Thinking at the Right Size: Amortized Distillation Across Post-Trained LLMs

Yan Zhou[1], 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

Correspondence: terryzhou@fas.harvard.edu

Abstract

Practical deployment of large language models (LLMs) requires families of post-trained variants—instruction-tuned, reasoning-tuned, and chat-style models—each at multiple sizes to meet diverse latency and memory budgets. Producing each (variant, size) pair independently is prohibitive, so model families typically span only a handful of coarse-grained sizes per post-trained variant. Boomerang distillation (Kangaslahti et al., 2026a) reduces this cost along the size axis for base models. Through model size interpolation, it constructs models of intermediate sizes from a single teacher-student pair without additional training. However, it still treats each post-trained variant as a separate object of optimization. We introduce **ADAPT**—Amortized Distillation Across Post-Trained LLMs—a framework for amortizing distillation across both axes of a model family: size and post-training variant, producing $L \times K$ models for L interpolated sizes across K post-trained variants with a *single* distillation run. ADAPT combines two components. First, a two-phase distillation procedure constructs post-trained students through pre-training alignment and supervised fine-tuning distillation, enabling smooth size–performance interpolation on generation and reasoning tasks. Second, weight-delta initialization approximates this construction across post-trained variants by transferring the distillation-induced weight change from the base model to students initialized from different post-trained variants. The resulting continuum of interpolated models also enables adaptive model-size selection at inference time, improving the compute–accuracy trade-off for long-form reasoning tasks.[1]

1 Introduction

Deployed LLM applications rely overwhelmingly on post-trained models—instruction-tuned, reasoning-tuned, and chat-style variants—rather than base pre-trained models. They also span a wide range of compute environments, from mobile devices to personal laptops to on-premise servers with multi-GPU clusters, with correspondingly diverse latency and memory constraints (Huyen, 2022; Narayan et al., 2025). Practical use thus requires families of models that vary along two axes: size, to meet deployment compute constraints, and post-trained variants, to support different downstream capabilities. Training each (variant, size) combination from scratch is computationally infeasible, so model families are typically released only at a few coarse-grained sizes per variant, and only at a few variants per base model (Qwen Team, 2026).

Existing approaches address one of these axes but not both. Task vectors (Ilharco et al., 2023) transfer fine-tuned capabilities across models without additional training, but are constrained to only models of the same size. Boomerang distillation (Kangaslahti et al., 2026a) transfers teacher capability to a fine-grained family of smaller models with a single distillation run. It thus efficiently covers the size axis, though only in the pre-trained setting and leaves the cross-variant axis unaddressed. Whether this strategy extends to post-trained models, and whether size interpolation can be applied zero-shot across multiple post-trained variants of the same base model, are the key questions this paper investigates.

A natural first approach is to apply boomerang distillation (BD) directly to a post-trained teacher: initialize a student from the teacher, distill on pre-training data, and patch the layers back in to create intermediate-sized models. Another approach to address the cross-variant axis is to cross-patch post-trained variants onto the boomerang-distilled base model, hoping that post-trained model capabilities can be transferred through patching alone. We find that neither

*Equal contribution.

[1]Code: https://github.com/dcml-lab/ADAPT.

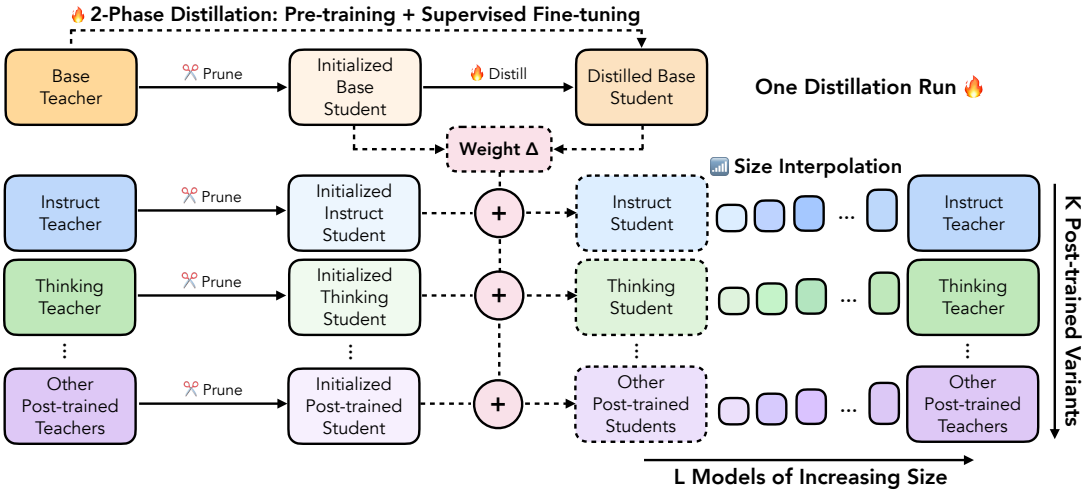

Figure 1: **ADAPT: Amortized Distillation Across Post-Trained LLMs.** We perform two-phase distillation (pre-training + SFT) on a base student once, yielding L interpolated model sizes from a single teacher–student pair. We then compute the distillation-induced weight delta and add it to the initialized students from multiple post-trained teachers. This amortizes distillation for post-trained LLMs: one distillation run can now produce $L \times K$ models across K post-trained variants.

approach works: generation and reasoning performance collapses across small to medium sizes on in-domain and out-of-domain tasks for both setups, despite the post-trained teacher itself achieving strong performance on these tasks (Figure 2). This asymmetry suggests that instruction-following depends on properties of the distillation process that the original BD recipe does not provide.

In response, we introduce **ADAPT**—Amortized Distillation Across Post-Trained LLMs—a framework for constructing post-trained model families across both model size and post-training variant axes. ADAPT combines two-phase distillation with weight-delta initialization. First, two-phase distillation distills a student for each post-trained variant, using both pre-training alignment and supervised fine-tuning distillation for instruction-following and reasoning, enabling smooth model size interpolation without any additional training. We then provide weight-delta initialization as an approximate construction that avoids separate distillation runs by transferring the distillation-induced weight change from the base model to students initialized from different post-trained variants of the same base model. This allows a single base-model distillation run to produce $L \times K$ models: L interpolated model sizes for each of K post-trained variants.

Our experiments show that ADAPT recovers smooth size-performance interpolation on generation and reasoning benchmarks and outperforms standard boomerang distillation and layer pruning baselines at equal compute. The resulting family of intermediate-sized models supports adaptive inference: routing easier inputs to smaller models traces an empirical compute–accuracy Pareto frontier that dominates any single fixed model. To understand why a single base-model distillation transfers to multiple post-trained variants, we examine the underlying weight-space geometry. Empirically, base and post-trained models—and their corresponding distilled students—are connected by low-loss linear interpolation paths, a form of smooth weight interpolation that is preserved through distillation. The distillation update thus operates within a connected region of weight space shared by base and post-trained models, providing a geometric explanation of why weight-arithmetic transfer succeeds in this setting.

In summary, our contributions are:

- We introduce ADAPT, a framework for amortizing distillation runs to create interpolated post-trained model families across sizes and post-training variants.
- We show that ADAPT outperforms strong baselines, including boomerang distillation and layer pruning approaches, while enabling improved performance–compute trade-offs through adaptive model-size selection.
- We provide an empirical analysis of size-agnostic smooth weight interpolation, showing that the weight-space structure connecting base and post-trained models is preserved after distillation and helps explain the success of weight-delta initialization.

2 Related Work

Knowledge distillation. Knowledge distillation is an effective technique for training a smaller model using a larger teacher model (Hinton et al., 2015; Sanh et al., 2020). Several LLM families use knowledge distillation to train or post-train smaller models, including Qwen (Yang et al., 2025), DeepSeek (Guo et al., 2025), and Gemma (Team et al., 2025). These smaller LLMs show improved performance when compared to naive training, but require individual distillation runs. In this work, we introduce a two-phase distillation pipeline and a weight-arithmetic procedure that produces a family of post-trained models using a single distillation run.

Post-training LLMs. Post-training is a critical step in the language model training pipeline, in which pre-trained LLMs are trained to follow instructions (Olmo et al., 2025; Guo et al., 2025; Bercovich et al., 2025). Because post-training is expensive, recent work aims to reduce its compute cost by training on smaller but effective datasets (Ye et al., 2025; Nayak et al., 2026), but it still requires training models at every size. Here, we distill a single post-trained student LLM and then create interpolated post-trained LLMs of varying sizes without training additional models, thereby dramatically reducing compute cost.

Model arithmetic. Model arithmetic enables practitioners to combine the skills of multiple models into a single unified model by simply adding or subtracting model weights (Ilharco et al., 2023). Prior work has used model arithmetic for instruction-following (Ilharco et al., 2023), creating multitask models (Yadav et al., 2025), and machine unlearning (Liu et al., 2025). In this work, we use model arithmetic to amortize student distillation by transferring the distillation-induced weight delta from a base student to students initialized from multiple post-trained variants.

Adaptive compute. Adapting LLM compute to input difficulty significantly reduces latency requirements during inference (Taghibakhshi et al., 2026). While recent work has shown promising results for test-time scaling (Muennighoff et al., 2025), many of these approaches require training fine-grained models from scratch (Kusupati et al., 2022) or rely on heuristics such as appending thinking tokens (Muennighoff et al., 2025). In this work, we dynamically adjust inference compute by adapting the model size based on the input difficulty for reasoning and generation tasks.

Layer pruning. Layer pruning removes transformer layers from a trained LLM under an importance criterion, sometimes followed by a healing step (Men et al., 2025; Chen et al., 2025). While effective at preserving classification accuracy, recent analyses show that pruning collapses open-ended generation (Shrestha et al., 2026). ADAPT instead recovers generation capability through a targeted two-phase distillation step, and substantially outperforms layer pruning approaches such as ShortGPT and LLM-Streamline (Appendix H).

3 ADAPT: Amortized Distillation Across Post-Trained LLMs

We begin by summarizing boomerang distillation (Kangaslahti et al., 2026a), a recently proposed method for zero-shot model size interpolation. We then introduce the two versions of ADAPT: ADAPT (distilled), a two-phase distillation framework for constructing interpolated post-trained models, and ADAPT (weight-delta), a weight-delta initialization method that approximates the two-phase distillation process across model variants and constructs multiple post-trained model families from a single distillation run.

3.1 Boomerang Distillation

Boomerang distillation (Kangaslahti et al., 2026a) distills a small student from a teacher in such a way that intermediate models can be constructed by combining student and teacher layers without additional training. We now discuss its three stages.

Student initialization. Let $\boldsymbol{T}$ be the teacher with N layers denoted by $\boldsymbol{\theta}_T$, and $\boldsymbol{S}$ with parameters $\boldsymbol{\theta}_S$ and M layers be the student, where $M < N$. The student is initialized by partitioning the N teacher layers into M blocks of consecutive layers. From each of these blocks, the first layer is used to initialize the corresponding student layer. The student's embedding and LM head are initialized from $\boldsymbol{T}$.

Knowledge distillation. The initialized student is trained on a pre-training corpus such as the Pile (Gao et al., 2020) on a combined loss involving a logit-level knowledge distillation loss, such as KL, and a block-wise alignment loss, such as cosine distance, that pushes each student layer's output toward the outputs of the corresponding block of teacher layers. Specifically, given a training

corpus $\mathcal{D}$ and a sample $x = (x_1, ..., x_L) \in \mathcal{D}$, an index j, and denoting the next-token distributions by $p(\cdot \mid x_{<j})$, the per-token loss $\mathcal{L}_{\boldsymbol{\theta_S}}(x_j)$ is:

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

Here, CE denotes the next-token cross-entropy loss and $\mathcal{L}^{(i)}_{\cos}$ is the cosine distance between the outputs of the ith student layer and the ith teacher block. $\lambda_{\mathrm{KL}} > 0$ and $\lambda_{\cos} > 0$ are hyperparameters that weight the KL and cosine terms relative to cross-entropy.

Student patching. After the distillation stage, boomerang distillation constructs intermediate models by *patching*—i.e., replacing student layers with their corresponding sets of teacher layers. We treat student patching as an important but complementary design choice and use simple front-to-back and back-to-front patching in the main experiments, selecting the better-performing of the two orders for each model (see Appendix J for details).

3.2 ADAPT

3.2.1 ADAPT (Distilled): Two-Phase Post-Trained Distillation

We propose a unified two-phase distillation procedure for model size interpolation in post-trained models. Replacing the original distillation stage in boomerang distillation, the student first undergoes a *pre-training* phase, followed by a *supervised fine-tuning* phase. The pre-training phase ensures that we recover the general language modeling capability in the initialized student model before restoring the instruction-following and reasoning capabilities. In Section 4.2, we find that the two-phase distillation prevents overfitting and shows better out-of-domain performance.

Pre-training phase. We follow the same boomerang distillation setup over a pre-training corpus $\mathcal{D}_{\text{pre}}$. Given a training sequence $x = (x_1, \ldots, x_L) \in \mathcal{D}_{\text{pre}}$, the loss from Equation 1 is summed over all positions $x_2, ..., x_L$. Training on pre-training data not only mitigates catastrophic forgetting, but has even been shown to improve downstream performance (Kotha and Liang, 2026).

Supervised fine-tuning phase. In this phase, we transition to instruction-following data $\mathcal{D}_{\text{SFT}}$, where each sequence $(x, y) \in \mathcal{D}_{\text{SFT}}$ consists of a prompt $x = (x_1, ..., x_L)$ and a response $y = (y_1, ..., y_{L'})$. We use the same distillation objective, but loss (Equation 1) is computed only over all the response tokens. This stage allows the student to adopt instruction-following capabilities and adapt to downstream generation tasks, while preserving alignment to the teacher.

We dedicate half of training to pre-training and half to supervised fine-tuning (see Appendix E.1). We also use a continuous LR schedule for both phases as we found in initial experiments that two separate schedulers can hurt final performance.

3.2.2 ADAPT (Weight-Delta): Weight-Delta Initialization

Although the two-phase distillation produces strong interpolated model families for generation and reasoning tasks (Figure 2), applying it independently to each post-trained variant still requires a separate distillation run. We propose **weight-delta initialization**, a weight-arithmetic procedure that transfers a base-model distillation run to a post-trained variant without additional training, approximately constructing distilled post-trained student models and amortizing the cost of constructing multiple post-trained model families.

Let $\boldsymbol{\theta}^{\text{base}}_{S,\text{init}}$ denote the student initialized from the base teacher, and let $\boldsymbol{\theta}^{\text{base}}_{S,\text{distill}}$ denote the corresponding student after two-phase distillation. We define the distillation delta as:

$$\Delta^{\text{base}}_{S,\text{distill}} = \boldsymbol{\theta}^{\text{base}}_{S,\text{distill}} - \boldsymbol{\theta}^{\text{base}}_{S,\text{init}}.$$

Given a student $\boldsymbol{\theta}^{\text{PT}}_{S,\text{init}}$ initialized from a post-trained teacher using the same layer-selection rule and parameter shapes as $\boldsymbol{\theta}^{\text{base}}_{S,\text{init}}$, we construct the weight-delta-initialized student $\boldsymbol{\theta}^{\text{PT}}_{S,\Delta}$ as:

$$\boldsymbol{\theta}^{\text{PT}}_{S,\Delta} = \boldsymbol{\theta}^{\text{PT}}_{S,\text{init}} + \Delta^{\text{base}}_{S,\text{distill}}.$$

The resulting model $\boldsymbol{\theta}^{\text{PT}}_{S,\Delta}$ is intended to approximate the directly distilled post-trained student $\boldsymbol{\theta}^{\text{PT}}_{S,\text{distill}}$ in downstream performance, while avoiding the cost of an additional distillation run.

