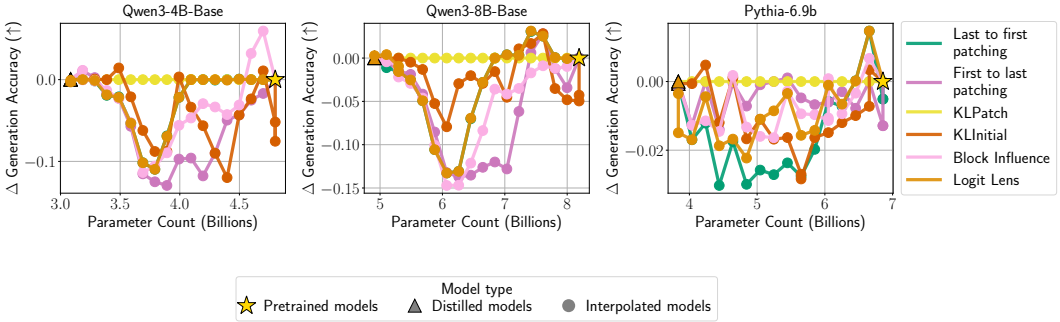


Figure 28: **Difference in generation accuracy between non-iterative approaches and KLPatch.** KLPatch has better performance than non-iterative approaches on generation tasks.

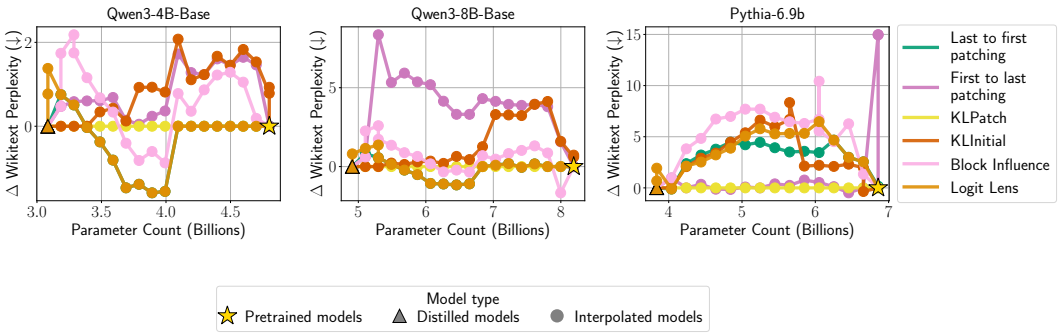


Figure 29: **Difference in wikitext perplexity between non-iterative approaches and KLPatch.** KLPatch has similar or better wikitext perplexity to non-iterative approaches.

Model	Metric	Classification AUIC (↑)	Wikitext AUPIC (↓)
Qwen3-4B-Base	Beam Search (Beam Width 2)	0.531	0.541
	Beam Search (Beam Width 3)	0.534	0.538
	1-Step Lookahead	0.531	0.541
	KLPatch	0.533	0.539
Qwen3-8B-Base	Beam Search (Beam Width 2)	0.505	0.784
	Beam Search (Beam Width 3)	0.505	0.784
	1-Step Lookahead	0.569	0.361
	KLPatch	0.571	0.357
Pythia-6.9b	Beam Search (Beam Width 2)	0.362	0.596
	Beam Search (Beam Width 3)	0.386	0.577
	1-Step Lookahead	0.362	0.596
	KLPatch	0.379	0.567

Table 11: **Spending more compute on search yields limited gains.** We compare more computationally expensive search algorithms to KLPatch and find limited or no gains in performance, so KLPatch provides a strong option for interpolation while requiring limited compute.

L.3 Additional search algorithms

In Table 11 below, we compare KLPatch to beam search and 1-step lookahead algorithms. We note that these search algorithms are computationally much more expensive than KLPatch, so they trade search compute for exploring a larger set of strong patching orders. We find that spending additional compute on search results in very small or no gains in performance and in some cases causes overfitting to the calibration set, so KLPatch produces near-best performance with limited compute.

