Mixture-Trained Merging for Unified Multi-Objective Models

SeongHyeon Kim[1*] **Chaeyun Jang**[1*] **Seungyoo Lee**[1] **Jiyeon Ham**[2] **Yunju Bak**[2]
Boseop Kim[2] **Juho Lee**[1†]
[1]KAIST [2]Kakao
{shkim0824, jcy9911, punctuate, juholee}@kaist.ac.kr
{chloe.py, juliet.bak, mat.mul}@kakaocorp.com

Abstract

Unified language models are increasingly expected to combine heterogeneous capabilities, such as mathematics, code, instruction following, and controllable thinking behavior, within a single set of parameters. A common solution is sequential post-training on multiple objectives, but this entangles all objectives along one optimization trajectory and makes the final model highly sensitive to training order, data ratios, schedules, and stopping criteria. Weight-space merging offers a modular alternative, but naive merging of single-objective experts often fails: domain capabilities degrade sharply, or `think`/`non-think` modes collapse into one dominant behavior. We attribute both failures to incompatible weight-space geometry: experts trained on single objectives drift to distant regions of parameter space, placing their interpolations outside any shared low-loss basin. We propose **Mixture-Trained Merging (MTM)**, which trains each branch on an objective-biased data mixture rather than a single objective, exposing it to cross-objective interactions and making branches compatible at merge time. MTM uses merged-model evaluations as a low-cost signal for selecting branch mixtures, avoiding expensive data-mixture ablations. The procedure is iterative: each round promotes the base model using globally selected merge coefficients and refines each branch mixture using domain-preferred coefficients under constraints that preserve other objectives. To scale beyond simplex grid search, MTM uses qNEHVI-based multi-objective Bayesian optimization. Across code, mathematics, instruction following, and `think`/`non-think` control, MTM outperforms naive merging and preserves behavioral separation where single-objective merging collapses, suggesting that effective unified models require branches trained to be mergeable.

1 Introduction

Recent large language models (LLMs) increasingly aim to unify diverse capabilities within a single model, including complex reasoning, direct response generation, coding, and instruction following [1, 2, 3, 4]. Beyond performance, such unified models are practically appealing because they reduce the need for separate model stacks and offer a simpler deployment interface across heterogeneous workloads.

A common practical approach to building such unified models is single-trajectory training, in which a model is post-trained across multiple objectives within a shared set of parameters [2, 4, 3]. While conceptually simple, this strategy introduces substantial engineering overhead. The final performance is often highly sensitive to training order, data mixing ratios, and scheduling choices, because these

*Equal contribution.
†Corresponding author.

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

factors strongly affect objective interference and catastrophic forgetting [5, 6, 7]. As a result, finding a robust recipe typically requires extensive hyperparameter tuning over near-full-budget runs, making unified model construction costly and difficult to iterate on.

Weight-space merging, which trains specialized branches separately and then combines them, offers an appealing alternative to optimizing all objectives along a single training trajectory [8, 9, 10]. This makes unified model construction more modular and avoids much of the optimization complexity of single-trajectory training. Merging aims to combine specialized branches into a single model that performs well on all target objectives. However, we empirically find that merging single-objective experts does not always lead to such a model. For example, when constructing a hybrid model that should support both non-think and think modes, merging a non-think expert with a think expert collapses the two modes toward the branch with the larger merge coefficient, producing uniformly verbose or concise outputs regardless of the requested format (Figure 3). Similarly, when constructing a multi-domain model that should handle both code and math, merging a code expert with a math expert leads to broad downstream degradation across both domain accuracy, as shown in Figure 2.

We hypothesize that these failures arise because independently trained branches drift into different regions of weight space. Consistent with this view, we observe substantial accuracy drops along the linear interpolation path between high-performing branches, indicating that they do not lie on an accuracy-preserving path [11, 12]. This suggests that mergeability must be encouraged during training rather than imposed only through post-hoc merging. We therefore train each branch on a controlled **mixture** of objectives. By exposing branches to non-primary objectives, mixture training reduces inter-branch divergence while preserving specialization. Unlike single-trajectory multi-objective training, which searches for a single global data ratio, our setting requires branch-specific mixtures that yield specialized yet mutually mergeable branches.

To identify these mixtures, we use the connection between weight-space merging and data-mixture training established by [13]: merge coefficients can approximate data-mixture ratios under first-order training dynamics. We use merged performance as a proxy signal and optimize per-branch mixtures with Multi-Objective Bayesian Optimization (MOBO) [14, 15]. Rather than running mixture search as a separate stage, we integrate it into an iterative branch specialization and merging procedure. Each round selects mixtures, continues branch training, and merges branches back into a common reference model to control divergence. Thus, mixture selection, specialization, and realignment are coupled within a single training loop.

Our contributions are summarized as follows:

- We show that merging hard single-objective experts induces task interference, manifesting as mode collapse in think/non-think merges and broad degradation in domain-level merges.
- We connect this failure to weight-space incompatibility, showing that independently trained branches exhibit accuracy barriers along linear interpolation paths.
- We introduce iterative branch specialization and merging, which optimizes branch-specific objective mixtures to preserve both specialization and mergeability.
- We demonstrate consistent gains on Qwen3-4B and OLMo-7B across diverse math, code, and instruction-following benchmarks.

2 Related Works

2.1 Building Unified Models via Data Mixtures

Modern LLMs are expected to handle multiple objectives such as math, code, and instruction following within a single model [2, 16], which makes the choice of the SFT data mixture a central design decision. A growing line of work therefore aims to determine mixture ratios in a principled way. DoReMi [17] reweights domains via a small proxy model trained with group distributionally robust optimization. RegMix [18] fits a regression model from many small-scale runs to predict performance of unseen mixtures. Data mixing laws [19] formulate scaling laws for the mapping from mixture proportions to validation loss. MergeMix [13] bypasses surrogate training entirely by approximating mixture-trained models with linear weight-space merges of objective-specific fine-tunes. Even with these advances, identifying a single global mixture that simultaneously balances all objectives still relies on heuristic choices and remains difficult to tune as the set of objectives grows.

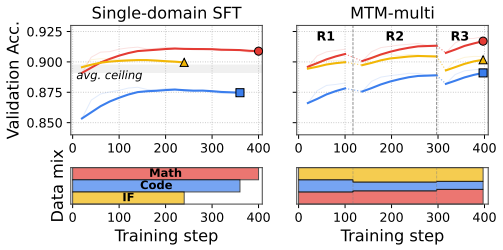


Figure 1: **Training dynamics of single-domain SFT vs. MTM-Multi in Qwen3-4B-Base.** MTM-Multi replaces one long heterogeneous trajectory with round-wise mixture-trained branches and promoted merges.

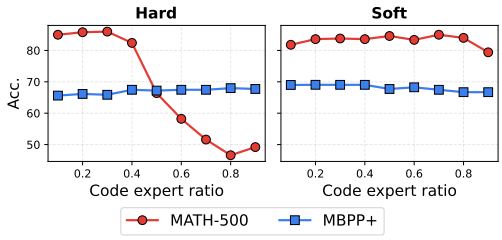


Figure 2: **Domain-expert merging on Qwen3-4B-Base.** Following Section 5, **Hard** branches are trained on single domains, whereas **Soft** branches use complementary Math/Code mixtures with a $(0.7, 0.3)$ split.