4 Experiments

Here we study the creation of post-trained model families using ADAPT. We show the benefits of ADAPT over strong baselines (Sections 4.2 and

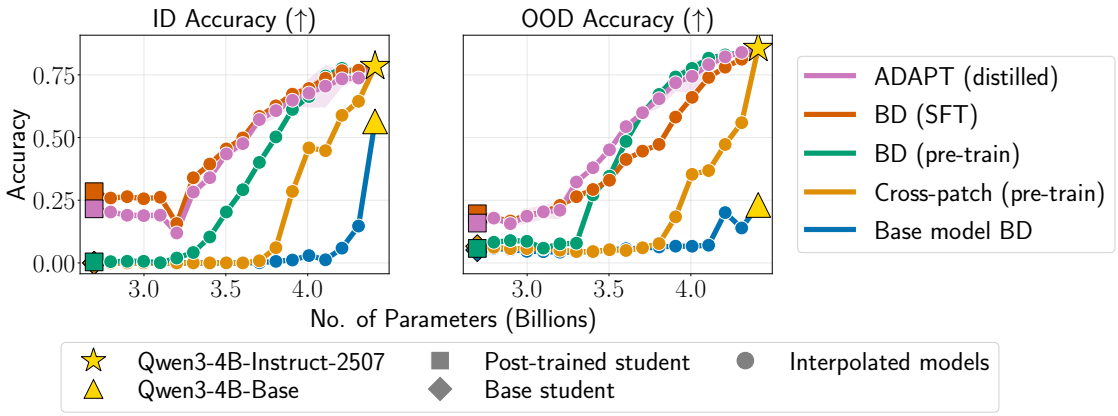

Figure 2: **ADAPT improves generation performance over boomerang distillation.** For generation tasks that are both in-domain (math reasoning) and out-of-domain (instruction-following and general knowledge) relative to our SFT dataset, SFT training is necessary to recover generation performance, especially for smaller models. For OOD tasks, two-phase distillation stabilizes performance compared to SFT-only training. We observe similar results for Qwen3-14B, Olmo-3-7B-Instruct, and Llama-3.1-8B-Instruct in Appendix M.

4.3). Then, we show ADAPT offers a better performance–compute trade-off than any single model (Section 4.4). Finally, we analyze the loss landscape to understand weight-delta initialization (Section 4.5).

4.1 Setup

In our experiments, we primarily use Qwen3-4B-Instruct-2507 (Yang et al., 2025) as the teacher model. We also study Qwen3-4B-Thinking-2507 and Qwen3-4B in Section 4.3. We initialize the student models with 2.7B inference-time parameters by removing every other layer from the teacher (excluding the final layer). We train on a total budget of 1B tokens. For more training details, see Appendices A and B. We also perform the same experiments with Qwen3-14B, Olmo (Olmo et al., 2025), and Llama (Grattafiori et al., 2024) families in Appendix M. We train on the deduplicated Pile (Gao et al., 2020) during the pre-training stage, and the math split of the Llama Nemotron Post-Training Dataset (Bercovich et al., 2025) during the SFT stage. We primarily evaluate our models on a set of five generation benchmarks measuring instruction-following and mathematical reasoning. We also apply our method to coding tasks and report the results in Appendix M.4. For more implementation details, see Appendix C. We include error bars of one standard deviation in the figures.

4.2 Two-Phase Distillation for Reasoning

We begin by comparing ADAPT (distilled) to boomerang distillation (BD), showing that the two-phase distillation pipeline is key to model size interpolation for generation and reasoning tasks, yielding substantially stronger interpolation performance.

Setup. We compare five setups under an equal training budget, all evaluated on three in-domain (ID) math reasoning tasks and two out-of-domain (OOD) tasks testing instruction-following and general understanding. (1) *ADAPT (distilled)*: Our proposed two-phase distillation setup. (2) *BD (pre-train)*: Directly applying BD to the post-trained teacher. (3) *BD (SFT)*: BD on post-trained model but with SFT instead of pre-training data in distillation stage. (4) *Cross-patch (pre-train)*: Patching post-trained teacher directly onto boomerang-distilled base student to study whether post-trained teacher capabilities can be transferred through patching alone. (5) *Base model BD*: BD on base model as a reference.

Results. Figure 2 shows that ADAPT (distilled) achieves the strongest interpolation performance on both ID and OOD tasks by combining pre-training alignment with SFT capability recovery. It also achieves the best teacher–student alignment, as measured via last-layer activation cosine similarity (Appendix D.1). We observe similar trends for Qwen3-14B, Olmo, and Llama in Appendix M and report additional ablation results in Appendix E.

BD (pre-train) remains weak at small and medium model sizes, improving only as model size increases. BD (SFT) does not generalize as well to OOD and classification tasks (Appendix D.2), despite comparable performance to ADAPT (distilled) at small model sizes on ID tasks. Base model BD collapses completely, indicating that

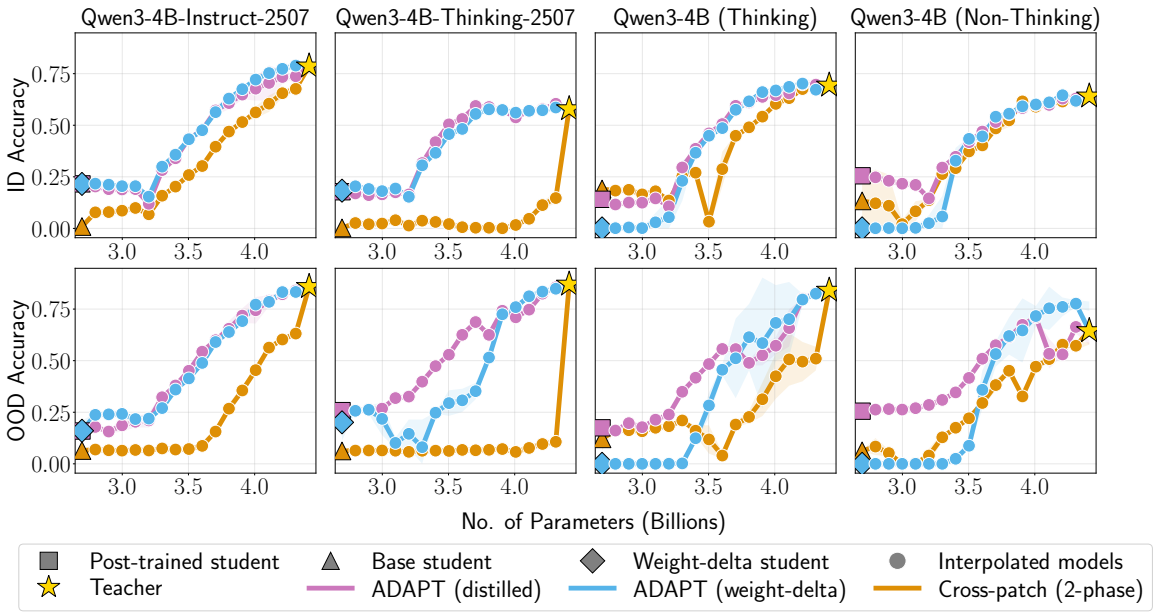

Figure 3: **ADAPT (weight-delta) enables reusable base model distillation across post-trained variants.** ADAPT (weight-delta) matches ADAPT (distilled) on ID tasks and remains competitive on OOD tasks, showing that one base model distillation run can transfer to multiple post-trained variants without additional training. ADAPT (weight-delta) outperforms baseline cross-patch (2-phase) across all variants. Degradation in thinking-style settings is mainly due to malformed <think> formatting, rather than broad loss of generation quality.

base model interpolation does not directly transfer to generation and reasoning tasks.

Interestingly, despite performing significantly worse than other setups, cross-patch (pre-train) recovers nontrivial performance as more post-trained teacher layers are inserted, outperforming base model BD. This suggests that base and post-trained models remain partially compatible, motivating our weight-delta transfer approach in Section 4.3 and smooth weight interpolation experiments in Section 4.5.

4.3 Weight-Delta Initialization Enables Reusable Base Distillation

We demonstrate that ADAPT (weight-delta) can reuse a single base-model distillation run across multiple post-trained variants, recovering much of the interpolation behavior of directly distilled post-trained students without additional distillation.

Setup. We construct the interpolated models for Qwen3-4B-Instruct-2507, Qwen3-4B-Thinking-2507, and Qwen3-4B using ADAPT (weight-delta). Specifically, we first perform two-phase distillation on the Qwen3-4B-Base student to obtain the base distillation delta, then add this delta to students initialized from each post-trained variant. We compare ADAPT (weight-delta) against two setups: (1) *ADAPT (distilled)*, the higher-compute upper bound that runs a separate two-phase distillation for each post-trained student. (2) *Cross-patch (2-phase)*, which patches post-trained teacher layers directly into the two-phase-distilled base student without applying the weight delta. Cross-patch (2-phase) serves as a natural, compute-matched baseline: like ADAPT (weight-delta), it produces a family of interpolated post-trained models from a single base-model distillation run. All methods use the same student architecture, layer-selection rule, patching order, and evaluation protocol.

Results. Figure 3 shows that ADAPT (weight-delta) achieves interpolation performance comparable to ADAPT (distilled) on ID tasks across the evaluated Qwen variants, while requiring no additional distillation on the post-trained teachers. On OOD tasks (IFEval and MMLU-Redux), ADAPT (weight-delta) remains competitive with ADAPT (distilled), with slight degradation concentrated in thinking-style models at smaller and intermediate sizes. Inspecting outputs shows this is largely a parsing artifact rather than a loss of generation quality: weight-delta models are initialized from a base distillation run without thinking-style <think> tags and can produce malformed delimiters, which IFEval penalizes since it parses only the response after </think>. Performance remains strong on MMLU-Redux, which does not

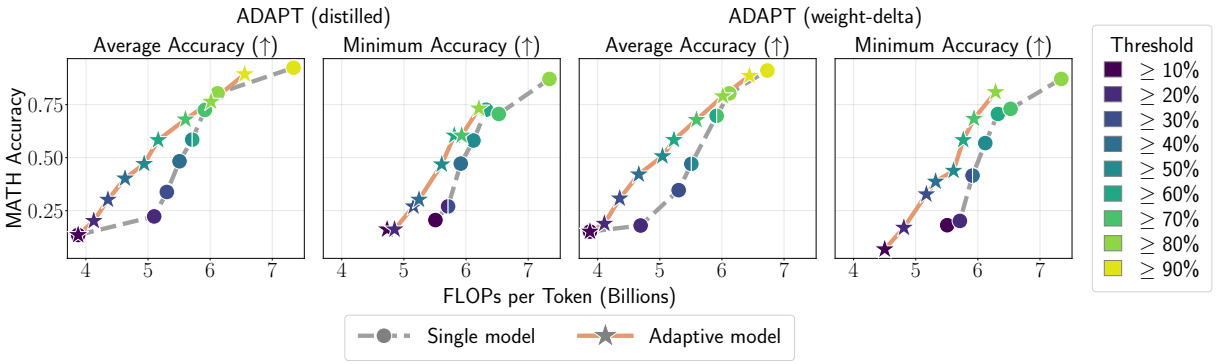

Figure 4: **ADAPT enables adaptive model-size selection based on input difficulty for more compute-efficient inference.** By producing a continuum of interpolated models with smoothly varying performance, ADAPT allows for more fine-grained matching between model capacity and task difficulty. This approach achieves a Pareto-optimal trade-off between average and worst-case accuracy and computational efficiency.

depend on </think> parsing, confirming the issue is specific to strict thinking-tag parsing rather than a broad degradation in capability. ADAPT (weight-delta) also substantially outperforms cross-patch (2-phase) across post-trained variants, suggesting that the base model distillation delta transfers useful instruction-following capability and enables stronger interpolation than patching alone.

Results for Olmo and Llama models (Appendix M) suggest that ADAPT (weight-delta) generally falls between ADAPT (distilled) and cross-patch (2-phase), and is comparable to ADAPT (distilled) in many cases. It also achieves better interpolation performance than cross-patch (2-phase) in all ID settings, making it a cheaper but still strong-performing alternative to separately distilling each post-trained variant. We provide preliminary explanation for when weight-delta initialization works best in Appendix G.

We also report results for single-phase weight-delta initializations in Appendix F.1, where the transferred distillation delta is computed from a base student distilled on the pre-training phase or SFT phase only. This setting still demonstrates some cross-model distillation transfer, but generally underperforms the two-phase weight-delta initialization used in ADAPT. We experiment with continued training from the weight-delta initialized model, $\boldsymbol{\theta}^{\mathrm{PT}}_{S,\Delta}$, but find that it does not improve interpolation performance (Appendix F.2). We include further ablation results on post-trained model deltas and SFT phase loss components in Appendices F.3 and F.4. Finally, we compare both variants of ADAPT to popular layer pruning methods in Appendix H.

4.4 Adaptive Model-Size Selection

In this section, we show that ADAPT achieves a Pareto-optimal trade-off between downstream performance and computational efficiency by dynamically choosing from the suite of interpolated models at inference time based on input difficulty. This improves compute utilization while preserving performance, satisfying a practical requirement for deployment.

Setup. We use the MATH dataset (Hendrycks et al., 2021b), which contains competition-style problems with varying difficulty levels. We estimate each problem's difficulty using a single forward pass with the teacher model, Qwen3-4B-Instruct-2507 (Appendix I.1). For each interpolated model, we calculate its accuracy on each of the difficulty levels on a training split. Given a target accuracy threshold, we route questions of each difficulty level to the smallest model with accuracy above the threshold.

We consider both average accuracy and minimum accuracy. Average accuracy measures a model's example-weighted mean accuracy across all the difficulty levels, allowing it to undershoot the threshold on harder levels. Minimum accuracy requires every difficulty level to individually exceed the threshold, yielding a more conservative policy. Sweeping over the threshold produces routing policies with different accuracy–compute trade-offs, which we compare against fixed single-model baselines.

Results. Figure 4 shows that the adaptive model-size selection approach outperforms the single-model baseline, tracing out a superior Pareto frontier across both average and minimum accuracy metrics for both ADAPT (distilled) and

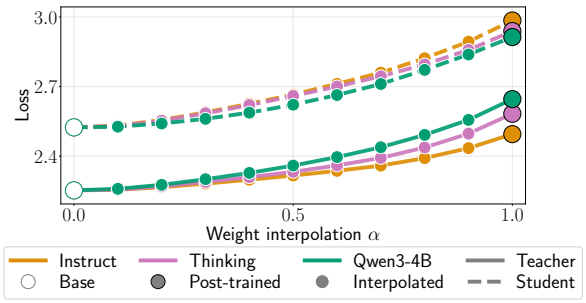

Figure 5: **Smooth weight interpolation is preserved after distillation.** Across post-trained variants, both teacher and distilled student models exhibit smooth interpolation paths between base and post-trained endpoints, suggesting that the relevant weight-space structure is preserved after distillation.

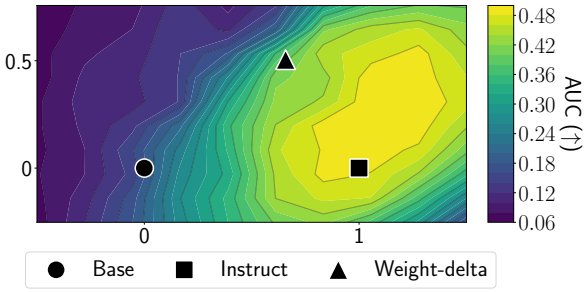

Figure 6: **Weight-delta initialization produces a student in a connected region of high generation performance.** We report AUC for generation accuracy between student and teacher models on the plane intersecting base, instruct, and weight-delta students. The weight-delta student has improved AUC compared to the base student and lies in a high-performance region connected to both the base and instruct students.

ADAPT (weight-delta). This improvement arises from a more fine-grained matching between model capability and question difficulty, enabled by the suite of models with smoothly increasing performance. Therefore, by assigning smaller models to easier questions and reserving larger models for the harder ones, ADAPT offers a superior performance–compute trade-off over using a single model.

We report the results for Olmo and Llama models in Appendix I.2. We also notice that smaller models tend to generate more tokens overall. We provide further discussion of this in Appendix I.3.

4.5 Size-Agnostic Smooth Weight Interpolation

The weight-delta initialization results in Section 4.3 suggest that base model distillation remains compatible with its post-trained variants. To better understand this, we study whether interpolation between base and post-trained models is preserved after distillation. We call this property **size-agnostic smooth weight interpolation**: the low-loss linear path between a base model and its post-trained variant is preserved in the corresponding student models after distillation. For models $\boldsymbol{\theta}^{(a)}$ and $\boldsymbol{\theta}^{(b)}$, we evaluate models $\boldsymbol{\theta}(\alpha) = (1-\alpha)\boldsymbol{\theta}^{(a)} + \alpha\boldsymbol{\theta}^{(b)}$ where $\alpha \in [0, 1]$.