L.4 Ablating the calibration set

We test varying calibration set sizes $|\mathcal{D}_{\text{cal}}|$ used for evaluating KL divergence in the KLPatch algorithm and report the mean and standard deviation over 5 random seeds in Table 12. We find that KLPatch performance stabilizes around $|\mathcal{D}_{\text{cal}}| = 64$ across models and downstream tasks, so we use this calibration set size in Section 6.2.

Model	$\lvert\mathcal{D}_{\text{cal}}\rvert$	Classification AUIC	Generation AUIC	AUPIC
Qwen3-4B-Base	16	0.486 ± 0.066	0.351 ± 0.112	0.575 ± 0.042
	32	0.534 ± 0.017	0.428 ± 0.021	0.553 ± 0.030
	64	0.533 ± 0.001	0.442 ± 0.001	0.539 ± 0.001
	128	0.533 ± 0.001	0.442 ± 0.002	0.539 ± 0.001
	256	0.533 ± 0.001	0.442 ± 0.002	0.539 ± 0.000
	512	0.533 ± 0.001	0.442 ± 0.002	0.539 ± 0.000
	1024	0.533 ± 0.001	0.442 ± 0.002	0.539 ± 0.000
Qwen3-8B-Base	16	0.515 ± 0.079	0.414 ± 0.108	0.611 ± 0.333
	32	0.553 ± 0.027	0.471 ± 0.069	0.449 ± 0.180
	64	0.570 ± 0.001	0.502 ± 0.002	0.357 ± 0.007
	128	0.571 ± 0.002	0.503 ± 0.001	0.361 ± 0.000
	256	0.571 ± 0.002	0.503 ± 0.001	0.361 ± 0.000
	512	0.570 ± 0.001	0.503 ± 0.001	0.361 ± 0.000
	1024	0.569 ± 0.000	0.503 ± 0.000	0.361 ± 0.000
Pythia-6.9B	16	0.376 ± 0.011	0.067 ± 0.002	0.5666 ± 0.066
	32	0.378 ± 0.007	0.067 ± 0.003	0.565 ± 0.023
	64	0.379 ± 0.006	0.067 ± 0.002	0.567 ± 0.017
	128	0.379 ± 0.004	0.067 ± 0.001	0.567 ± 0.016
	256	0.379 ± 0.001	0.067 ± 0.001	0.567 ± 0.001
	512	0.379 ± 0.000	0.067 ± 0.000	0.567 ± 0.000
	1024	0.379 ± 0.000	0.067 ± 0.000	0.567 ± 0.000

Table 12: **KLPatch performance summary with varying calibration set size.** Values are reported as mean $\pm$ standard deviation, computed over 5 random seeds.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We support the theoretical and empirical claims made in the abstract and introduction in Sections 4, 5, and 6.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of the paper in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We provide the main theoretical claim in Section 4 and provide the complete proofs in Appendix E and F.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: In Section 5 and Section 6, each experiment is accompanied by the setup required to reproduce the experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide a URL to the codebase.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: In each experiment, in Section 5 and 6, we provide dataset and model details in the Setup paragraph.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: In Section 5, we report the 25-75 inter-quartile range over all the orderings and the distribution of the performance of the model size interpolation.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We include all the details on compute and walltime in Appendix G.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This research conforms to the NeurIPS Code of Ethics. The work does not involve human subjects. We use only publicly available datasets for evaluation and do not introduce new identifiable information. We discuss the limitations (Appendix A) and broader impact (Appendix C) of our work, including any potential harms we anticipate.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the potential societal impacts of this work in Appendix C.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This work does not release new models or scraped datasets.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All the code and models used were obtained through HuggingFace.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This work does not release any new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This work does not involve any human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This work does not involve any human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: We discuss our LLM usage in Appendix D.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.