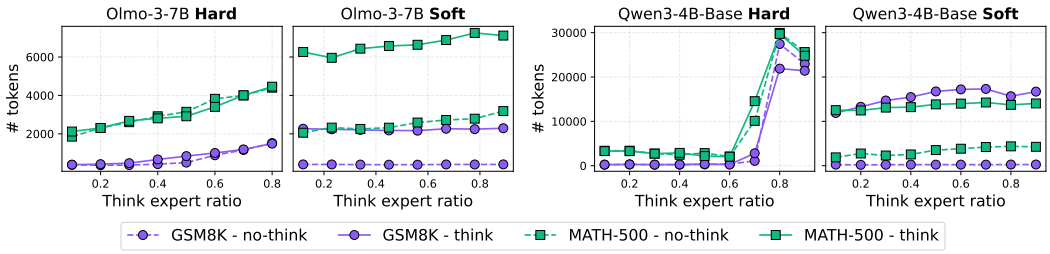


Figure 3: **Mode-expert merging on OLMo-7B and Qwen3-4B-Base.** Following Section 5, **Hard** branches are trained on single response modes, while **Soft** branches use complementary `think`/`non-think` mixtures with a $(0.7, 0.3)$ split. Merging hard experts loses mode control under large branch imbalance, leading to outputs that follow the dominant branch (think or non-think) regardless of the prompt, whereas mixture-trained branches better preserve mode distinction.

2.2 Weight-Space Model Merging

Model merging offers a parameter-space alternative to multi-objective training, combining multiple specific objective models into a single model without increasing inference cost [20, 10, 9, 21, 22]. Model Soups [20] showed that averaging weights from multiple fine-tuned checkpoints can outperform the best individual checkpoint, while task arithmetic [10] generalized this idea by representing fine-tuned models as task vectors and combining them through linear operations in weight space. For LLMs, Branch-Train-Merge [8] trains domain experts from a shared branching point to keep them close enough in parameter space to remain mergeable.

A central challenge in merging is parameter interference between objectives. Several methods address this interference post hoc: TIES Merging [9] trims low-magnitude updates and resolves sign conflicts, DARE [21] sparsifies and rescales SFT deltas as a plug-in for existing mergers, and SCE [22] sets fine-grained fusion coefficients from update magnitudes. More recently, application-specific mergers such as RAIN-Merging [23] target the fusion of reasoning and instruction-tuned models by constraining thinking-format-aware subspaces. In contrast to these post-hoc correction methods, MTM does not take independently trained experts as fixed. Instead, it trains each branch on a domain-biased mixture, shaping task vectors to be mutually compatible before merging.

3 Background and Motivation

Problem Setting We start from a base model $\mathcal{M}_0 = \mathcal{M}(\theta_0)$ and K objectives, each with a training corpus $\mathcal{D}_k$ and an evaluation metric f_k normalized to a common scale. We seek a single parameter vector θ such that $\mathcal{M}(\theta)$ performs well on all $\{f_k\}_{k=1}^K$. Throughout, Δ^{K-1} denotes the $(K-1)$-simplex, $\textsc{Mix}(\mathcal{D}_{1:K}; q)$ the mixture corpus that draws from $\mathcal{D}_k$ with probability q, and $\textsc{SFT}(\mathcal{M}, \mathcal{D})$ supervised fine-tuning of $\mathcal{M}$ on $\mathcal{D}$. Given K models $\{\mathcal{M}_k\}_{k=1}^K$ fine-tuned from a common reference

$\mathcal{M}_{\text{ref}} = \mathcal{M}(\theta_{\text{ref}})$ and coefficients $\alpha \in \Delta^{K-1}$, we write

$$\textsc{Merge}(\mathcal{M}_{1:K}; \alpha) \;=\; \mathcal{M}\left(\theta_{\text{ref}} + \sum_{k=1}^{K} \alpha_k(\theta_k - \theta_{\text{ref}})\right) \tag{1}$$

for the linear merge.

A standard approach to building such a multi-objective model is to train a single model on an appropriately chosen global data mixture. However, Figure 1 shows that single-domain SFT runs on math, code, and instruction following saturate at different training steps and at different metric values. A single-trajectory recipe must therefore compress this heterogeneity into one training schedule. This makes recipe design a high-dimensional engineering problem in which the mixture ratio, stage ordering, learning-rate schedule, and stopping criterion must all be chosen so that no objective is undertrained while others have already saturated, despite their intrinsically mismatched saturation profiles.

A natural alternative is to train objective-specific branches in parallel and combine them in weight space. For each k, we fine-tune a hard expert $E_k = \text{SFT}(\mathcal{M}_0, \mathcal{D}_k)$ and merge with weights $\alpha \in \Delta^{K-1}$ as $\mathcal{M}_{\textsc{merge}}(\alpha) = \textsc{Merge}(E_{1:K}; \alpha)$. This decouples branches and reduces unified-model construction to many cheap, independent fine-tunes plus a simple mixing step. Most existing weight-space merging methods [10, 8] operate in this regime: take hard experts as given, then post-hoc combine them.

3.1 Failure Modes of Hard-Expert Merging

In this section we examine two failure modes that arise across our experimental settings. First, hard-expert merging exhibits a sharp trade-off between target domains: gains on one domain come at the cost of another, so no merge ratio recovers all per-domain experts at once. Second, when merging a non-think expert with a think expert to obtain a hybrid model, the resulting model collapses into a single mode rather than retaining both behaviors as intended. Both failure modes are alleviated when the branches are trained on objective mixtures rather than as hard experts, motivating the approach we develop in Section 4.

Specialization at the cost of other domains. Hard experts specialize in their target domain, but underperform the base model in other domains. As a result, their linear merge recovers expert-level performance only near the simplex endpoints, as shown in the left panel of Figure 2. As the code-expert ratio increases, MBPP+ accuracy improves, but MATH-500 accuracy drops sharply from about 85 to below 50. Thus, no interior merge ratio reaches expert-level performance on both objectives. In contrast, the right panel shows that mixture-trained branches maintain strong performance on both tasks across a wide range of merge ratios, indicating that mixture-trained components are easier to combine.

Mode collapse between qualitatively different branches. The problem becomes more pronounced when branches encode different generation behaviors. A non-thinking expert is trained to give short direct answers, whereas a thinking expert produces longer Chain-of-Thought (CoT) reasoning. A useful hybrid model should preserve both behaviors and switch between them according to an explicit mode signal, but linear interpolation of hard experts fails to do so. In the left panels of Figure 3, generation lengths under `no-think` and `think` prompts move together as the think-expert weight changes, so the mode gap largely disappears across the interior of the simplex; Qwen3-4B-Base additionally shows a length blow-up at one merge ratio. In contrast, the right panels show that mixture-trained branches preserve the separation across merge ratios, maintaining short `no-think` outputs and long `think` outputs simultaneously.

4 MTM: Mixture-Trained Merging

We now present Mixture-Trained Merging (MTM), which addresses the failures of hard-expert merging identified in Section 3 by training each branch on a mixture of objectives. MTM has two variants. The single-round variant (Section 4.1), **MTM-Single**, trains mixture-biased branches once and selects merge weights on top of them using the MergeMix approximation. The iterative variant

(Section 4.2), **MTM-Multi**, spreads the same budget across several shorter rounds and reuses each round's merged model as the starting point of the next, so that mixture search and model construction share a single training trajectory. The candidate merge weights evaluated in either variant are produced by multi-objective Bayesian optimization over the simplex (Section 4.3).