Figure 5 shows that interpolation paths between base and post-trained models do not exhibit sharp loss barriers for either teachers or students. The loss increases toward the post-trained endpoints because they are evaluated on the Pile, which is better matched to base models than post-trained variants. Importantly, students retain similar interpolation behavior after distillation, suggesting that distillation preserves much of the weight-space structure connecting base and post-trained models. We observe similar trends for generation accuracy and SFT loss in Appendix K. We further examine this geometry using a two-dimensional generation-accuracy AUC landscape over the plane spanned by the base, instruct, and weight-delta students. Here, AUC denotes the area under the generation-accuracy curve, measured between the smaller student and the corresponding teacher model. Figure 6 shows that the weight-delta initialized student lies in a high-performing region connected to both the base and distilled instruct students. These results suggest that distillation preserves smooth weight interpolation across model sizes, suggesting a possible connection to linear-mode connectivity (Garipov et al., 2018).

5 Conclusion

We introduce ADAPT, a framework for amortizing distillation across both the size and post-training-variant axes of a model family. ADAPT substantially outperforms boomerang distillation baselines on reasoning and instruction-following benchmarks under matched compute, and the resulting continuum of interpolated models enables adaptive model-size selection at inference time. Crucially, we find that smooth linear weight interpolation between base and post-trained models is preserved across model sizes after distillation, providing a plausible mechanism for the success of weight-arithmetic transfer in our setting. This connection between size interpolation and linear-mode connectivity points to a broader principle—that distillation can be reused across related models, not just sizes—and we view its further investigation as a promising direction.

Limitations

ADAPT is effective across the model families and benchmarks we study, but several aspects warrant further investigation. We highlight open questions about the underlying mechanism, assumptions implicit in our procedure, and the empirical scope of our evaluation.

Interpolation dips at specific sizes. As also observed by Kangaslahti et al. (2026a), some interpolation curves show sharp drops at particular intermediate sizes. A plausible explanation is that the inserted teacher layers are locally misaligned with the surrounding student layers at those configurations, but the precise mechanism remains unclear. Characterizing and predicting these failure modes remain open directions. We also observe that the weight-delta transfer is not uniformly reliable across Olmo and Llama (Figures 35 and 37), especially in some OOD settings and intermediate sizes. This suggests that weight-delta effectiveness depends on model-family compatibility and post-training details, and should be validated beyond the settings studied here.

Model and data scope. Our experiments cover three model families (Qwen, Olmo, Llama) at the 4B-14B scale and use the math and coding splits of the Llama Nemotron Post-Training Dataset for the SFT phase. It remains to be verified whether ADAPT retains its size-amortization benefits at larger model scales and across broader instruction-tuning mixtures (e.g., multilingual).

Weight-delta assumptions. Weight-delta initialization requires access to the base model used to produce each post-trained variant. For some open-weight releases (and most closed-weight releases) the corresponding base checkpoint is unavailable or not exactly identifiable, restricting which model families ADAPT (weight-delta) can target without additional approximation.

Adaptive inference depends on a difficulty signal. The compute–accuracy gains from adaptive model-size selection (Section 4.4) rely on a per-input difficulty estimate. Our MATH experiments use teacher-predicted difficulty, which is cheap but task-specific; we have not characterized how well this approach generalizes to settings where difficulty is not well-defined or where a separate difficulty predictor would be expensive to obtain.

Ethical Considerations

Our experiments use publicly available datasets and evaluation benchmarks, primarily focused on mathematical reasoning, instruction following, and general model evaluation. We also use publicly available model checkpoints as teachers and base models. We do not collect new human-subject data. However, because pre-training and post-training datasets may contain web-derived text, they may inherit biases, harmful content, or privacy concerns from their source corpora. Similarly, the models used in this work may inherit biases or unsafe behaviors from their original training pipelines or the datasets we use. Our work does not attempt to remove these risks, and downstream users should evaluate generated outputs for safety, bias, and reliability before deployment.

Reproducibility Statement

We implement all our experiments using PyTorch (Paszke et al., 2019) and HuggingFace transformers (Wolf et al., 2020) packages. We also experiment with public models available on HuggingFace Hub. We provide our code and models at https://github.com/dcml-lab/ADAPT.

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). Francesco Locatello's contribution to this research was funded in part by the Austrian Science Fund (FWF) 10.55776/COE12.

References

AI-MO. 2025. Aimo validation aime dataset. https://huggingface.co/datasets/AI-MO/aimo-validation-aime. Accessed: 2026-03-29.

Art of Problem Solving. n.d. Aops wiki: Competition ratings. https://artofproblemsolving.com/wiki/index.php/AoPS_Wiki:Competition_ratings. Accessed: 2026-03-23.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. *Preprint*, arXiv:2108.07732.

Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, Ido Shahaf, Oren Tropp, Ehud Karpas, Ran Zilberstein, Jiaqi Zeng, Soumye Singhal, Alexander Bukharin, Yian Zhang, Tugrul Konuk, and 114 others. 2025. Llama-nemotron: Efficient reasoning models. *Preprint*, arXiv:2505.00949.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7432–7439.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code.

Xiaodong Chen, Yuxuan Hu, Jing Zhang, Yanling Wang, Cuiping Li, and Hong Chen. 2025. Streamlining redundant layers to compress large language models. *Preprint*, arXiv:2403.19135.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *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. Association for Computational Linguistics.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *Preprint*, arXiv: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. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

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. 2020. The pile: An 800gb dataset of diverse text for language modeling. *Preprint*, arXiv: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, and 5 others. 2023. A framework for few-shot language model evaluation.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Preprint*, arXiv:1802.10026.

Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, Claire Barale, Robert McHardy, Joshua Harris, Jean Kaddour, Emile van Krieken, and Pasquale Minervini. 2025. Are we done with mmlu? *Preprint*, arXiv:2406.04127.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. *Preprint*, arXiv:2103.03874.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *Preprint*, arXiv:1503.02531.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, and 3 others. 2022. Training compute-optimal large language models. *Preprint*, arXiv:2203.15556.

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

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing models with task arithmetic.

In *The Eleventh International Conference on Learning Representations*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2025. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*.

Sara Kangaslahti, Nihal V. Nayak, Jonathan Geuter, Marco Fumero, Francesco Locatello, and David Alvarez-Melis. 2026a. Boomerang distillation enables zero-shot model size interpolation. In *The Fourteenth International Conference on Learning Representations*.

Sara Kangaslahti, Jonathan Geuter, Nihal V. Nayak, Marco Fumero, Francesco Locatello, and David Alvarez-Melis. 2026b. Understanding layer patching in model size interpolation. *Preprint*, arXiv:2607.08170.

Yeganeh Kordi, Nihal V. Nayak, Max Zuo, Ilana Nguyen, and Stephen H. Bach. 2025. Revisiting generalization across difficulty levels: It's not so easy. *Preprint*, arXiv:2511.21692.

Suhas Kotha and Percy Liang. 2026. Replaying pre-training data improves fine-tuning. *Preprint*, arXiv:2603.04964.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham M. Kakade, Prateek Jain, and Ali Farhadi. 2022. Matryoshka representation learning. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Sijia Liu, Yuanshun Yao, Jinghan Jia, Stephen Casper, Nathalie Baracaldo, Peter Hase, Yuguang Yao, Chris Yuhao Liu, Xiaojun Xu, Hang Li, and 1 others. 2025. Rethinking machine unlearning for large language models. *Nature Machine Intelligence*, 7(2):181–194.

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2025. ShortGPT: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20192–20204, Vienna, Austria. Association for Computational Linguistics.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *Preprint*, arXiv:2501.19393.

Avanika Narayan, Dan Biderman, Sabri Eyuboglu, Avner May, Scott Linderman, James Zou, and Christopher Re. 2025. Cost-efficient collaboration between on-device and cloud language models. In *Forty-second International Conference on Machine Learning*.

Nihal V. Nayak, Paula Rodriguez-Diaz, Neha Hulkund, Sara Beery, and David Alvarez-Melis. 2026. A critical look at targeted instruction selection: Disentangling what matters (and what doesn't). In *Forty-third International Conference on Machine Learning*.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, and 49 others. 2025. Olmo 3. *Preprint*, arXiv:2512.13961.

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, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. Pytorch: An imperative style, high-performance deep learning library. *Preprint*, arXiv:1912.01703.

Qwen Team. 2026. Qwen3.5: Towards native multimodal agents.

Adir Rahamim, Asaf Yehudai, Boaz Carmeli, Leshem Choshen, Yosi Mass, and Yonatan Belinkov. 2026. Will it merge? on the causes of model mergeability. *Preprint*, arXiv:2601.06672.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *Preprint*, arXiv:1910.01108.

Safal Shrestha, Anubhav Shrestha, Aadim Nepal, Minwu Kim, and Keith Ross. 2026. On the limits of layer pruning for generative reasoning in llms. *Preprint*, arXiv:2602.01997.

Ali Taghibakhshi, Ruisi Cai, Saurav Muralidharan, Sharath Turuvekere Sreenivas, Ameya Sunil Mahabaleshwarkar, Marcin Chochowski, Akhiad Bercovich, Ran Zilberstein, Ran El-Yaniv, Yonatan Geifman, Daniel Korzekwa, Yoshi Suhara, Oluwatobi Olabiyi, Ashwath Aithal, Nima Tajbakhsh, and Pavlo Molchanov. 2026. Star elastic: Many-in-one reasoning LLMs with efficient budget control. In *Forty-third International Conference on Machine Learning*.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Abdul Waheed, Chancharik Mitra, Laurie Z. Wang, Deva Ramanan, and Bhiksha Raj. 2025. Less is more tokens: Efficient math reasoning via difficulty-aware chain-of-thought distillation. *Preprint*, arXiv:2509.05226.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Huggingface's transformers: State-of-the-art natural language processing. *Preprint*, arXiv:1910.03771.

Prateek Yadav, Tu Vu, Jonathan Lai, Alexandra Chronopoulou, Manaal Faruqui, Mohit Bansal, and Tsendsuren Munkhdalai. 2025. What matters for model merging at scale? *Transactions on Machine Learning Research*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. 2025. Limo: Less is more for reasoning. *Preprint*, arXiv:2502.03387.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *Preprint*, arXiv:2311.07911.

A Training Implementation Details

We use a mixture of non-thinking and thinking models in this paper. Non-thinking models include Qwen3-4B-Instruct-2507, Qwen3-4B, Olmo-3-7B-Instruct, Llama-3.1-8B-Instruct, and Qwen3-14B. Thinking models include Qwen3-4B-Thinking-2507, Qwen3-4B, Olmo-3-7B-Think, and Qwen3-14B. Note that Qwen3-4B and Qwen3-14B can toggle between thinking and non-thinking modes.

For model training, we use a total token budget of 1B for the Qwen 4B-sized models. For the larger models, we scale up the total budget proportionally, to 2B and 4B tokens respectively (Hoffmann et al., 2022). Across different configurations of the same model, we choose hyperparameters such that the number of tokens per update remains approximately constant between the pre-training phase and SFT phase. The hyperparameters reported in Table 3 correspond to using the deduplicated Pile (Gao et al., 2020) for pre-training and the Llama Nemotron Post-Training Dataset (Bercovich et al., 2025) for the SFT phase. For Nemotron, we primarily use the math split for all models (see Appendix M.4 for coding tasks). For non-thinking models, we use examples with non-thinking ground truths (i.e., responses not enclosed in `<think>` and `</think>`), while for thinking models we use examples with thinking ground truths (version 1.1).

Each 4B model training run takes approximately 12 hours on 4 NVIDIA H100 GPUs. The Olmo and Llama models are trained using 4 NVIDIA H200 GPUs, each taking approximately 24 hours. Training Qwen3-14B takes approximately 48 hours on 16 H200 GPUs. We will release the distilled models under an Apache 2.0 license.

B Hyperparameters

We choose the majority of the hyperparameters (Table 1) following Kangaslahti et al. (2026a). We determine the KL and cosine loss weights such that cross entropy, KL, and cosine loss are approximately equal in magnitude early in training. M denotes the number of student layers. When doing two-phase distillation, we use a single LR scheduler across the two phases.

C Evaluation Implementation Details

For generation, we use a custom evaluation pipeline for more fine-grained control over chat template and system prompt settings. Table 4 shows the system prompts used for each benchmark. We report the average accuracy on 5 benchmarks: AIME (AIMO, 2025), GSM8K (Cobbe et al., 2021), IFEval (Zhou et al., 2023), MATH500 (Hendrycks et al., 2021b), MMLU-Redux (Gema et al., 2025). We consider AIME, GSM8K, and MATH500 to be in-domain (ID) math reasoning tasks and IFEval and MMLU-Redux to be out-of-domain (OOD) tasks. We use a separate set of ID benchmarks described in Appendix M.4 for coding tasks. For Section 4.4, we use the MATH dataset (Hendrycks et al., 2021b), which contains ground truth question difficulty labels. We elaborate on this further in Appendix I.1.

Hyperparameters	Values
Learning rate	3e-4
LR scheduler	cosine
Warmup ratio	0.01
Optimizer	AdamW
Adam betas	(0.9, 0.95)
Adam epsilon	1e-8
Weight decay	0.1
Max gradient norm	1.0
Mixed precision	bf16
KL weight λ_{KL}	0.1
Cosine weight λ_{cos}	10.0 / (M+1)

Table 1: **Hyperparameters for distillation training.**

For classification tasks, we use `lm-evaluation-harness` (Gao et al., 2023) and report the average accuracy on 10 benchmarks: ARC-easy and ARC-challenge (Clark et al., 2018), BoolQ (Clark et al., 2019), HellaSwag (Zellers et al., 2019), MMLU (Hendrycks et al., 2021a), OpenBookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), RACE (Lai et al., 2017), RTE (Wang et al., 2019), and WinoGrande (Sakaguchi et al., 2020).

Table 2 shows the sampling parameters used for non-thinking and thinking models, which follow the recommended values by Yang et al. (2025).

Models	Non-thinking	Thinking
Max output len.	16384	32768
Temperature	0.7	0.6
Top p	0.8	0.95
Top k	20	20
Presence penalty	0.5	0.5
Min. tokens	32	32

Table 2: **Sampling parameters for non-thinking and thinking models.**

Model Type	Setup	Pre-training Phase			SFT Phase		
		Steps	Effective BS	Seq. len.	Steps	Effective BS	Seq. len.
4B (Non-think)	Pile only	480	2048	1024	–	–	–
	Nemotron only	–	–	–	585	4096	1024
	Pile+Nemotron	240	2048	1024	293	4096	1024
4B (Think)	Pile only	500	512	4096	–	–	–
	Nemotron only	–	–	–	400	1024	4096
	Pile+Nemotron	250	512	4096	200	1024	4096
7B/8B (Non-think)	Pile only	960	2048	1024	–	–	–
	Nemotron only	–	–	–	1170	4096	1024
	Pile+Nemotron	480	2048	1024	585	4096	1024
7B/8B (Think)	Pile only	1000	512	4096	–	–	–
	Nemotron only	–	–	–	800	1024	4096
	Pile+Nemotron	500	512	4096	400	1024	4096
14B	Pile only	875	1024	4096	–	–	–
	Nemotron only	–	–	–	1400	1024	4096
	Pile+Nemotron	438	1024	4096	700	1024	4096

Table 3: **Training hyperparameters across model families.** We report the number of steps, effective batch size, and sequence length for the pre-training and SFT phases across all distillation setups.

Benchmarks	System Prompts
AIME, GSM8K, MATH500, MATH	Please reason step by step, and put your final answer within `\boxed{}`.
IFEval	None
MMLU-Redux	Please reason step by step, and select the answer from the given choices A, B, C, or D. Respond only with the letter of the correct answer. Put the index within `\boxed{}`.

Table 4: **System prompts for generation tasks.**

D Additional Evaluation Results for ADAPT (Distilled)

We provide additional results for ADAPT (distilled). We first study final-layer cosine similarity to the teacher in Appendix D.1, then show results on classification tasks in Appendix D.2.

D.1 Cosine Similarity Analysis

In this section, we report the final-layer activation cosine similarity between each interpolated model and the corresponding teacher for ADAPT (distilled), BD (pre-train), BD (SFT), and cross-patch (pre-train). We compute activations by averaging over all downstream tasks.

As shown in Figure 7, ADAPT (distilled) achieves the highest cosine similarity across nearly the entire interpolation curve. BD (SFT) also maintains strong alignment, but is consistently below ADAPT (distilled). Cross-patch (pre-train) has the weakest alignment, especially at smaller model sizes, indicating that patching alone does not sufficiently align the student with the post-trained teacher. These results provide additional motivation for the two-phase distillation approach.