4.1 Merge-Based Mixture Search

Section 3 showed that merging hard experts fails because the experts drift into mutually incompatible regions of weight space. MTM addresses this by replacing each hard expert with a *soft-expert*: a branch trained on a domain-biased *mixture* of objectives. Each soft-expert remains specialized toward its target domain keeping it close enough to the other branches in weight space. The unified model is then constructed by merging these K soft-experts. We initialize the reference model as the base model, $\mathcal{M}_0 = \mathcal{M}(\theta_0)$, and associate each branch $k \in [K]$ with its own mixture $p_k \in \Delta^{K-1}$ that is biased toward domain k. Each soft-expert is trained on its mixture,

$$E_k = \text{SFT}(\mathcal{M}_0, \textsc{Mix}(\mathcal{D}_{1:K}; p_k)), \tag{2}$$

and the K soft-experts are combined with merge weights $\alpha \in \Delta^{K-1}$ to form the final unified model,

$$\mathcal{M}(\alpha) = \textsc{Merge}(E_{1:K}; \alpha), \qquad F^{(j)}(\alpha) = f_j(\mathcal{M}(\alpha)), \tag{3}$$

where $F^{(j)}(\alpha)$ is the score of the merged model on objective j. Crucially, each p_k is a *branch-specific* mixture: branch k leans toward $\mathcal{D}_k$ but contains controlled amounts of the other objectives, so that the K branches together cover the simplex of objectives from K different angles rather than collapsing onto a single shared mixture.

This raises the question of how to choose the K branch mixtures $\{p_k\}_{k=1}^K$ without fine-tuning a new model for every candidate. For this *search* step, we borrow the approximation of [13]: in a local fine-tuning regime, the behavior of a linear merge $\textsc{Merge}(E_{1:K}; \alpha)$ approximates that of a model trained on the corresponding data mixture $\textsc{Mix}(\mathcal{D}_{1:K}; \alpha)$. We can therefore evaluate candidate mixture ratios by evaluating merge weights on already-trained branches, turning mixture search into an evaluation problem over the merge simplex. Whereas MergeMix uses this approximation to pick a single global mixture and trains one model on it, MTM uses it to pick K branch-specific mixtures and deploys their merge. We refer to the resulting single-round procedure as **MTM-Single**.

4.2 Iterative Merging with Branch Reuse

MTM-Single decouples mixture search from training, but at a cost: the branches trained for search and the model deployed at the end are different objects, so producing a final unified model still requires a second, full training pass on the chosen mixture. The total budget is roughly twice that of a single-trajectory baseline, and the final model never benefits from the merge evaluations carried out during search.

A more efficient alternative is to spread the same total budget over several shorter rounds, and at each round use the merge-based search of Section 4.1 not only to inform the next mixture but also to construct the model itself. The merged model selected at round t becomes the reference model from which round $t{+}1$ branches are trained, so the search trajectory and the training trajectory are the same sequence of models. We call this iterative variant **MTM-Multi**. Each round t performs three steps:

1. train branches $E_{1:K}^{(t)}$ from the current reference model $\mathcal{M}_{t-1}$ on mixtures $\{p_k^{(t)}\}$;
2. evaluate a set of candidate merges over the simplex to obtain a score landscape $\{F_t^{(j)}(\alpha)\}$;
3. use this landscape to choose both the next reference model $\mathcal{M}_t$ and the next-round mixtures $\{p_k^{(t+1)}\}$.

Reusing the round-t branches in step (iii) – rather than discarding them after search – is what brings the total training cost back in line with a single-trajectory baseline while preserving branch-level modularity. The remainder of this subsection details steps (i)–(iii).

Training mixture-trained branches. We initialize the first-round mixtures at the simplex vertices, $p_k^{(1)} = e_k$ for $k \in [K]$, so round 1 starts from hard-domain branches. Round-t branches are obtained

by applying Eq. (2) from the current reference model $\mathcal{M}_{t-1}$ with branch mixtures $\{p_k^{(t)}\}$. Round 1 thus trains pure-domain experts from $\mathcal{M}_0$, and later rounds train mixture-biased branches from a merged starting point.

Promoting the reference model. After evaluating a candidate set $\mathcal{X}_t \subset \Delta^{K-1}$, whose construction is deferred to Section 4.3, we select a global merge weight that maximizes mean normalized performance,

$$\alpha_\star^{(t)} \;=\; \arg\max_{\alpha \in \mathcal{X}_t} \; \frac{1}{K} \sum_{j=1}^{K} F_t^{(j)}(\alpha), \tag{4}$$

and promote the corresponding merged model to the next-round reference,

$$\mathcal{M}_t \;=\; \textsc{Merge}\Big(E_{1:K}^{(t)};\, \alpha_\star^{(t)}\Big)\,. \tag{5}$$

This step plays a dual role. It produces the unified model that MTM ultimately returns at round T, and it acts as a *realignment* operation between rounds: collapsing the branches back into a common reference prevents the cumulative drift that otherwise breaks weight-space compatibility over long fine-tuning horizons.

Updating branch mixtures. For each branch k, we look for a merge weight that produces a strong domain-k specialist *without* sacrificing the other objectives,

$$\alpha_k^{(t)} \;=\; \arg\max_{\alpha \in \mathcal{X}_t} \; F_t^{(k)}(\alpha) \quad \text{s.t.} \quad F_t^{(j)}(\alpha) \;\geq\; \tau_j^{(t)}, \quad \forall j \neq k, \tag{6}$$

where $\tau_j^{(t)}$ is a per-objective preservation threshold expressed on normalized scores, e.g., the score of $\mathcal{M}_{t-1}$ on objective j. The selected $\alpha_k^{(t)}$ then propagates back to the data mixture used to train branch k in the next round,

$$p_k^{(t+1)} \;=\; \sum_{\ell=1}^{K} \alpha_{k,\ell}^{(t)}\, p_\ell^{(t)}. \tag{7}$$

By the MergeMix approximation, $\alpha_k^{(t)}$ identifies a point in mixture space that sharpens domain k while preserving other objectives, and Eq. (7) uses this point as the next-round training mixture for branch k. Composing these updates across rounds carries information from each round's merge evaluations into the next-round mixture, so $p_k^{(t)}$ reflects the full search trajectory rather than the latest round alone.

4.3 Multi-Objective Bayesian Optimization over the Merge Simplex

The candidate set $\mathcal{X}_t$ in Eqs. (4) and (6) is constructed by MOBO over the merge simplex. Each candidate $\alpha \in \Delta^{K-1}$ defines a merged model $\mathcal{M}^{(t)}(\alpha)$ and a normalized score vector

$$\mathbf{F}_t(\alpha) = \big(F_t^{(1)}(\alpha), \ldots, F_t^{(K)}(\alpha)\big), \qquad F_t^{(j)}(\alpha) = f_j\Big(\mathcal{M}^{(t)}(\alpha)\Big)\,. \tag{8}$$

A dense grid over Δ^{K-1} becomes expensive as K grows, while a coarse grid can miss useful interior trade-offs. We therefore use MOBO to adaptively propose merge weights that expand the Pareto frontier of $\mathbf{F}_t(\alpha)$.