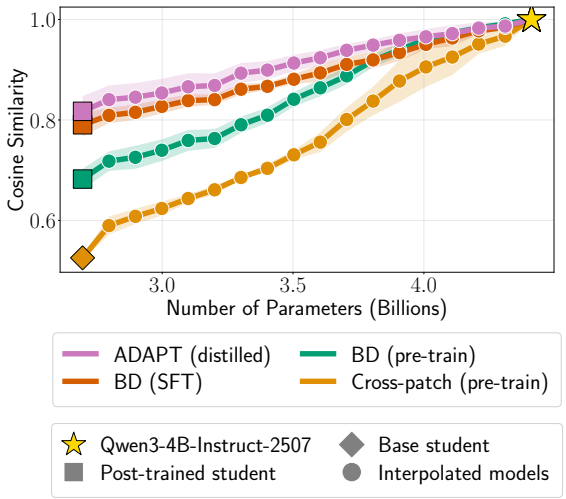

Figure 7: **Final layer activation cosine similarity between student and teacher models.** The two-phase distillation setup yields the best student-teacher alignment, further motivating our approach.

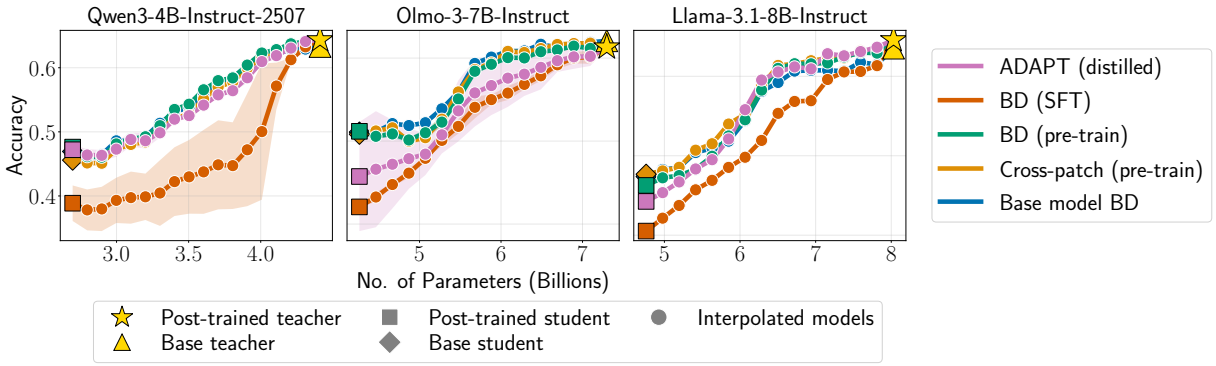

Figure 8: **Classification accuracy is relatively insensitive to the distillation strategy.** Across Qwen, Olmo, and Llama, most setups exhibit similar interpolation behavior on classification benchmarks. The main exception is BD (SFT), which underperforms across much of the interpolation curve, suggesting that task-specific distillation alone can degrade broader alignment with the teacher.

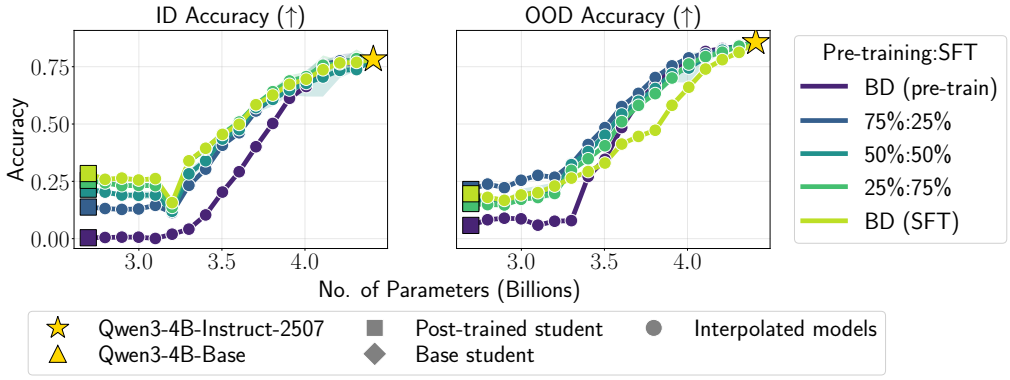

Figure 9: **Distilling with equal pre-training and SFT phase offers an appropriate tradeoff between ID and OOD performance.** Increasing the SFT proportion improves ID accuracy but degrades OOD accuracy, while increasing the pre-training proportion has the opposite effect; the 50:50 split balances the two. The single-phase endpoints are consistently dominated by the two-phase settings.

D.2 Classification Performance

Figure 8 shows that classification benchmarks are relatively insensitive to the distillation strategy. Most variants achieve similar interpolation performance, indicating that classification tasks do not fully expose the alignment challenges that arise in post-trained generation. The main exception is the BD (SFT) setup, which underperforms across much of the interpolation curve for all three model families, suggesting that task-specific distillation alone can degrade broader alignment with the teacher.

E Additional Ablations for ADAPT (Distilled)

E.1 Pre-training vs. SFT Phase Ratios

Here we study how sensitive ADAPT (distilled) is to varying training budget between pre-training and SFT phase. All settings use the same total budget.

In Figure 9, we observe a consistent trade-off across the two-phase settings: increasing the SFT proportion improves ID accuracy but degrades OOD accuracy, while increasing the pre-training proportion has the opposite effect. The 50:50 split sits between these extremes, achieving strong ID accuracy without sacrificing OOD performance. We note that the three two-phase ratios perform comparably at larger model sizes, indicating that ADAPT is not overly sensitive to the exact split.

Finally, both single-phase endpoints are dominated by the two-phase settings, consistent with our finding in Section 4.2.

E.2 Distillation with Cosine Loss Only

In Figure 10, we show that distilling with cosine loss alone can yield performance comparable to the full objective combining cross-entropy, KL, and cosine losses. This suggests that intermediate representation alignment may be a key driver of interpolation behavior, and points to an orthogonal direction for improving distillation efficiency by simplifying the training objective.

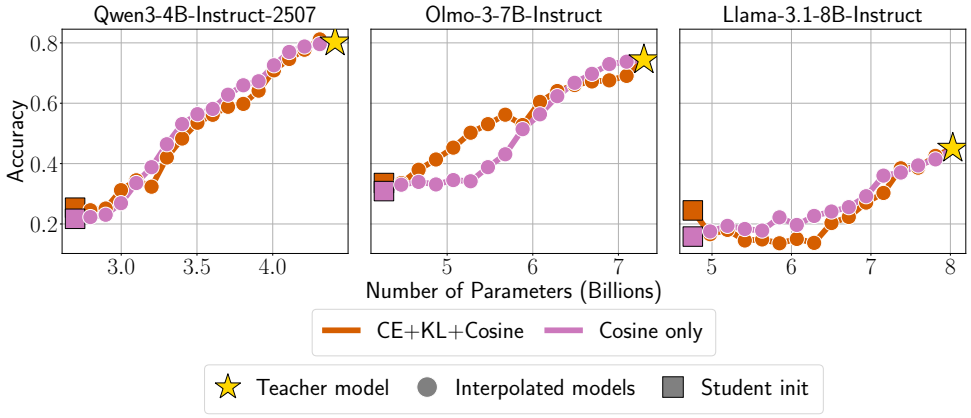

Figure 10: **Distilling with cosine loss only yields interpolation performance comparable to the full objective.** We compare training with cosine alignment loss only to training with all of the loss terms in Equation 1 and find that cosine only training has similar performance, indicating that the efficiency of interpolation training can potentially be improved by using only alignment loss.

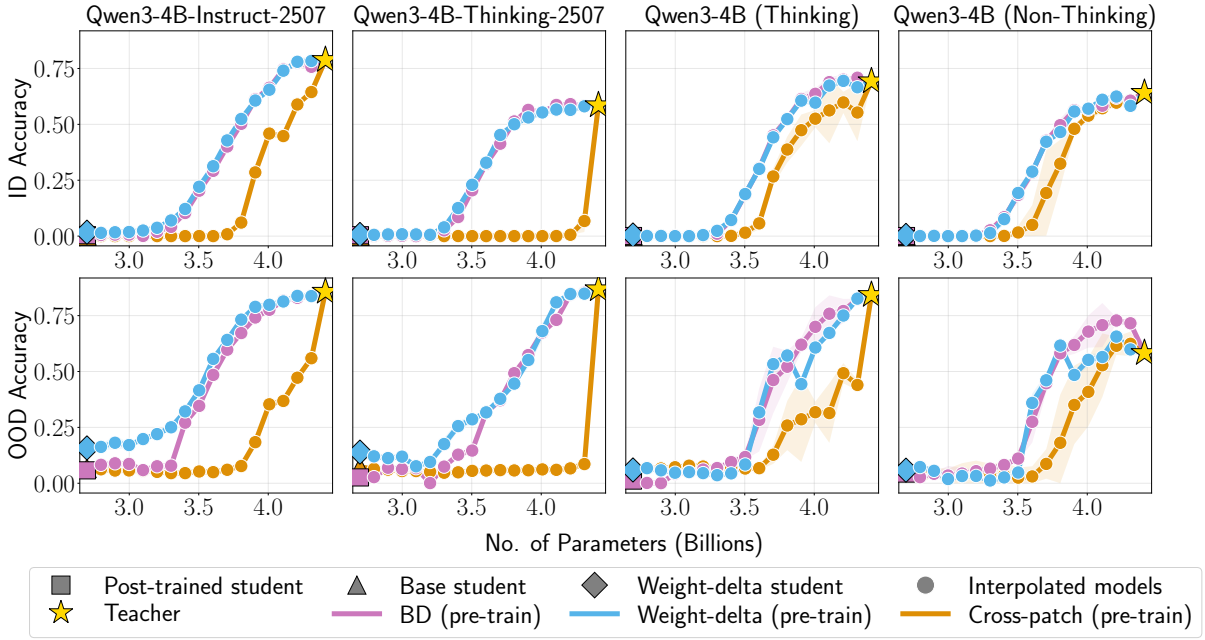

Figure 11: **Pre-training weight-delta initialization for post-trained Qwen models.**

F Additional Ablations for ADAPT (Weight-Delta)

F.1 Weight-Delta Transfer from Pre-training-only and SFT-only Deltas

We evaluate whether weight-delta initialization works when we calculate the weight delta from base model distilled with pre-training or SFT phase only. This setting tests whether the transferable component of distillation can be obtained without the two-phase distillation setup.

Setup. We test whether weight-delta transfer works with single-phase deltas, i.e., deltas computed from a base student distilled on only one type of data (pre-training or SFT). We then add the respective deltas to initialized post-trained students across model families. For comparison, we evaluate these single-phase weight-delta models against two references: the directly distilled BD (pre-train) and BD (SFT) students (which use no delta transfer), and the corresponding cross-patch baselines. We report the results for Qwen (Figures 11 and 12), Olmo (Figure 13), and Llama (Figure 14) families.

Results. Across model families, pre-training weight-delta initialization performs comparably to, and in some cases better than, BD (pre-train), indicating that the alignment learned during base-model pre-training distillation transfers across post-trained variants. The SFT-only delta transfers less smoothly. Weight-delta initialization tracks

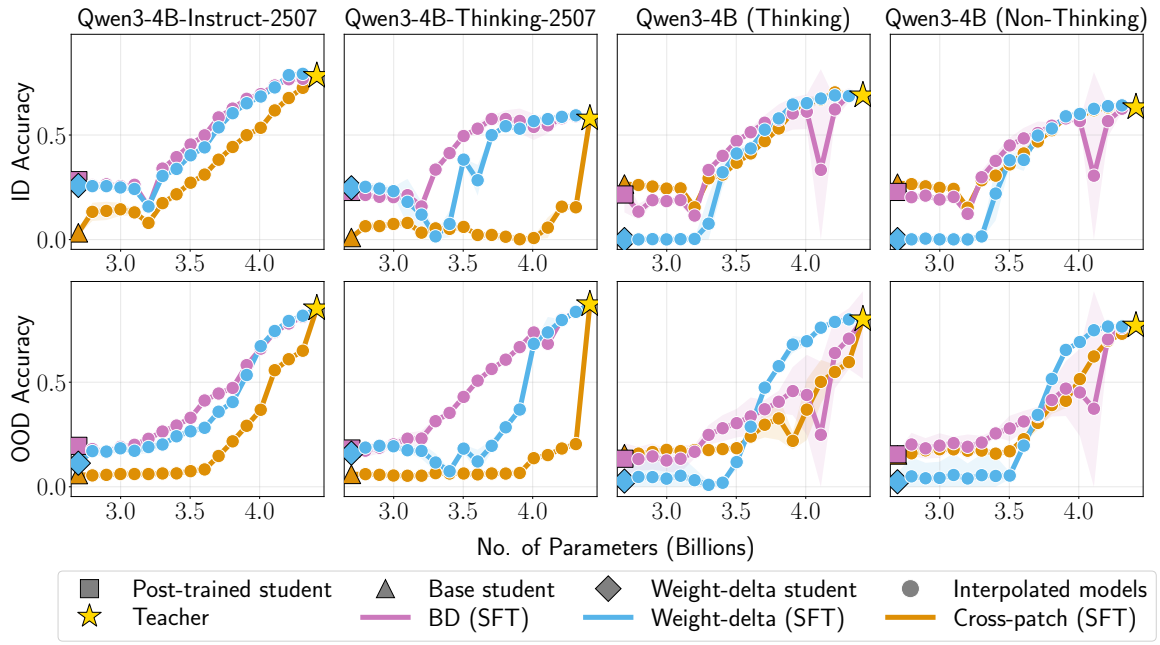

Figure 12: **SFT weight-delta initialization for post-trained Qwen models.**

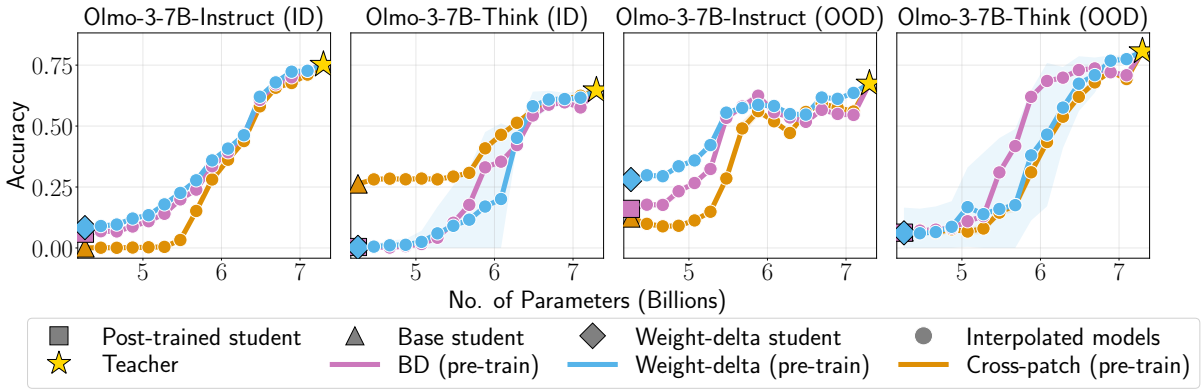

Figure 13: **Pre-training weight-delta initialization performance for post-trained Olmo models.**

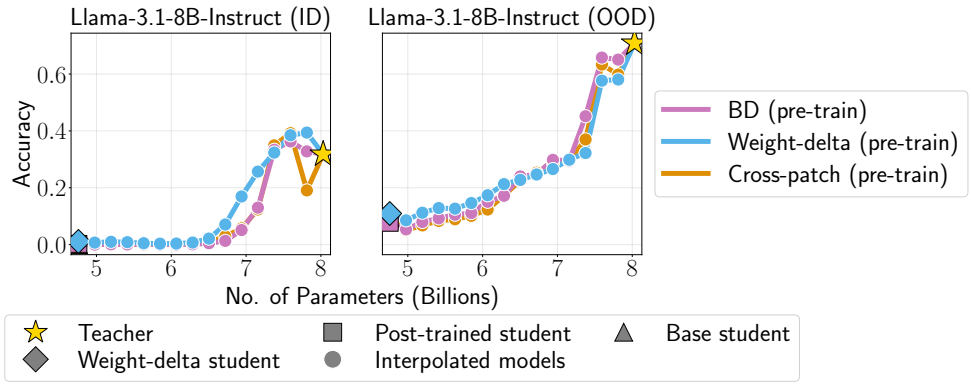

Figure 14: **Pre-training weight-delta initialization performance for post-trained Llama models.**

BD (SFT) closely for the instruct variant, but meaningfully underperforms on the other three variants. The weight-delta setup even performs worse than the cross-patch (SFT) baseline for Qwen3-4B, indicating that SFT-only distillation on the base model is not enough to facilitate effective weight-delta transfer across post-trained models.