Each round is initialized with a small anchor set consisting of the simplex vertices $\{e_k\}_{k=1}^K$, the uniform weight $\frac{1}{K}\mathbf{1}$, and Dirichlet samples. Starting from these observations, BO proposes additional batches of merge weights under the simplex constraint $\alpha \in \Delta^{K-1}$. We use qNEHVI [24] as the acquisition function, since it handles noisy batched evaluations and directly targets hypervolume improvement over multiple objectives.

MOBO is used only to construct the Pareto-aware candidate set $\mathcal{X}_t$, not to directly choose the final merge. Once $\mathcal{X}_t$ is obtained, MTM applies the scalarized selection rules in Eqs. (4) and (6): the averaged score promotes the next reference model, while constrained objective-specific scores update the branch mixtures. Thus, one batch of merge evaluations per round supports both model promotion and all branch-specific mixture updates. Algorithm 1 summarizes the full procedure.

Table 1: **Results for Qwen3-4B-Base and OLMo-7B under the SFT setting (3-eval pass@1 avg ± std). Bold** marks the best result on the Avg. column and underline marks the second best.

	Qwen3-4B-Base						OLMo-7B					
Method	GSM8K	MATH	MBPP+	LCB	IFEval	Avg.	GSM8K	MATH	MBPP+	LCB	IFEval	Avg.
Math Expert	92.85±0.22	84.40±0.93	64.02±1.63	20.59±0.48	55.95±0.39	63.56±0.73	89.84±0.65	79.20±1.39	54.32±0.55	18.58±0.46	38.92±1.24	56.17±0.86
Code Expert	79.53±6.63	53.85±11.13	67.46±0.49	22.03±1.40	67.99±0.53	58.17±4.04	63.46±3.72	66.20±0.80	63.67±0.15	16.01±0.51	50.51±1.29	51.97±1.29
IF Expert	90.01±0.47	76.40±0.71	67.92±0.82	18.72±0.78	80.57±0.50	66.72±0.66	67.65±2.10	52.07±2.91	63.67±0.81	14.61±1.11	76.06±1.51	54.81±1.69
Linear Merge	91.91±0.38	83.10±0.59	67.72±0.42	23.40±0.29	77.26±3.05	68.68±0.95	83.83±1.30	75.00±0.53	64.55±0.00	14.61±0.64	61.62±2.10	59.92±0.91
Ties Merge	92.04±0.50	83.07±0.50	68.08±1.46	23.35±1.91	78.30±1.18	68.97±1.11	87.87±0.39	76.20±1.97	63.05±1.76	14.90±1.04	60.76±2.35	60.56±1.50
DARE-Ties	91.61±0.38	81.67±2.00	64.20±1.76	19.53±0.71	76.77±1.42	66.76±1.25	88.40±0.85	76.73±0.90	59.26±0.79	16.45±0.64	60.74±1.94	60.32±1.03
SCE	91.43±0.99	83.67±2.12	70.99±0.93	22.03±1.38	74.29±0.38	68.48±1.16	81.15±1.21	75.53±2.05	59.61±4.71	14.24±3.77	56.95±1.80	57.50±2.71
MergeMix	88.64±0.87	80.35±1.69	67.46±0.82	21.81±0.64	80.65±0.65	67.78±0.93	89.21±0.61	76.93±0.50	62.17±0.26	12.92±0.25	74.81±0.50	63.21±0.43
MTM-Single	90.88±0.90	82.45±1.02	69.11±0.80	22.69±0.72	82.87±0.62	69.60±0.81	89.76±0.66	78.13±0.46	62.35±1.36	13.00±0.58	74.10±0.15	63.47±0.64
MTM-Multi	92.32±0.93	82.40±0.80	68.78±0.75	21.04±0.62	84.16±0.59	**69.74**±0.74	90.32±0.83	78.27±1.40	64.64±1.19	13.36±0.64	73.89±0.78	**64.10**±0.97

5 Main Experiments

We evaluate MTM in two settings that reflect the failure modes identified in Section 3: a **multi-domain** setting that merges math, code, and instruction-following branches (Section 5.2), and a **hybrid-mode** setting that merges non-think and think branches (Section 5.3).

5.1 Experimental Setup

Training. We use two open base models that differ in scale and pretraining recipe: Qwen3-4B-Base [2] and OLMo-3-1025-7B (referred to as OLMo-7B) [16]. For the multi-domain setting, we build math, code, and instruction-following data from Dolci-Instruct-SFT[3] by sampling 10K examples from each corresponding category. We generate model-specific trajectories using Qwen3-4B-Instruct[4] for Qwen3-4B and OLMo-Instruct[5] for OLMo-7B. For the hybrid-mode setting, we sample 10K non-think examples from Dolci-Instruct-SFT and 10K think examples from Dolci-Think-SFT-32B[6], generating think trajectories with Qwen3-4B[7] in think mode and the OLMo3-7B-Think for the respective experiments. Mixture datasets contain 10K examples sampled from the branch-specific datasets, matching the total training size of each individual branch. See Section A for details.

Benchmarks. We evaluate along three axes that correspond to the target objectives. For *Math*, we report accuracy on GSM8K [25] and MATH-500 [26]. For *Code*, we report MBPP+ [27] and LiveCodeBench (LCB) [28]; For *IF (Instruction Following)*, we report IFEval [29]. In the hybrid setting we additionally report the average response length (in tokens) under both `no-think` and `think` prompts, which measures whether the merged model preserves behavioral separation between the two modes. See Section A for evaluation prompts and additional details.

Baselines. We compare MTM against several baselines constructed from the same set of single-objective experts. **(1) Linear Merge** [10] linearly combines expert updates, with merge weights chosen by grid search over the simplex. **(2) MergeMix** [13] uses the selected merge weights as data-mixture ratios and trains a single model on the resulting mixture, rather than merging at deployment time. **(3) TIES-Merge** [9] resolves interference by sparsifying updates, electing a shared sign, and averaging sign-consistent parameters. **(4) DARE** [21] randomly drops and rescales update entries, and we apply it as a preprocessing step before TIES. **(5) SCE** [22] assigns matrix-wise coefficients based on update magnitudes to enable fine-grained fusion without hard sparsification.

5.2 Multi-Domain

Table 1 shows that MTM-Multi achieves the highest average score on both Qwen3-4B-Base and OLMo-7B, outperforming all merging-only baselines. Across both models, MTM is the only method

[3] https://huggingface.co/datasets/allenai/Dolci-Instruct-SFT
[4] https://huggingface.co/Qwen/Qwen3-4B-Instruct-2507
[5] https://huggingface.co/allenai/Olmo-3-7B-Instruct
[6] https://huggingface.co/datasets/allenai/Dolci-Think-SFT-32B
[7] https://huggingface.co/Qwen/Qwen3-4B

Table 2: **Relative epoch analysis.** Cost denotes total training budget; Path denotes the effective optimization path length of the returned model. Merge-only includes Linear Merge, TIES Merge, and DARE-TIES. MTM-Multi reports Qwen3-4B-Base/OLMo-7B.