Furthermore, single-phase weight-delta initialization setups remain weaker than the two-phase methods. Both ADAPT (weight-delta) and ADAPT (distilled) achieve stronger interpolation performance, reflecting that both phases are necessary to fully recover downstream generation and reasoning capabilities.

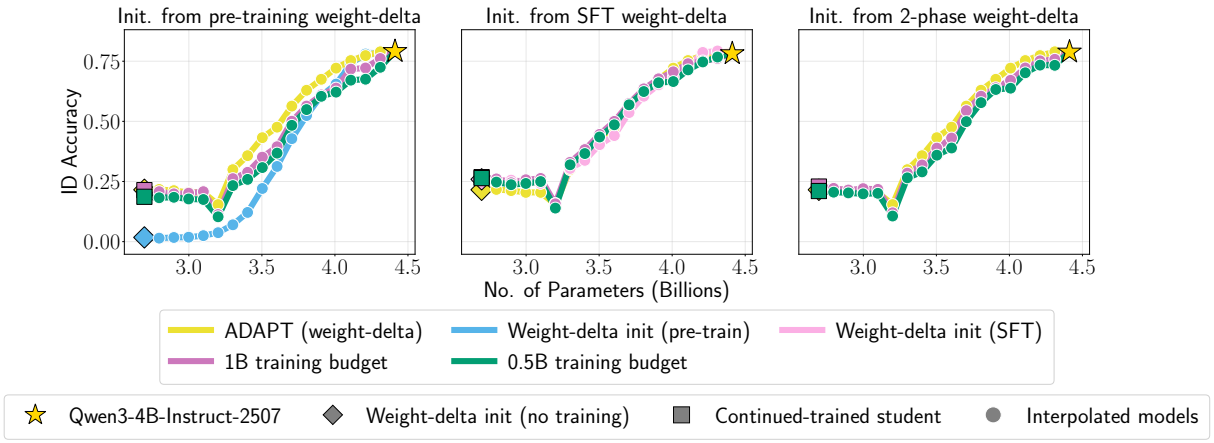

Figure 15: **Continued training from weight-delta initialization does not improve interpolation performance.** We initialize the instruct student using a pre-training, SFT, or two-phase weight delta, computed from base distillation on pre-training data, SFT data, or both respectively. We then continue training for 0.5B or 1B tokens. Continued training does not meaningfully improve over weight-delta initialization.

F.2 Continued Training from Weight-Delta Initialization

We investigate whether weight-delta initialization can serve as a stronger initialization for additional distillation. Specifically, we ask whether continued training from a weight-delta-initialized student can outperform ADAPT (weight-delta).

Setup. We initialize the student by adding one of three weight deltas to it, each obtained by distilling the base model on different data: pre-training data only, SFT data only, or both phases (two-phase). Starting from each initialization, we continue training for either 0.5B or 1B tokens using the same distillation setup described in Appendix B. We also sweep the learning rates and find that 3e-4, the value used in Appendix B, performs the best.

Results. Figure 15 shows that continued training from weight-delta initialization does not meaningfully improve interpolation performance beyond ADAPT (weight-delta). Training for 0.5B or 1B tokens converges to similar interpolation curves, regardless of whether the initialization comes from pre-training (left panel), SFT (middle panel), or two-phase (right panel) weight delta. This suggests that continued distillation largely washes out the difference between the different initializations rather than improving the final interpolated family.

Overall, these results indicate that the main benefit of weight-delta initialization comes from its zero-training transferability, not from providing a better starting point for further training.

F.3 Weight-Delta Initialization with Post-trained Model Deltas

In this section, we test whether distillation done directly on post-trained students (instead of the base student) can be transferred to other post-trained variants via weight-delta initialization.

Setup. We obtain the post-trained model weight deltas by directly performing two-phase distillation on Qwen3-4B-Instruct-2507 and Qwen3-4B-Thinking-2507, and subtracting their corresponding initialized (pre-distillation) students. We then initialize the other post-trained models using the weight deltas from the instruct and thinking models separately following Section 3.2.2.

Results. Figures 16 and 17 show that distillation effects can be transferred onto other post-trained variants using post-trained weight deltas. ADAPT (weight-delta) using instruct and thinking model weight deltas both substantially outperform the cross-patch (2-phase) baseline. Compared against the higher-compute upper bound ADAPT (distilled), the instruct model delta performs competitively on ID tasks, while suffering slight degradation on OOD tasks similar to the base model delta. The thinking model delta matches the performance of ADAPT (distilled) on both ID and OOD tasks. Overall, these results indicate that the transferable distillation delta is not specific to the base model, deltas computed from post-trained students transfer to other variants equally effectively.

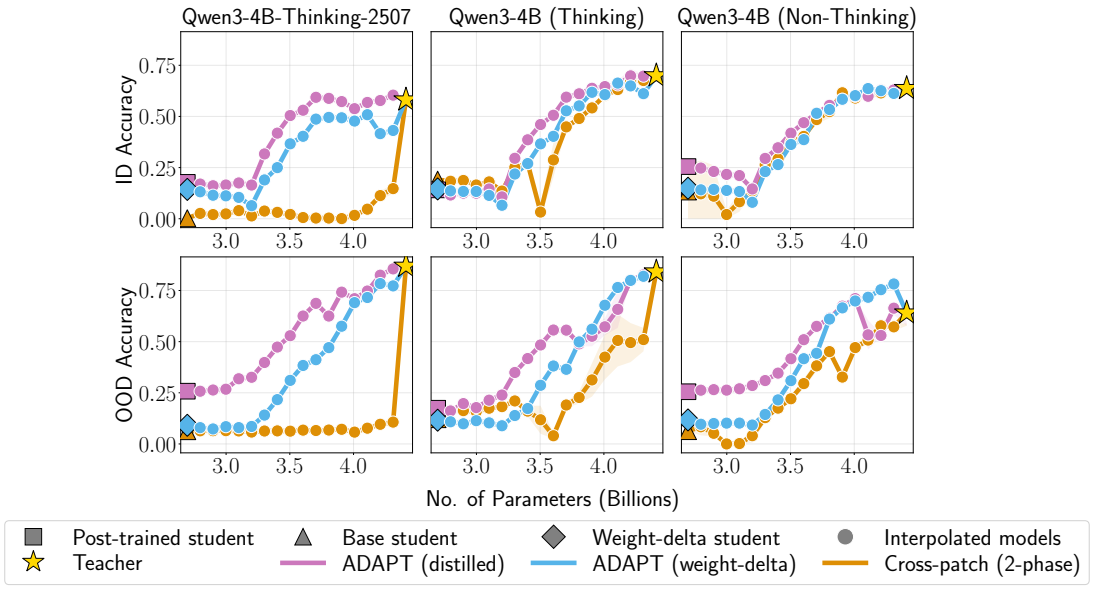

Figure 16: **Results for ADAPT (weight-delta) using delta from Qwen3-4B-Instruct-2507.**

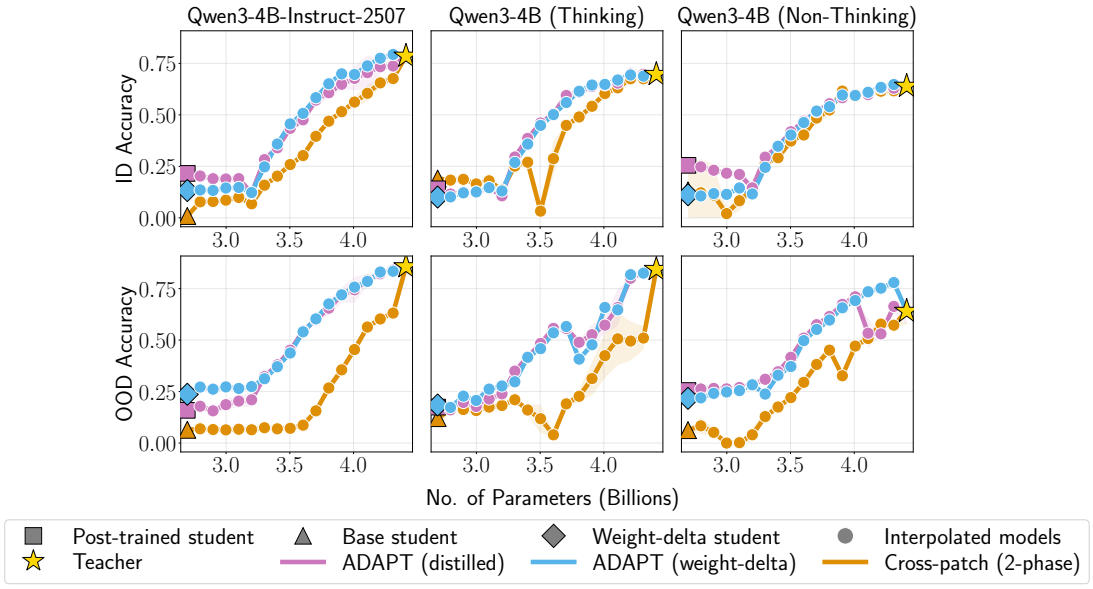

Figure 17: **Results for ADAPT (weight-delta) using delta from Qwen3-4B-Thinking-2507.**

F.4 Ablating SFT Phase Loss Components

While the combination of cross entropy, KL, and cosine loss terms is optimal for base model distillation under the boomerang distillation setup (Kangaslahti et al., 2026a), aligning the student too closely with the less capable base teacher on reasoning tasks during the SFT phase may hinder its learning. In this section, we explore whether relaxing the base student's alignment with the teacher can improve the weight-delta transfer performance.

Setup. We keep the pre-training phase of base model distillation unchanged and ablate the loss terms used in the SFT phase, always retaining cross-entropy while removing the KL term, the cosine term, or both. For each ablated objective, we compute the resulting base model weight delta and use it to initialize the four Qwen post-trained variants following Section 3.2.2. We compare each ablation against the higher-compute upper bound ADAPT (distilled) and the cross-patch (2-phase) baseline.

Results. Removing the KL term leaves interpolation behavior largely unchanged (Figure 18), indicating that it contributes little during the SFT phase. Removing the cosine term has a substantially larger effect (Figure 19): ID performance improves across all four variants, matching ADAPT (distilled),

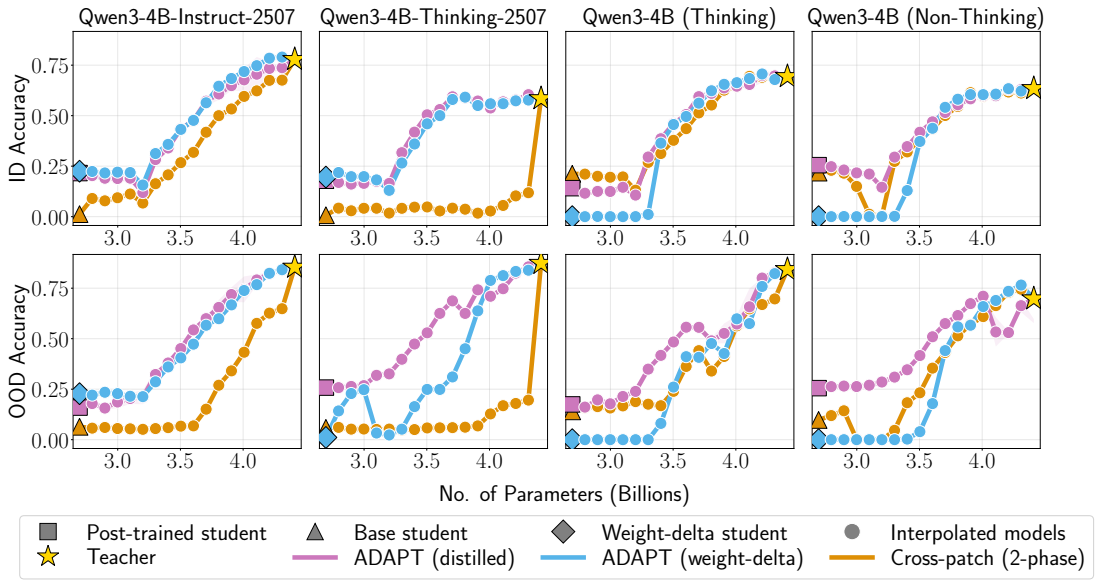

Figure 18: **Results for ablating KL loss term in SFT phase.**

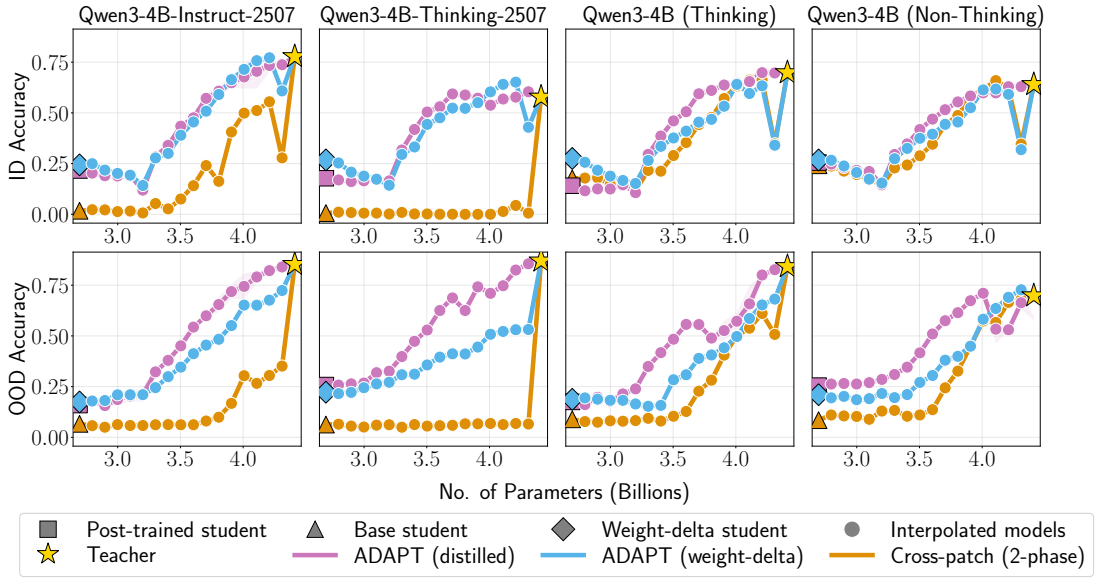

Figure 19: **Results for ablating cosine loss term in SFT phase.**

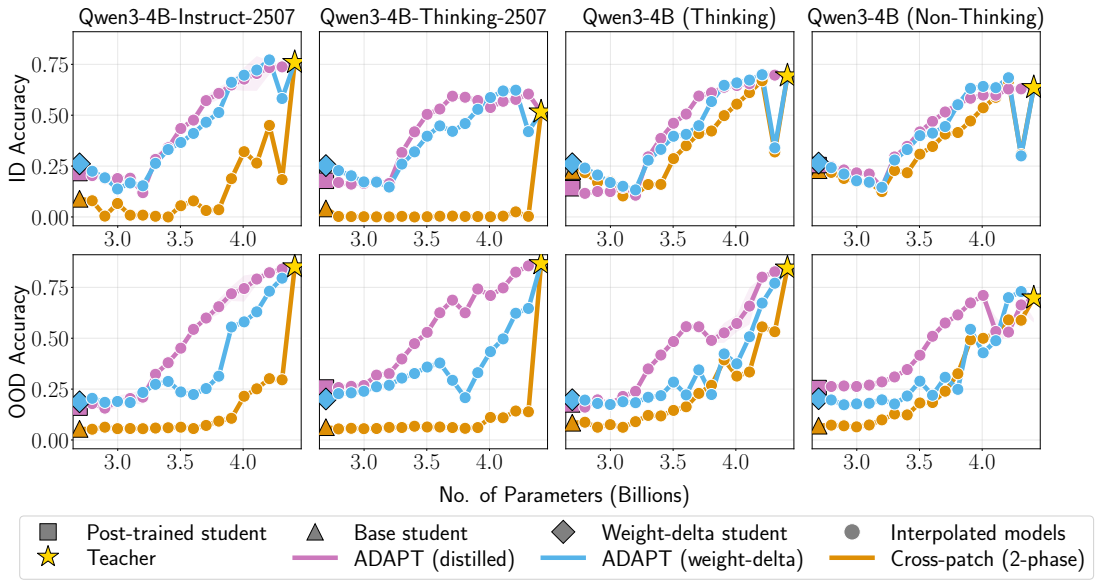

Figure 20: **Results for ablating both KL and cosine loss terms in SFT phase (cross-entropy loss only).**

while OOD performance degrades overall. This split is size-dependent — the ID gains come almost entirely from smaller interpolated models, where the full objective left the weight-delta students near total collapse until roughly 3.3B parameters, whereas OOD performance falls off at intermediate and larger sizes. Removing both terms behaves similarly but with even weaker OOD performance (Figure 20). Therefore, the cosine term in SFT phase trades ID performance for OOD generalization, and the choice of whether to retain it depends on the specific use cases.