Method family	Cost	Path
Expert SFT	1.0	1.0
Merge-only	1.0	1.0
MergeMix / MTM-Single	2.0	1.0
MTM-Multi	**0.70/0.75**	**0.70/0.75**

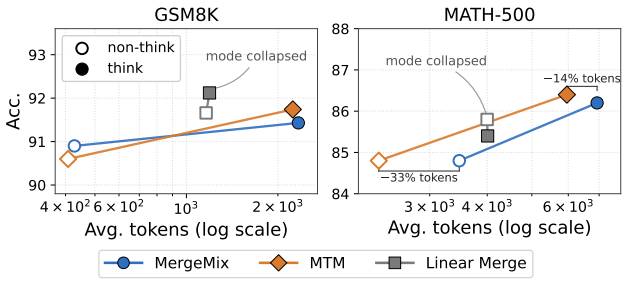


Figure 4: **Length-accuracy frontier under `non-think`/`think` modes on OLMo-7B.** Open/filled markers share a model; segment length = mode separation, Linear Merge collapses both modes.

that keeps math, code, and instruction-following performance simultaneously close to the best single-objective experts, rather than recovering one axis at the cost of another. MTM also consistently improves over MergeMix, which uses merge-based search only to select a single global data mixture. This indicates that the gains come from combining *branch-specific* mixtures with *deployment-time merging*, not from mixture selection alone. The gap is most pronounced on IF metrics, where MTM can retain an IF-leaning branch and recombine it with other branches at merge time. We further present the experiments on 5 and 7 domains using OLMo-3-7B and Qwen3-4B, as detailed in Section C.2.

Iterative refinement matters. MTM-Multi consistently outperforms MTM-Single, with the largest gains on interference-sensitive metrics such as IFEval. This supports the role of the iterative procedure in Section 4: each round realigns branches around the promoted shared reference model and adapts the next branch mixtures using specialist ratios.

Training cost. Table 2 compares the relative training budget and effective optimization path length of each method. MTM introduces additional cost by training mixture-biased branches, but these branches are independent within each round and can be trained in parallel. Moreover, MTM-Multi reuses the initial experts as the first stage of the iterative procedure rather than discarding them. As a result, its wall-clock cost remains comparable to a single-trajectory mixture training run with the same number of stages, while preserving the deployment-time flexibility of weight-space merging.

5.3 Hybrid Mode

We now turn to the harder hybrid setting in which the two branches encode *behaviors* rather than domains: a `no-think` branch trained to produce short, direct answers, and a `think` branch trained to produce long CoT reasoning. As shown in Section 3, naively merging these two branches collapses the two modes onto a single behavior, motivating mixture training and ultimately MTM as remedies. We probe this on OLMo-3-7B by comparing Linear Merge, Mixture Training, and MTM. Each method is evaluated on the same unified model under both `no-think` and `think` prompts, yielding two (token, accuracy) points per method (Figure 4). The horizontal extent between the two points measures how strongly the modes diverge in generation length, while their vertical positions reflect accuracy: a short segment means the two modes have collapsed into near-identical behavior, whereas a longer segment positioned further to the upper-left means modes remain distinct while overall using fewer tokens at comparable accuracy.

MTM preserves mode separation and shifts the frontier. Reading Figure 4 under this convention, Linear Merge (gray) produces a near-degenerate segment, indicating that the two modes have collapsed into a single behavior. Mixture Training (blue) yields a clearly extended segment, recovering the separation between short `no-think` and long `think` generations. MTM (orange) preserves this separation and additionally shifts the segment toward the upper-left, using fewer tokens at `no-think` while matching or exceeding Mixture Training accuracy at `think`. Together, these results confirm that mode separation must be built in through mixture training rather than merging hard experts,

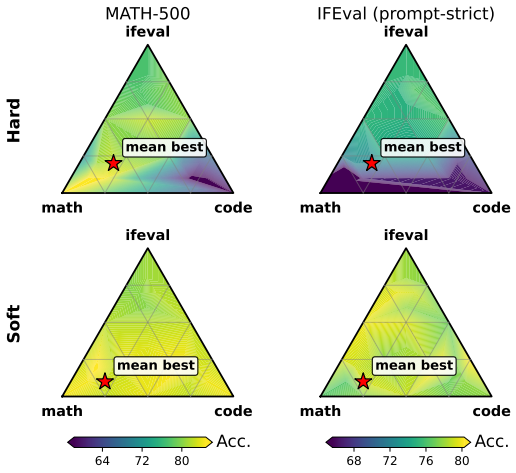


Figure 5: **Merge-simplex accuracy on Qwen3-4B-Base.** Ternary heatmaps over $\alpha \in \Delta^2$ for **Hard** vs.**Soft** branches on MATH-500 and IFEval. **Soft** branches yield a flatter, higher-valued interior.

Table 3: **Recipe transfer across architectures.** MTM recipes (branch mixtures $\{p_k\}$ and merge coefficients $\alpha_\star$) selected on Qwen3-4B-Base are reused on Qwen2.5-0.5B without re-running the search. For a controlled comparison, all methods are trained for the same fixed budget of 3 epochs. Merge (Uniform Merge) and Mixture (Uniform Mixture) are size-matched baselines that ignore the source recipe.

Method	MATH-500	MBPP+	IFEval	Avg.
Merge	37.00	27.78	37.65	34.14
Mixture	32.40	24.07	43.63	33.37
MTM	32.00	34.66	45.93	**37.53**

and that MTM further exploits branch-level mixture control to push the hybrid model onto a more favorable length/accuracy trade-off than mixture training alone can reach.

6 Additional Experiments

Merge-simplex landscape. To visualize how mixture training reshapes the weight-space geometry exploited by MTM, we evaluate $\mathcal{M}^{(t)}(\alpha)$ on a dense grid of $\alpha \in \Delta^2$ for the three-branch (math, code, IF) setting on Qwen3-4B-Base, comparing **Hard** branches (single-domain SFT) against **Soft** branches (mixture-trained). Figure 5 shows that the Hard branches concentrate accuracy near the corresponding vertex and decay sharply along interior rays, consistent with independently drifted experts lying outside any shared high-accuracy basin. Soft branches yield a flatter, higher-valued interior, so the selectors $\alpha_\star^{(t)}$ in Eq. (5) and $\alpha_k^{(t)}$ in Eq. (6) admit non-degenerate interior solutions that satisfy the cross-objective constraints $\tau_j^{(t)}$. This is direct geometric evidence that mergeability is a property of the branches, not the merger.

Cross-architecture recipe transfer. We next ask whether the recipe found by MTM is tied to the specific source model, or whether it captures a more general structure of the multi-objective search space. To test this, we take the branch mixtures $\{p_k\}$ and merge coefficients $\alpha_\star$ selected on Qwen3-4B-Base and apply them directly to Qwen2.5-0.5B, without any additional search or tuning. As shown in Table 3, MTM Transfer achieves a higher average score than both size-matched baselines. This suggests that the learned recipe can serve as a strong starting point for other architectures and model scales.

7 Conclusion

We study multi-objective language models built through weight-space merging. Merging single-objective experts often produces sharp cross-domain trade-offs or mode collapse in `think`/`non-think` hybrids. We attribute these failures to weight-space incompatibility: independently trained experts drift away from a shared high-performing basin. We propose **Mixture-Trained Merging** (MTM), which trains each branch on an objective-biased mixture so that task vectors remain compatible. MTM uses merged-model evaluations as a proxy for data-mixture training and iterates between MOBO based mixture search, branch specialization, and merging. Across Qwen3-4B-Base and OLMo-7B, MTM-Multi achieves the best average performance, keeps all objectives close to their single-domain experts, and preserves `think`/`non-think` separation. The flatter, higher-valued sim-

plex induced by mixture-trained branches supports our main conclusion: *mergeability is determined by the branches, not only by the merging rule*. Our study is limited to SFT-stage merging; extending MTM to larger objective sets and post-RL training remains an important direction for future work.

Acknowledgments and Disclosure of Funding

This work was supported by Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No.RS-2019-II190075, Artificial Intelligence Graduate School Program(KAIST)). This work was supported by Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No.RS-2022-II220713, Meta-learning Applicable to Real-world Problems). This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) with a grant funded by the Ministry of Science and ICT (MSIT) of the Republic of Korea in connection with the Global AI Frontier Lab International Collaborative Research Agreement. (No. RS-2024-00469482 & RS-2024-00509279). This work was conducted in collaboration with and supported by Kakao Corp.

References

[1] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[2] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[3] Boxin Wang, Chankyu Lee, Nayeon Lee, Sheng-Chieh Lin, Wenliang Dai, Yang Chen, Yangyi Chen, Zhuolin Yang, Zihan Liu, Mohammad Shoeybi, et al. Nemotron-cascade: Scaling cascaded reinforcement learning for general-purpose reasoning models. *arXiv preprint arXiv:2512.13607*, 2025.

[4] Qi Yang, Bolin Ni, Shiming Xiang, Han Hu, Houwen Peng, and Jie Jiang. R-4b: Incentivizing general-purpose auto-thinking capability in mllms via bi-mode annealing and reinforce learning. *arXiv preprint arXiv:2508.21113*, 2025.

[5] Zhilin Wang, Yafu Li, Xiaoye Qu, and Yu Cheng. See: Continual fine-tuning with sequential ensemble of experts. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 7418–7432, 2025.

[6] Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*, 2025.

[7] Woosung Koh, Jeyoung Jeon, Youngjin Song, Yujin Cheon, Soowon Oh, Jaehyeong Choi, and Se-Young Yun. msft: Addressing dataset mixtures overfiting heterogeneously in multi-task sft. *arXiv preprint arXiv:2603.21606*, 2026.

[8] Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A Smith, and Luke Zettlemoyer. Branch-train-merge: Embarrassingly parallel training of expert language models. *arXiv preprint arXiv:2208.03306*, 2022.

[9] Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[10] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.

[11] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in neural information processing systems*, 31, 2018.

[12] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

[13] Jiapeng Wang, Changxin Tian, Kunlong Chen, Ziqi Liu, Jiaxin Mao, Wayne Xin Zhao, Zhiqiang Zhang, and Jun Zhou. Mergemix: Optimizing mid-training data mixtures via learnable model merging. *arXiv preprint arXiv:2601.17858*, 2026.

[14] Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[15] Peter I Frazier. Bayesian optimization. In *Recent advances in optimization and modeling of contemporary problems*, pages 255–278. Informs, 2018.

[16] Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

[17] Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *Advances in Neural Information Processing Systems*, 36:69798–69818, 2023.

[18] Qian Liu, Xiaosen Zheng, Niklas Muennighoff, Guangtao Zeng, Longxu Dou, Tianyu Pang, Jing Jiang, and Min Lin. Regmix: Data mixture as regression for language model pre-training. *arXiv preprint arXiv:2407.01492*, 2024.

[19] Jiasheng Ye, Peiju Liu, Tianxiang Sun, Jun Zhan, Yunhua Zhou, and Xipeng Qiu. Data mixing laws: Optimizing data mixtures by predicting language modeling performance. *arXiv preprint arXiv:2403.16952*, 2024.

[20] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2022.

[21] Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.

[22] Fanqi Wan, Longguang Zhong, Ziyi Yang, Ruijun Chen, and Xiaojun Quan. Fusechat: Knowledge fusion of chat models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 21629–21653, 2025.

[23] Zhehao Huang, Yuhang Liu, Baijiong Lin, Yixin Lou, Zhengbao He, Hanling Tian, Tao Li, and Xiaolin Huang. Rain-merging: A gradient-free method to enhance instruction following in large reasoning models with preserved thinking format. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026.

[24] Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. Parallel bayesian optimization of multiple noisy objectives with expected hypervolume improvement. *Advances in neural information processing systems*, 34:2187–2200, 2021.

[25] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

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

[27] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[28] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

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

A Experimental Details

A.1 Detail for Qwen3-4B-Base

We fine-tune Qwen3-4B-Base[8] on trajectories generated by Qwen3-4B-Instruct-2507[9] in NOTHINK mode. Prompts are drawn from Dolci-Instruct-SFT[10] across three domains: math, coding, and precise-IF. We construct a separate corpus for each domain. Trajectories are generated using vLLM[11] with nucleus sampling (T=0.6, top-p=0.95, top-k=20) and a maximum of 8,192 new tokens. Samples exceeding 15,000 total tokens are filtered out, yielding 10,000 examples per training run. We use the same chat template for both training and inference.

For mixture-data SFT (used in the BO branch retrains; §A.3), we sample domain proportions $(\lambda_{\text{math}}, \lambda_{\text{code}}, \lambda_{\text{if}}) \in \Delta^2$ from the per-domain corpora.

Training is performed on 4× H200 GPUs using DeepSpeed ZeRO-2, bf16 weights, and TF32 matmul. The loss is computed only over assistant tokens.

Instead of a fixed epoch budget, we apply plateau-based early stopping. Training stops when the EMA (β=0.9) of validation token-level accuracy (512 samples, evaluated every 20 steps) fails to improve by τ=0.02 for three consecutive evaluations after a 100-step warm-up.

For BO branch retraining, we additionally impose a maximum step budget with geometric decay: $T_c = T_1 r^{c-1}$, where r=0.5.

Table 4: SFT hyperparameters for Qwen3-4B-Base.

Optimizer / LR	AdamW (fused), 2×10^{-5}, constant w/ 0.03 warmup
Weight decay / clip	0.01 / 1.0
Effective batch size	128 ($2 \times 16 \times 4$ GPUs)
Max sequence length	10,000
Precision	bf16 weights, tf32 matmul, grad ckpt
Distributed	DeepSpeed ZeRO-2
Plateau stop (τ, K, β)	0.02, 3, 0.9 (val 512, every 20 steps)
Seed	42

A.2 Detail for OLMo-7B

We fine-tune OLMo-3-1025-7B[12] under two settings: (1) multi-domain and (2) hybrid-mode (think / nothink).

In the multi-domain setting, we generate math, coding, and precise-IF trajectories using Olmo-3-7B-Instruct[13] with prompts from Dolci-Instruct-SFT. Each domain forms a separate corpus.

In the hybrid-mode setting, we additionally include THINK trajectories generated by Olmo-3-7B-Think[14] using prompts from Dolci-Think-SFT-32B[15]. These trajectories prepend a ⟨`think`⟩ . . . ⟨/`think`⟩ reasoning block before the final answer, while NOTHINK trajectories directly output the answer.

The sampling configuration matches that of the Qwen setup.

Training is conducted on 4× H200 GPUs with DeepSpeed ZeRO-2, bf16 weights, and TF32 matmul. The loss is computed over assistant tokens only, including the reasoning block in THINK mode.

Unlike Qwen, we use a fixed epoch budget without early stopping. Hyperparameters are summarized in Table 5.