G Weight-Delta Alignment Experiments

Results in the preceding appendices and Appendix M show that ADAPT (weight-delta) is more successful for some models and setups than others, matching the ADAPT (distilled) performance at its best. However, understanding when exactly weight-delta transfer succeeds remains an important open question. Notably, this is a recognized and challenging problem even in the closely related setting of model merging and task vectors, where identifying the factors that govern whether weight-arithmetic combination succeeds is an active area of research (Rahamim et al., 2026). Here, we present preliminary results that yield some useful signals.

The task-vector view (Ilharco et al., 2023) treats a capability acquired through finetuning as a shift in weight space—in the language of this paper, a weight delta. A natural predictor of transfer success is therefore the alignment between the base distillation delta ($\boldsymbol{\theta}^{\text{base}}_{S,\text{distill}} - \boldsymbol{\theta}^{\text{base}}_{S,\text{init}}$) and each variant's own distillation delta ($\boldsymbol{\theta}^{\text{PT}}_{S,\text{distill}} - \boldsymbol{\theta}^{\text{PT}}_{S,\text{init}}$). Table 5 reports the cosine similarity and norm ratio between these deltas.

Alignment tracks transfer reliability. Instruct and hybrid variants align most closely (cosine of 0.73-0.75 for Qwen and Olmo), thinking variants less so (0.57-0.64), and Llama least of all (0.30), mirroring the order in which we observe transfer to be reliable, partially reliable, and unreliable. Norm ratios stay near 1, so the deltas differ in direction rather than magnitude. Although this is a correlational analysis, it suggests directional alignment as a candidate diagnostic for when weight-delta transfer will hold. We leave further investigation to future work.

Variant	Cosine	Norm ratio
Qwen3-4B-Instruct-2507	0.73	1.00
Qwen3-4B-Thinking-2507	0.57	0.92
Qwen3-4B	0.75	1.00
Qwen3-14B	0.75	1.00
Olmo-3-7B-Instruct	0.73	1.02
Olmo-3-7B-Think	0.64	0.94
Llama-3.1-8B-Instruct	0.30	0.95

Table 5: Cosine similarity and norm ratio between base and post-trained deltas for each variant.

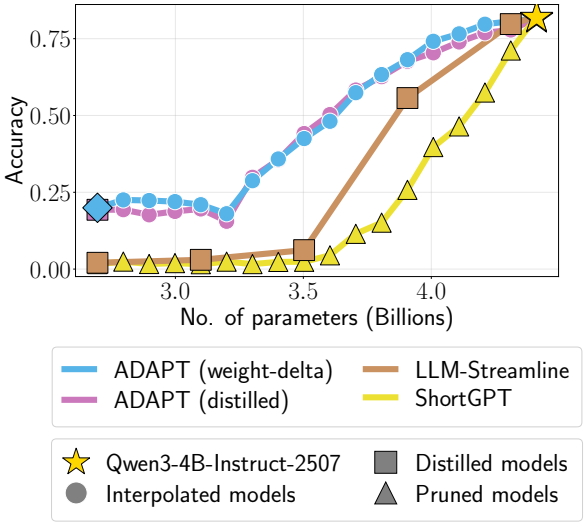

Figure 21: **ADAPT significantly outperforms layer pruning baselines on generation tasks.** We compare ADAPT (distilled) and ADAPT (weight-delta) to two popular layer pruning methods for generation tasks, ShortGPT (Men et al., 2025) and LLM-Streamline (Chen et al., 2025), and show that our method performs substantially better across all sizes.

H Comparison to Layer Pruning Methods

We compare ADAPT against layer pruning methods on generation benchmarks to show that it produces substantially stronger intermediate models.

Setup. We consider two popular layer pruning methods: ShortGPT (Men et al., 2025) and LLM-Streamline (Chen et al., 2025). ShortGPT prunes the least influential layers, as determined by the block influence score (see Appendix H.1). LLM-Streamline identifies the block of layers with the lowest block influence score and replaces it with a lightweight network (see Appendix H.2).

Results. Figure 21 shows that across all model sizes, ADAPT (distilled) and ADAPT (weight-delta) models consistently outperform ShortGPT and LLM-Streamline, which struggle to recover

meaningful generation capability, especially at smaller sizes.

H.1 ShortGPT

For ShortGPT (Men et al., 2025), we first calculate the Block Influence score (BI), which is the cosine distance between the input and output activations. A higher BI score indicates higher importance of a specific layer. The BI score of the ℓ^{th} layer can be calculated as follows:

$$BI_\ell = 1 - \mathbb{E}_{X,t}\left[\frac{\boldsymbol{x}_t^{(\ell)} \cdot \boldsymbol{x}_t^{(\ell+1)}}{\|\boldsymbol{x}_t^{(\ell)}\|\|\boldsymbol{x}_t^{(\ell+1)}\|}\right]$$

where $\boldsymbol{x}_t^{(\ell)}$ is the t^{th} row of hidden states of the ℓ^{th} layer. When pruning, we sequentially remove layers with the lowest BI score. We compute BI using a held-out set of 128 calibration samples from Nemotron.

H.2 LLM-Streamline

For LLM-Streamline (Chen et al., 2025), we identify the block of n layers $(\ell^*, \ell^* + n)$ with the lowest BI score (using the same calibration set from Nemotron as ShortGPT) and replace it with a single lightweight network. We then train the replacement network to imitate the pruned block using mean squared error (MSE):

$$\min_h \mathbb{E}_{(\boldsymbol{x}^{(\ell^*)}, \boldsymbol{x}^{(\ell^*+n)})\in\mathcal{D}} \text{MSE}\left(h(\boldsymbol{x}_i^{(\ell^*)}), \boldsymbol{x}_i^{(\ell^*+n)}\right)$$

where h denotes the lightweight network, $\mathcal{D}$ denotes the recorded hidden states of samples, and $\boldsymbol{x}^{(i)}$ is the hidden state of the i^{th} layer.

Following Chen et al. (2025), we further post-train the replaced layer on language modeling loss in order to have a fairer comparison with our method.

In our implementation, we use a transformer block from the original Qwen3-4B-Instruct-2507 model as the lightweight layer and train it on the same data and token budget as ADAPT (distilled). We allocate 80% of the total training budget to lightweight network training and the remaining 20% to post-training. The MSE stage runs for 192 Pile steps and 234 Nemotron steps, while the LM-loss stage runs for 48 Pile steps and 59 Nemotron steps. All stages use sequence length of 1024, with effective batch size of 2048 for Pile and 4096 for Nemotron. We use the training hyperparameters shown in Table 6. We perform evaluation following the same sampling strategy as discussed in Appendix C, using the non-thinking sampling parameters.

Hyperparameters	Values
LR (MSE)	2e-4
LR (LM-loss)	5e-5
LR scheduler	cosine
Warmup ratio	0.01
Optimizer	AdamW
Adam betas	(0.9, 0.999)
Adam epsilon	1e-8
Weight decay	1e-3
Max gradient norm	5.0
Mixed precision	bf16

Table 6: **Hyperparameters for LLM-Streamline training.**

I Additional Results for Adaptive Model-Size Selection

I.1 Input Difficulty Classification

As described in Section 4.4, we use the MATH dataset (Hendrycks et al., 2021b), which contains competition-style math problems with ground truth difficulty labels according to the Art of Problem Solving (AoPS) competition ratings (Art of Problem Solving, n.d.). For our task, we sample 2 separate subsets of 1050 problems from the original dataset, each with 210 problems for each difficulty level. We use one as the training set to determine the optimal model routing policy, and the other as a test set to measure the efficiency gains of our method.

To create the model routing policy, we require a lightweight and reliable method for estimating input difficulty. We test both the student and teacher models as classifiers by prompting them to assign a difficulty rating to each question. Specifically, each input question is paired with a prompt describing the AoPS difficulty scale (Figure 22), and the model is asked to output a rating from 1 to 10. We merge all levels above 5 into level 5, since we know *a priori* that the MATH dataset contains problems of at most level 5 difficulty. To obtain predictions efficiently, we perform a single forward pass and extract the predicted class directly from the output logits.

To evaluate classification performance, we use

Difficulty Grading Prompt

You will be given a math problem. Your job is to grade the difficulty level from 1-10 according to the AoPS standard.

Here is the standard:

All levels are estimated and refer to averages. The following is a rough standard based on the USA tier system AMC 8 - AMC 10 - AMC 12 - AIME - USAMO/USAJMO - IMO, representing Middle School - Junior High - High School - Challenging High School - Olympiad levels. Other contests can be interpolated against this.

Notes:

- Multiple choice tests like AMC are rated as though they are free-response.
- Some Olympiads are taken in 2 sessions, with similarly difficult sets of questions.

Scale

1: Problems strictly for beginners (e.g., AMC 8 1-20, AMC 10 1-10).
2: For motivated beginners (e.g., AMC 8 21-25, AMC 10 11-20).
3: Advanced beginner problems (e.g., AMC 10 21-25, AIME 4-6).
4: Intermediate-level problems (e.g., AMC 12 21-25, AIME 7-9).
5: More difficult AIME problems (10-12), simple Olympiad problems.
6: High-level AIME (13-15), introductory Olympiad questions.
7: Tougher Olympiad questions requiring technical knowledge.
8: High-level Olympiad questions.
9: Expert Olympiad questions.
10: Extremely difficult problems unsuitable for standard competitions.

Examples

<1: Counting edges/corners/faces of a cube (2003 AMC 8 Problem 1).
1: How many integers satisfy $|x| < 3\pi$? (2021 AMC 10B Problem 1).
2: Probability problem with die rolls (2021 AMC 10B Problem 18).
3: Geometry problem with area calculation (2018 AMC 10A Problem 24).
4: Recursive sequence problem (2019 AMC 10B Problem 24).
5: System of equations (JBMO 2020/1).
6: Complex geometry problem (2020 AIME I Problem 15).
7: IMO 2015 Problem 1 about balanced sets.
8: IMO 2014 Problem 5 about coin denominations.
9: IMO 2022 Problem 3 about prime arrangements.
10: IMO 2020 Problem 6 about point separation.

Task:

Problem to be labeled: {problem}.

Please put your estimation of the difficulty as a single number from 1-10.

Important: You MUST respond with only a single number between 1-10 indicating the difficulty of problem, not the solution of the problem.

Output: <difficulty level from 1-10>

Figure 22: **Prompt for difficulty classification.** Adapted from Waheed et al. (2025).

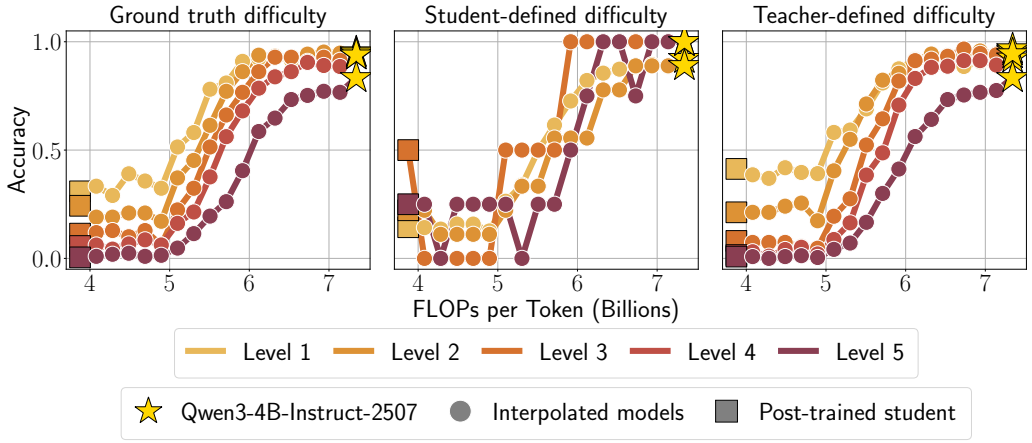

Figure 23: **Interpolation performance of Qwen3-4B-Instruct-2507 for different difficulty levels.** We find that the teacher-defined difficulty score produces similar interpolation curves to the ground truth difficulty scores, but student-defined difficulty is unreliable. This shows that the teacher-defined labels can act as a reasonable approximation for groundtruth difficulties.

Levels	Teacher	Student
1	184	1035
2	282	9
3	149	2
4	195	0
5	240	4

Table 7: **Breakdown of input-difficulty classification results from teacher and student models.** The student is unreliable for difficulty classification, as it chooses the lowest difficulty level for the vast majority of examples.

the mean absolute error (MAE):

$$\text{MAE}_{\mathcal{D}_{\text{train}}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{i \in \mathcal{D}_{\text{train}}} |\hat{y}_i - y_i|$$

where $\mathcal{D}_{\text{train}}$ denotes the training set, $\hat{y}_i$ is the predicted difficulty for the i^{th} problem, and y_i is the corresponding ground-truth difficulty.

On our training set, the teacher achieves an MAE of 0.970, while the student has an MAE of 1.995. This indicates that the teacher's predictions are typically within one difficulty level of the ground truth, whereas the student's predictions deviate by approximately two levels on average. This suggests that the teacher provides substantially more accurate estimates for adaptive model selection. The class breakdown of the teacher and student predictions is shown in Table 7.

While the MATH dataset provides human-annotated difficulty labels, prior work has shown that human difficulty does not always align with model-perceived difficulty (Kordi et al., 2025). Figure 23 verifies that teacher-predicted labels produce well-separated and consistently ordered performance curves, closely matching the trends from ground-truth labels, whereas student-predicted labels are noisy and do not yield clear separation. This further supports using the teacher for difficulty estimation in adaptive model selection.

We omit the cost of difficulty classification from the results in Section 4.4, as it requires only a single teacher forward pass and is negligible relative to the average response length of 3770.5 tokens.

I.2 Adaptive Model-Size Selection Results for Olmo and Llama

In Figures 24 and 25, we report the results for the adaptive model size selection experiments for Olmo-3-7B-Instruct and Llama-3.1-8B-Instruct. Note that we use Qwen3-4B-Instruct-2507 to provide the difficulty labels (same as in Section 4.4) as it is smaller than both the teacher models of Olmo and Llama, and therefore cheaper.

I.3 Discussion of Total Token Count

While smaller models offer more favorable FLOPs per token performance, we empirically observe that they often generate more tokens overall for the same set of questions. This is because smaller models are typically less capable, leaving more problems unsolved. On these questions, the model keeps generating tokens until it hits the `max_tokens` limit, whereas on questions it solves correctly, it completes them using far fewer tokens. We believe that with further training, this issue can be mitigated, but this is beyond the scope of this paper and we leave it for future work.

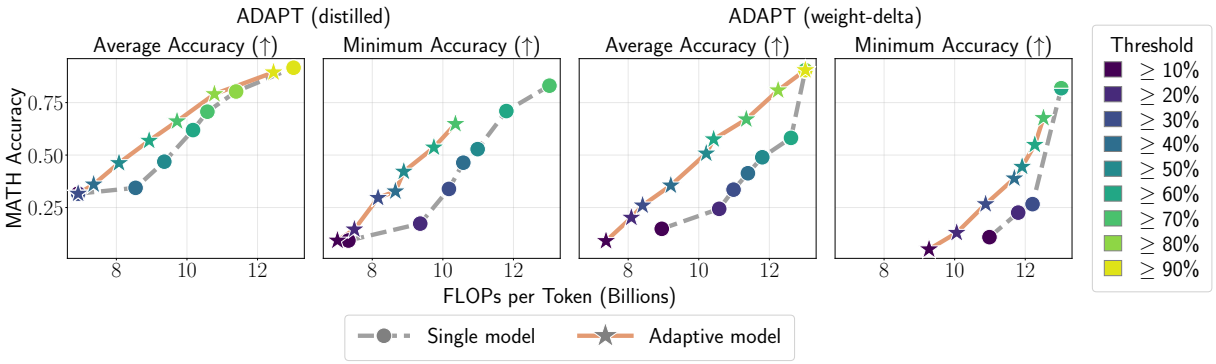

Figure 24: **Adaptive model size selection results for Olmo-3-7B-Instruct.**

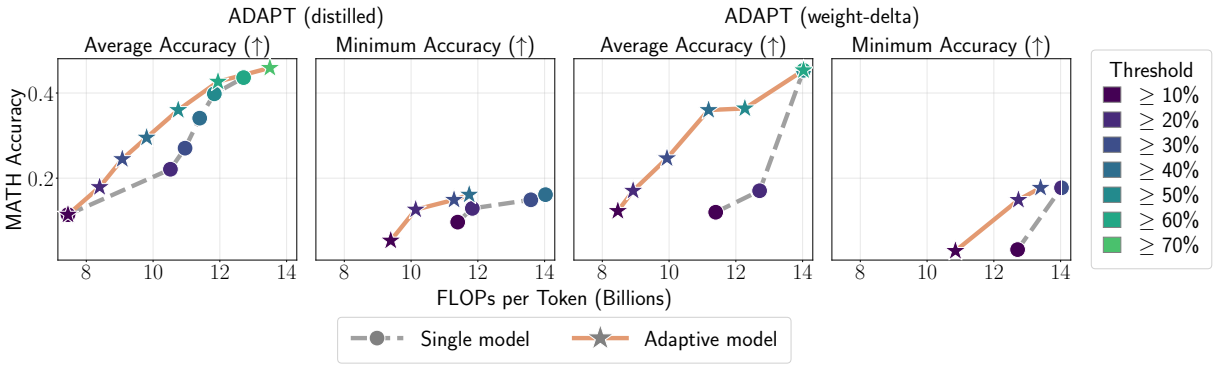

Figure 25: **Adaptive model size selection results for Llama-3.1-8B-Instruct.**

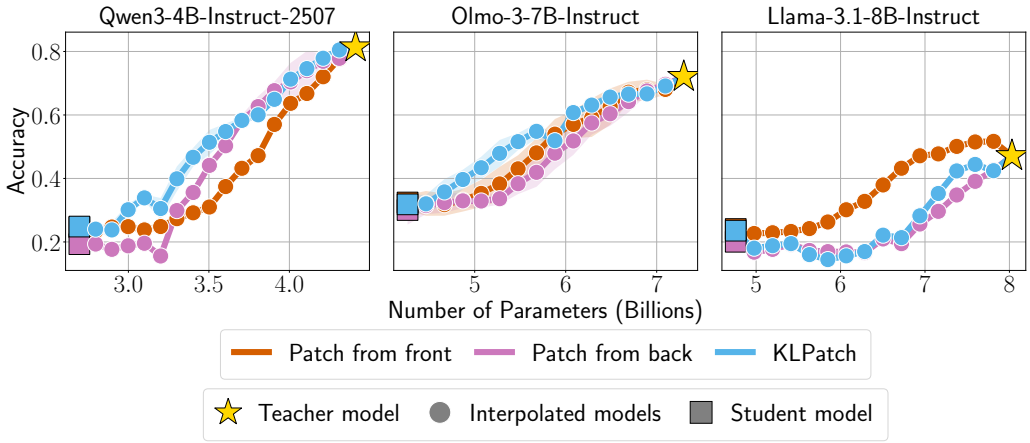

Figure 26: **Comparison of interpolation performance for different patching orders.** We test patching from the front, patching from the back, and KLPatch (Kangaslahti et al., 2026b). We find that KLPatch can further improve performance, indicating that more sophisticated patching strategies are orthogonal to our approach and can be used in conjunction with our student models.

J Patching Order Experiment

In Figure 26, we show the full interpolation curves for all three model families when testing front-to-back and back-to-front patching orders. We also test KLPatch, which is a recently proposed iterative greedy KL divergence-based algorithm for selecting the patching order (Kangaslahti et al., 2026b). At each size, KLPatch patches each student layer individually and selects the layer with minimum KL divergence to the teacher. We use a calibration set of 64 examples from the Nemotron non-thinking set in order to compute the KL divergence. These plots provide a more detailed view of how patching order affects interpolation behavior across model sizes.

We observe that the effect of patching order varies substantially across models. For Qwen3-4B-Instruct-2507 and Olmo-3-7B-Instruct models, KLPatch improves performance over patching from the front and patching from the back. However, for Llama-3.1-8B-Instruct, patching from the front yields the best performance. This suggests that the

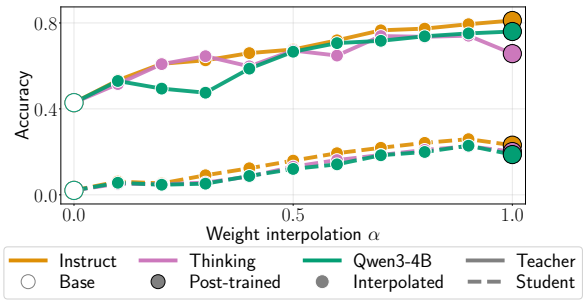

Figure 27: **Smooth weight interpolation results for generation accuracy.** Base and post-trained students display smooth interpolation behavior in downstream generation accuracy.

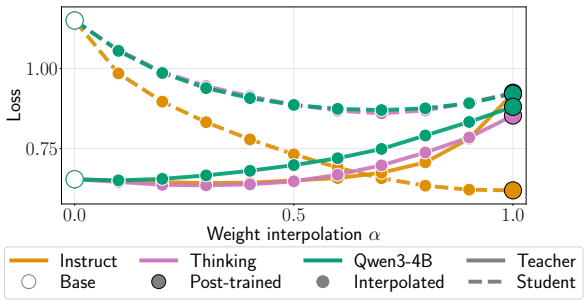

Figure 28: **Smooth weight interpolation results for loss on non-thinking Nemotron data.** Base and post-trained students display smooth interpolation behavior in loss evaluated on non-thinking Nemotron data.

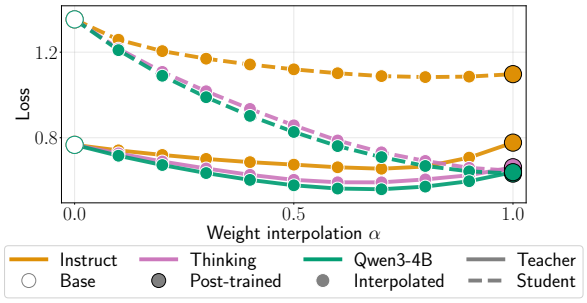

Figure 29: **Smooth weight interpolation results for loss on thinking Nemotron data.** Base and post-trained students display smooth interpolation behavior in loss evaluated on thinking Nemotron data.

optimal patching depends on how different layers contribute to downstream generation performance in each model and architecture. In our experiments, we use back to front for Qwen3-4B-Instruct-2507 and front to back for Olmo-3-7B-Instruct and Llama-3.1-8B-Instruct. As KLPatch improves performance for small Qwen and Olmo models, this indicates that more sophisticated patching methods can be used in conjunction with ADAPT to improve the performance of post-trained model families.

K Additional Smooth Weight Interpolation Results

In this section, we report the smooth weight interpolation results for generation accuracy (Figure 27) and loss on SFT data (Figures 28 and 29). We observe the same smooth weight interpolation behavior between the base and post-trained students across all three setups. The difference between Figures 28, 29, and Figure 5 from Section 4.5 can be attributed to the calibration data we used to calculate the loss terms, as different models (base, instruct, thinking) naturally have lower loss on different types of calibration data. Therefore, the results provide additional evidence that distilled students mirror teacher-level connectivity in the weight space.

L Training Dynamics

In Figure 30, we study how the interpolation performance behaves during the two-phase training. During the pre-training phase, the model converges to a lower interpolation performance curve very similar to the BD (pre-train) curve in Figure 2. The SFT phase on Nemotron data initially produces improved student model performance but has a dip in interpolation performance at higher model sizes,

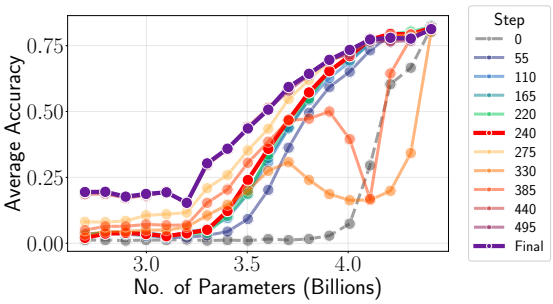

Figure 30: **Training dynamics study of two-phase distillation.** We find that two-phase distillation converges to two distinct interpolation curves during the pre-training and SFT phases, which allows the model to retain general-purpose and in-domain performance.

then converges to a solution with both improved student model performance and smooth interpolation behavior. Taken together, this indicates that both phases converge to distinct solutions with general purpose versus in-domain alignment to the teacher model.

M ADAPT for Additional Models and Tasks

In this section, we show the additional results for Qwen, Olmo and Llama models. We also show the performance of ADAPT on coding tasks.

M.1 Results for Qwen3-14B

In Figures 31, 32, and 33, we show the two-phase distillation and weight-delta initialization results for Qwen3-14B with both thinking and non-thinking modes.

M.2 Results for Olmo

In Figures 34 and 35, we show the two-phase distillation and weight-delta initialization results for Olmo-3-7B-Instruct and Olmo-3-7B-Think.

M.3 Results for Llama

In Figures 36 and 37, we show the two-phase distillation and weight-delta initialization results for Llama-3.1-8B-Instruct.

M.4 Results for Coding Tasks

In this section, we report the performance of ADAPT for coding tasks and show that it works beyond mathematical reasoning. We replace the math data with coding data for the SFT phase, sampled from the coding split of Llama Nemotron Post-Training Dataset (Bercovich et al., 2025). For ID tasks, we use MBPP+ (Austin et al., 2021), HumanEval+ (Chen et al., 2021), and LiveCodeBench (Jain et al., 2025). MBPP+ and HumanEval+ are augmented versions created with EvalPlus (Liu et al., 2023). We use the same benchmarks for OOD tasks (IFEval and MMLU-Redux).

In Figures 38 and 39, we show the two-phase distillation and weight-delta initialization results for Qwen3-4B under thinking mode. The results show that, similar to math tasks, two-phase distillation continues to outperform BD (pre-train) and BD (SFT) setups in both ID and OOD tasks. ADAPT (weight-delta) remains as a trade-off between the higher-compute upper bound ADAPT (distilled) and compute-matched baseline cross-patch (2-phase). It is cheaper than ADAPT (distilled) while performing better than cross-patch (2-phase).

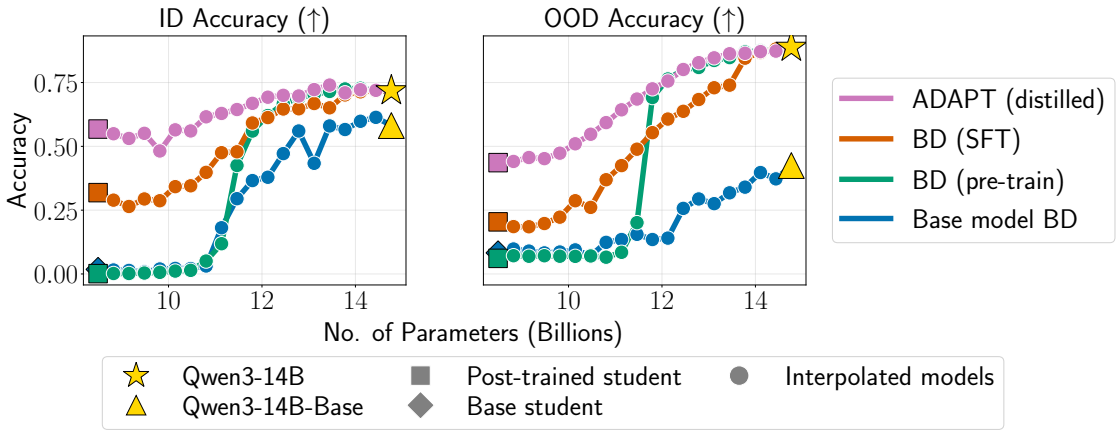

Figure 31: **ADAPT (distilled) results for Qwen3-14B (Thinking).** Two-phase distillation produces the highest and most stable performance over both ID and OOD tasks.

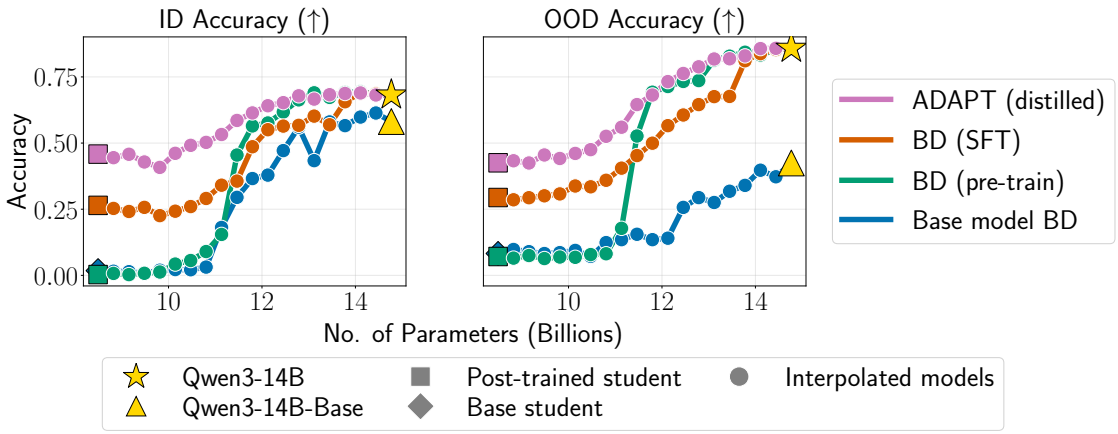

Figure 32: **ADAPT (distilled) results for Qwen3-14B (Non-Thinking).** Two-phase distillation produces the highest and most stable performance over both ID and OOD tasks.

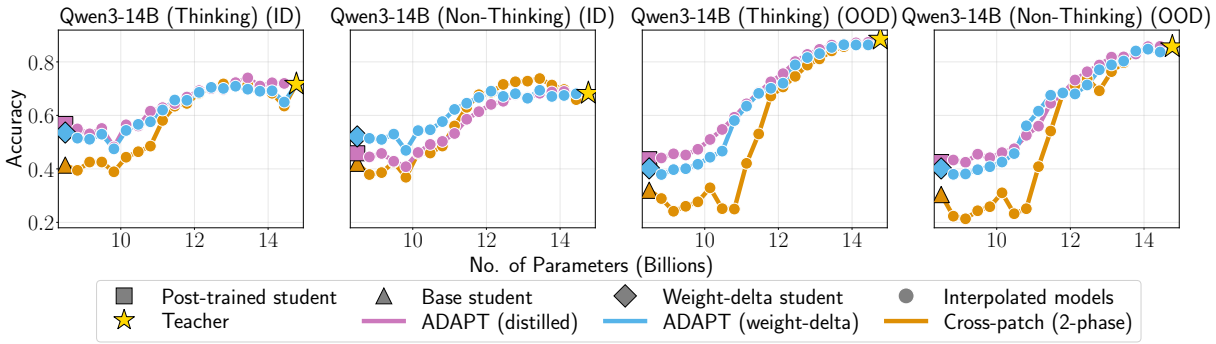

Figure 33: **ADAPT (weight-delta) results for Qwen3-14B with both thinking and non-thinking modes.** ADAPT (weight-delta) remains competitive with ADAPT (distilled) across both ID and OOD settings, while consistently outperforming cross-patch (2-phase).