[8] https://huggingface.co/Qwen/Qwen3-4B-Base
[9] https://huggingface.co/Qwen/Qwen3-4B-Instruct-2507
[10] https://huggingface.co/datasets/allenai/Dolci-Instruct-SFT
[11] https://github.com/vllm-project/vllm
[12] https://huggingface.co/allenai/Olmo-3-1025-7B
[13] https://huggingface.co/allenai/Olmo-3-7B-Instruct
[14] https://huggingface.co/allenai/Olmo-3-7B-Think
[15] https://huggingface.co/datasets/allenai/Dolci-Think-SFT-32B

Table 5: SFT hyperparameters for OLMo-7B. Identical across THINK and NOTHINK runs; only the input corpus and trajectory length cap differ (16,384 vs. 8,192 new tokens).

Optimizer / LR	AdamW (fused), 2×10^{-5}, constant w/ 0.03 warmup
Weight decay / clip	0.01 / 1.0
Effective batch size	64 ($1 \times 16 \times 4$ GPUs)
Max sequence length	10,000
Precision	bf16 weights, tf32 matmul, grad ckpt
Distributed	DeepSpeed ZeRO-2
Early stopping	disabled (fixed epoch budget)
Seed	42

A.3 Bayesian Optimization Details

We use Bayesian optimization to search over merge coefficients on the simplex. At each MTM iteration, the initial design includes the simplex vertices, a central or previously best merge point, and a small number of space-filling random candidates. Each candidate corresponds to a merged checkpoint, which is evaluated on all objectives.

Each objective is modeled with an independent Gaussian process surrogate using a Matérn-5/2 kernel with automatic relevance determination and outcome standardization. The surrogates are fit by maximizing the marginal likelihood and refit after each new observation. We use qNEHVI as the acquisition function, optimized sequentially with batch size $q = 1$. The reference point is defined relative to the single-objective expert scores, encouraging hypervolume improvement only for candidates that approach the expert frontier while maintaining balanced performance across objectives.

After exhausting the BO budget, we select a global merge coefficient from the observed candidates using a balanced multi-objective score that prioritizes the weakest objective and avoids near-vertex solutions. We also select specialist merge coefficients for the next MTM iteration by requiring the target objective to remain close to its expert score while constraining degradation on the other objectives. These coefficients are then used for base promotion and mixture propagation in the next iteration.

A.4 Evaluation Settings

Table 6: Evaluation settings for each benchmark. n denotes the number of generations per prompt. All benchmarks use one greedy generation and three sampled generations with T=0.7 and top-p=0.95 across different random seeds.

Benchmark	# Samples	n	Metric
IFEval[1]	541	4	prompt/inst. acc. (strict, avg.)
MBPP+[2]	378	4	pass@1 (avg.)
MATH-500[3]	500	4	pass@1 (avg.)
GSM8K[4]	1,319	4	exact match (avg.)
LiveCodeBench[5]	268 (v5, 2024-08 – 2025-02)	4	pass@1 (avg.)

[1] https://huggingface.co/datasets/google/IFEval
[2] https://github.com/evalplus/evalplus
[3] https://huggingface.co/datasets/HuggingFaceH4/MATH-500
[4] https://huggingface.co/datasets/openai/gsm8k (test split)
[5] https://livecodebench.github.io/

We evaluate all models using the NeMo-Skills framework[16], serving checkpoints with vLLM[17] on 16× NVIDIA H200 80GB GPUs. We use tensor parallelism TP $\in \{2, 4\}$ and bfloat16 weights. The maximum generation length is set between 8,196 and 30,000 tokens depending on the benchmark. Generations are scored using the official evaluators bundled with NeMo-Skills. For each prompt, we generate four responses: one greedy response with T=0 and three sampled responses using T=0.7,

[16] https://github.com/NVIDIA/NeMo-Skills
[17] https://github.com/vllm-project/vllm

Algorithm 1 Iterative Mixture–Merge with Bayesian Optimization

Require: Base model M_0; corpora $\{D_k\}_{k=1}^{K}$; iterations T; BO budget B; thresholds $\{\tau_j^{(t)}\}$

Initialize $p_k^{(1)} \leftarrow e_k$ for all k
for $t = 1, \ldots, T$ **do**
  Train experts: $E_k^{(t)} \leftarrow \textsc{FineTune}\Big(M_{t-1}, \textsc{Mix}(D_{1:K}; p_k^{(t)})\Big)$ for all k
  Initialize candidate set $\mathcal{X}_t$ with vertices $\{e_k\}$, the uniform weight, and random Dirichlet samples
  for $b = 1, \ldots, B$ **do**
    $\alpha_b \leftarrow \text{BO-}\textsc{Propose}(\mathcal{X}_t)$
    Evaluate $f_k^{(t)}(\alpha_b) \leftarrow \textsc{Eval}_k\Big(\textsc{Merge}(E_{1:K}^{(t)}; \alpha_b)\Big)$ for all k
    $\mathcal{X}_t \leftarrow \mathcal{X}_t \cup \{\alpha_b\}$
  end for
  Select global ratio $r_\star^{(t)} \leftarrow \arg\max_{\alpha \in \mathcal{X}_t} \frac{1}{K} \sum_{j=1}^{K} f_j^{(t)}(\alpha)$
  Select each specialist ratio $r_k^{(t)}$ by constrained target-domain performance
  Promote base: $M_t \leftarrow \textsc{Merge}(E_{1:K}^{(t)}; r_\star^{(t)})$
  Update mixtures: $p_k^{(t+1)} \leftarrow \sum_{j=1}^{K} r_{k,j}^{(t)} p_j^{(t)}$ for all k
end for
return M_T

top-p=0.95, and different random seeds. We report the average score across these four generations. Dataset sizes and metrics are summarized in Table 6.

B Algorithm

Algorithm 1 provides the full MTM iteration, including branch training, BO-based merge-ratio search, base promotion, and mixture updates.

C Multi-Objective Benchmark Results

C.1 Evaluation Settings

We evaluate the domain sets below. Experts, benchmarks, and baseline tuning differ across settings, so these experiments do not isolate the effect of increasing the number of domains.

Setting	Domains	Benchmarks
Qwen / OLMo, three	Math, Code, IF	GSM8K, MATH-500, MBPP+, LiveCodeBench, IFEval
OLMo, five	IF, Law, Medical, Finance, Safety	IFEval, CaseHOLD, MedQA, FinQA, BeaverTails
Qwen, five	Math, Code, IF, Medical, Safety	MATH-500, MBPP+, IFEval, MedMCQA, TensorTrust
Qwen, seven	Qwen five plus Law and Finance	Qwen five plus CaseHOLD and FPB

Safety covers BeaverTails response safety on OLMo and TensorTrust prompt-injection robustness on Qwen; Finance covers FinQA question answering and Financial PhraseBank (FPB) sentiment analysis, respectively. The additional Qwen datasets are `openlifescienceai/medmcqa`, `qxcv/tensor-trust`, `coastalcph/lex_glue` (CaseHOLD), and `takala/financial_phrasebank`. OLMo coefficient search uses 20% of the validation set. Expanded-domain results are point estimates; Avg. is the arithmetic mean of benchmark scores.