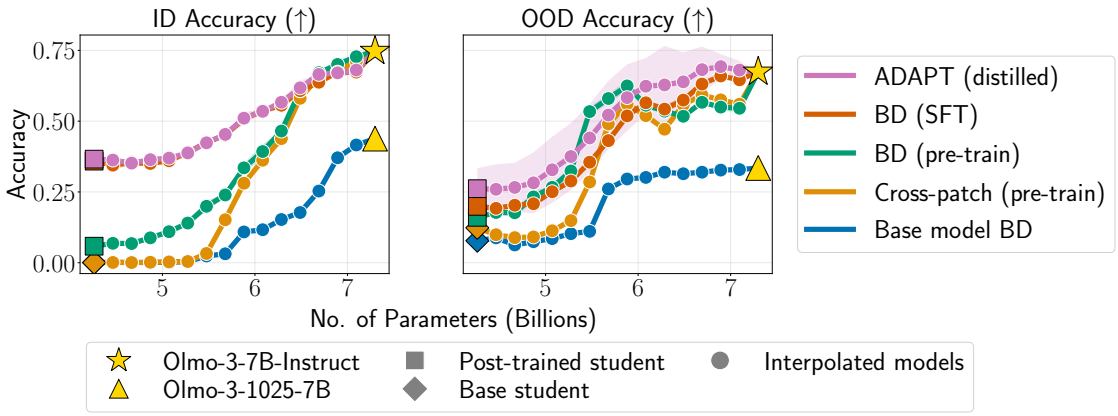

Figure 34: **ADAPT (distilled) results for Olmo models.** Two-phase distillation produces the highest and most stable performance over both ID and OOD tasks.

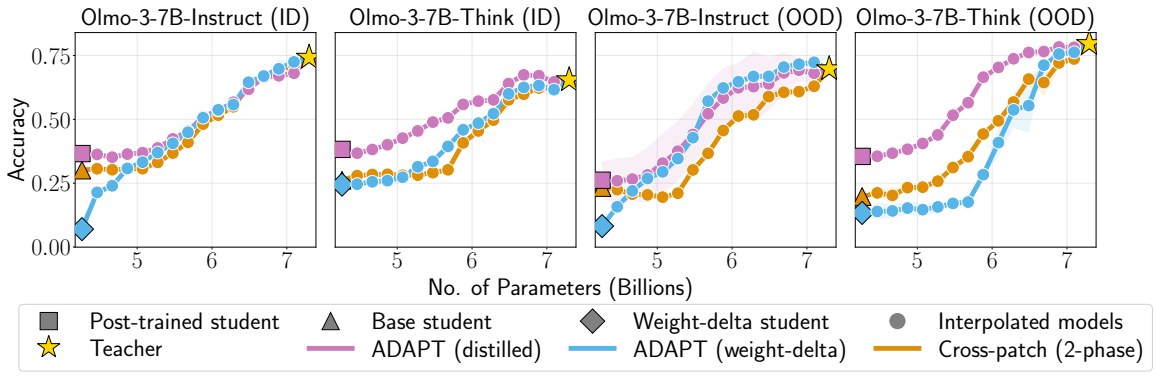

Figure 35: **ADAPT (weight-delta) results for Olmo models.** ADAPT (weight-delta) remains competitive with ADAPT (distilled) across both ID and OOD settings, while consistently outperforming cross-patch (2-phase) (with the exception of OOD performance for Olmo-3-7B-Think).

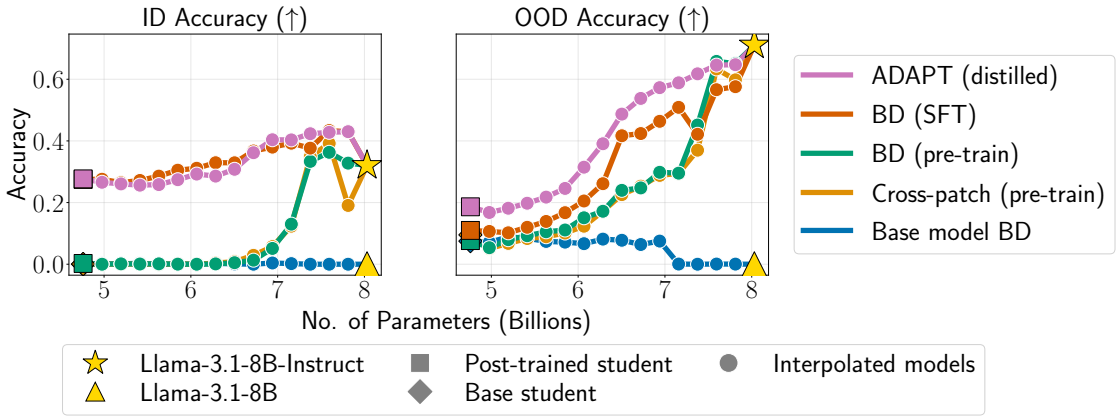

Figure 36: **ADAPT (distilled) results for Llama models.** Two-phase distillation produces the highest and most stable performance over both ID and OOD tasks.

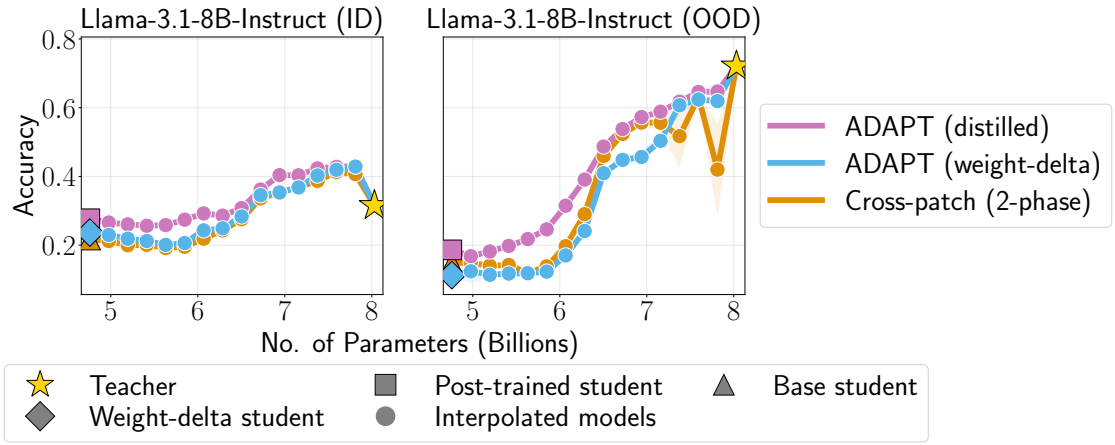

Figure 37: **ADAPT (weight-delta) results for Llama models.** ADAPT (weight-delta) remains comparable with ADAPT (distilled) across both ID and OOD settings.

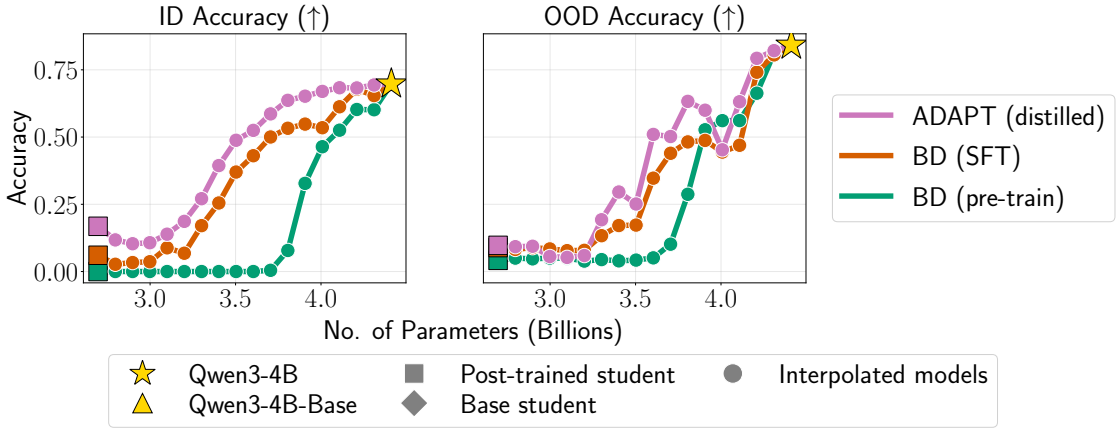

Figure 38: **ADAPT (distilled) results for Qwen3-4B on coding tasks.** Similar to the math setup, two-phase distillation produces the highest and most stable performance over both ID and OOD tasks. The evaluation is done on thinking mode only.

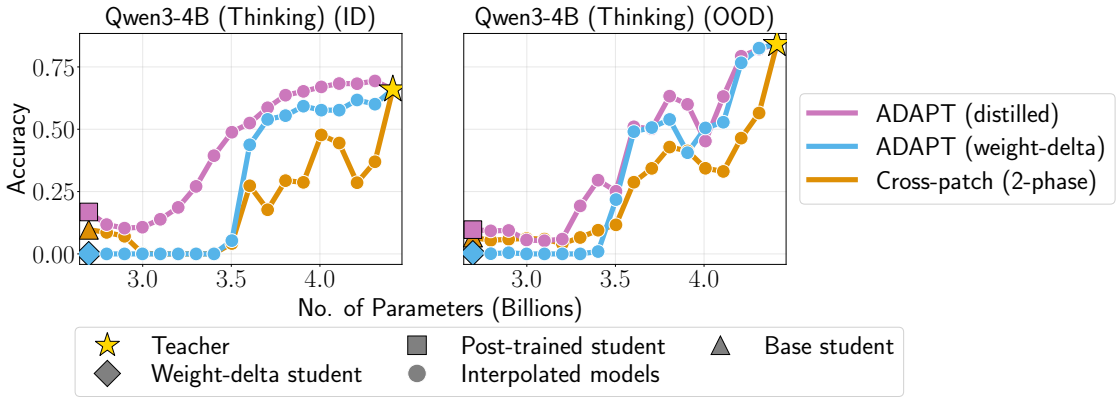

Figure 39: **ADAPT (weight-delta) results for Qwen3-4B on coding tasks.** ADAPT (weight-delta) maintains a trade-off between the higher-compute upper bound ADAPT (distilled) and compute-matched baseline cross-patch (2-phase). It is cheaper than ADAPT (distilled) while performing better than cross-patch (2-phase).

N Individual Benchmark Results

N.1 Two-phase Post-trained Distillation Results

In Figure 40, we show the interpolation performance of Qwen3-4B-Instruct-2507 on individual benchmarks.

N.2 Weight-delta Initialization Results

In Figure 41, we show the results from the weight-delta initialization experiments for Qwen model family on individual benchmarks.

O 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.

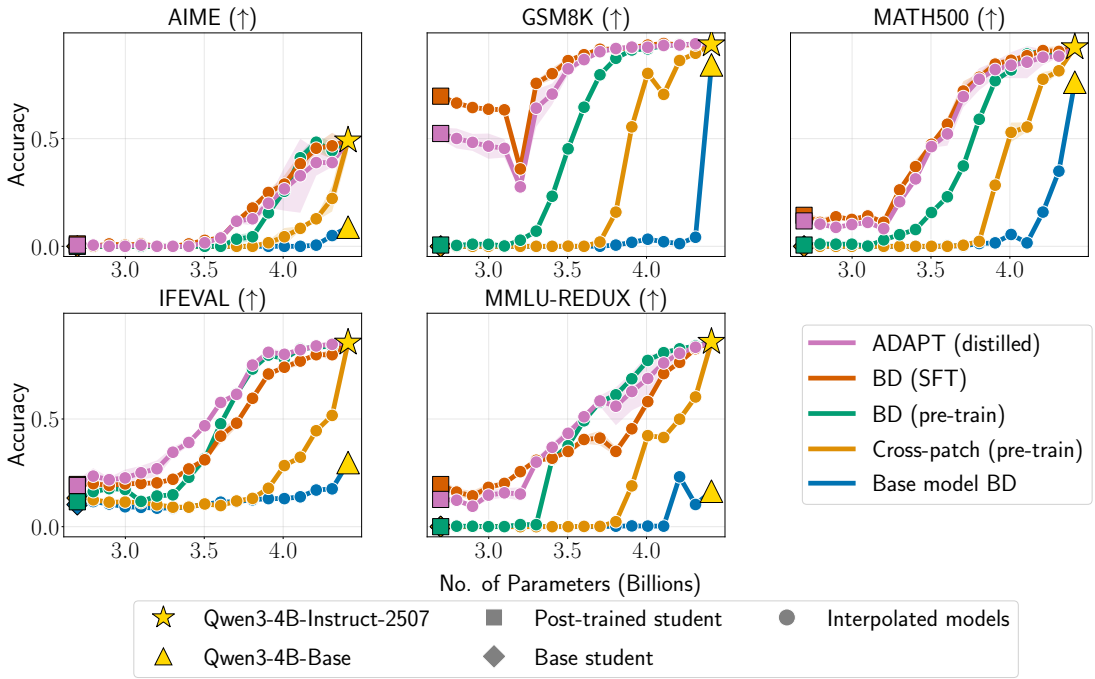

Figure 40: **Interpolation results for Qwen3-4B-Instruct-2507 on individual benchmarks.**

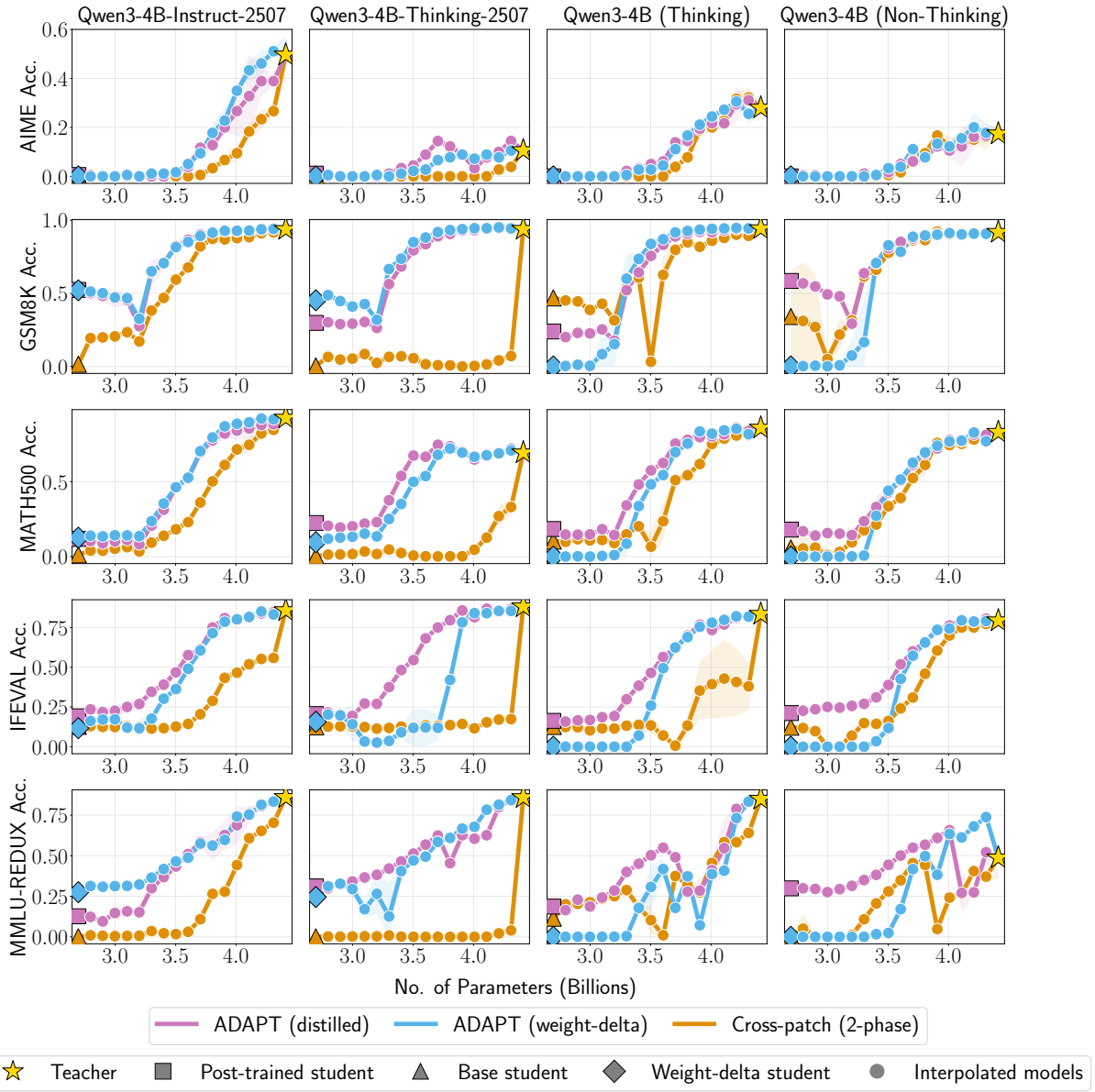

Figure 41: **Weight-delta initialization experiment results for Qwen 4B model family on individual benchmarks.**