C.2 Domain Extention Experiment Results

In Table 7, we evaluate five more heterogeneous domains on OLMo-3-7B. MTM achieves a five-domain average of 63.2, substantially outperforming uniform and BO-searched merging. And we also evaluate five and seven more domains on Qwen3-4B. The two panels of Table 8 show average scores of 73.5 and 72.5 for MTM-Multi, compared with 70.3 and 70.1 for TIES. Gains are strongest in IF and Safety, while TIES remains better on some benchmarks, including Math. In the seven-domain setting, MTM-Multi is 0.4 points below the expert oracle average. Overall, MTM's advantage grows as domains become more heterogeneous and specialized.

Table 7: **Five-objective results on OLMo-3-7B.** The oracle uses the corresponding expert for each domain, with Avg. computed over those five scores. Bold marks the best unified-model score in each column.

Method	IF IFEval	Law CaseHOLD	Medical MedQA	Finance FinQA	Safety BeaverTails	Avg.
Expert oracle	65.0	73.6	54.6	32.3	91.0	63.3
Uniform Merge	53.1	65.3	**52.8**	26.8	57.9	51.2
BO Merge	62.4	67.2	51.1	27.7	73.6	56.4
MTM-Multi	**67.3**	**70.3**	52.6	**31.4**	**94.4**	**63.2**

Table 8: **Five- and seven-domain results on Qwen3-4B.** Panels use separate expert sets. The oracle selects a domain expert for each benchmark; bold marks the best unified-model score.

(a) Five domains: Math, Code, IF, Medical, Safety

Method	Math MATH-500	Code MBPP+	IF IFEval	Medical MedMCQA	Safety TensorTrust	Avg.
Expert oracle	85.2	68.3	80.8	59.6	76.6	74.1
TIES Merge	**85.4**	68.3	72.9	**59.8**	65.3	70.3
MTM-Multi	80.6	**68.6**	**80.2**	**59.8**	**78.5**	**73.5**

(b) Seven domains: the five-domain set plus Law and Finance

Method	Math MATH-500	Code MBPP+	IF IFEval	Medical MedMCQA	Safety TensorTrust	Law CaseHOLD	Finance FPB	Avg.
Expert oracle	84.6	68.3	80.1	60.0	76.5	67.1	74.0	72.9
TIES Merge	**81.8**	**69.3**	70.3	**58.8**	68.6	63.0	**79.0**	70.1
MTM-Multi	81.0	69.0	**81.0**	57.6	**77.7**	**64.1**	77.0	**72.5**

C.3 Expert-Relative Retention

For benchmark score s_j and expert score s_j^{expert}, we report

$$R_j = 100\,\frac{s_j}{s_j^{\text{expert}}}, \qquad R_{\text{mean}} = \frac{1}{J}\sum_{j=1}^{J} R_j, \qquad R_{\min} = \min_{j\in[J]} R_j. \tag{9}$$

The oracle combines benchmark-wise expert scores, not a single unified model; retention may exceed 100%. In Table 9, MTM-Multi retains 99.3% on average and at least 95.5% on each benchmark, versus 96.3% and 87.8% for TIES. The seven-domain TIES baseline uses searched coefficients $\lambda = (0.7, 1.2, 1.2, 1.0, 1.0, 1.0, 1.0)$ in the listed domain order.

Table 9: **Expert-relative retention (%) in the seven-objective setting.** Avg. and Min. summarize the seven benchmark-level ratios; values above 100% indicate performance above the corresponding expert.

Method	Math MATH	Code MBPP+	IF IFEval	Medical MedMCQA	Safety TensorTrust	Law CaseHOLD	Finance FPB	Avg.	Min.
Expert oracle	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
TIES Merge	96.7	101.5	87.8	98.0	89.7	93.9	106.8	96.3	87.8
MTM-Multi	95.7	101.0	101.1	96.0	101.6	95.5	104.1	**99.3**	**95.5**

In the three-domain setting, reported worst-domain retention is 98.53% for Qwen MTM-Multi versus 97.18% for TIES, and 92.49% for OLMo MTM-Multi versus 89.17% for MergeMix. These are reported domain-level summaries for a setting with multiple Math and Code benchmarks, distinct from the benchmark-level ratios above.

D Continual Extension to New Domains

We start from an OLMo-3-7B MTM model covering IF, Medical, and Finance, then add Law and Safety. We compare two extension strategies: **Naive merge** independently trains Law and Safety experts and merges them with the existing model using the best BO-selected coefficients; **Continual MTM** continues the MTM procedure over the two new domains, training mixture-based branches instead of adding independent experts. Both start from the same three-domain MTM model, but only the latter uses MTM for the extension. All five benchmarks use greedy evaluation.

Table 10: **Extending a three-domain OLMo MTM model with Law and Safety.** Naive merge uses independently trained new-domain experts with BO-selected merge coefficients; Continual MTM trains mixture-based branches for the extension. One-shot MTM has all five domains available from the outset.

Method	Old objectives			New objectives		Avg.
	IF IFEval	Medical MedQA	Finance FinQA	Law CaseHOLD	Safety BeaverTails	
Before extension (3-domain MTM)	71.9	54.5	33.9	55.4	49.6	53.1
Naive merge (independent experts + BO)	66.5	48.7	20.8	64.5	88.7	57.8
Continual MTM (mixture-trained branches)	71.9	50.9	31.1	71.2	92.0	**63.4**
One-shot 5-domain MTM (reference)	67.3	52.6	31.4	70.3	94.4	63.2

Preserving old domains. Naive merge improves Law and Safety but reduces IF from 71.9 to 66.5, Medical from 54.5 to 48.7, and Finance from 33.9 to 20.8. Continual MTM preserves IF at 71.9, retains more Medical and Finance performance, and also scores higher on Law and Safety. Its average is 63.4, compared with 57.8 for Naive merge and 63.2 for one-shot MTM. Some forgetting remains on Medical and Finance.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: Abstract and introduction contains our main contributions and scope.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss the limitations of our work in conclusion section.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

Justification: Our paper does not include theoretical results.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We describe the experimental details in Section 5 and Section A

   Guidelines:

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

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

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

Answer: [No]

Justification: The supplemental material provides implementation details, evaluation settings, and benchmark configurations needed to understand and reproduce the experimental setup. Our experiments build on open-source frameworks, including NeMo-Skills for evaluation and Hugging Face Trainer for fine-tuning. However, we do not release the full codebase and experiment scripts at submission time to preserve anonymity. Upon acceptance, we plan to release the code, evaluation scripts, configuration files, and detailed reproduction instructions, including data access and preprocessing details for the public benchmarks used in the paper.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We provide the experimental details including data, hyperparameters, optimizer, in Section 5 and Section A

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We run the experiments using three different random seeds to compute the standard deviation of the main results reported in Table **??**.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We provide our compute resources in Section A.4

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: Our researach satisfies the NeurIPS Code of Ethics

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [No]

    Justification: This work is primarily a methodological study of multi-objective post-training and weight-space merging for language models. It does not introduce a new application, dataset, deployment setting, or user-facing system. As such, the paper does not include a dedicated broader impact discussion. Potential societal impacts would arise mainly through downstream uses of the resulting language models, rather than from the proposed merging procedure itself.

Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: Our work focuses on combining existing safe open-source models as a methodology and does not involve releasing any new models. Therefore, it poses minimal risk of misuse.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We properly cited the datasets used in Section 5.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: We provide asset, including descriptions of the experiments in Section 5

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: Our research does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: Our research does not involve research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: We only use LLMs for editing the paper.

    Guidelines:

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