SELECTIVE MODULAR MERGING OF HETEROGENEOUS LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Model merging offers a training-free strategy for model adaptation and knowledge transfer, with a significant efficiency advantage over existing post-training methods (such as reinforcement learning-based distillation and supervised fine-tuning). However, current merging methods either apply to homogeneous models only or require additional data and training costs, limiting their feasibility and practical usefulness. In this study, we propose **Selective Modular Merging (SMM)**, a simple yet effective training-free method for merging heterogeneous language models. In principle, given a task-specific student model and its corresponding task vector, SMM further improves performance on the task by selectively merging a heterogeneous but more powerful source model into the target module by module. For each of the target's FFN layers, SMM aligns the source's grouped FFN layers to it via orthogonal Procrustes, then selects the most informative one to merge in an optimization framework based on the task vector. For the target's attention layers, SMM retains them because of their contextual dependency, a choice supported by the same framework. We evaluate SMM on merging heterogeneous Llama and Qwen models of different sizes. Experiments across language and mathematical reasoning tasks show that SMM helps target models approach source performance while keeping their small model sizes, demonstrating that independently trained heterogeneous language models can serve as reusable carriers for training-free knowledge transfer and enhancement. Our code is available at https://anonymous.4open.science/r/SMM-code-97BC

1 INTRODUCTION

Model merging (Wortsman et al., 2022; Ilharco et al., 2023; Yang et al., 2026) has emerged as a promising paradigm that combines the parameters of multiple models into a single model without additional training. By directly manipulating model weights, model merging enables practitioners to integrate the capabilities of different models, such as multi-task abilities (Yadav et al., 2023) and domain-specific expertise (Yu et al., 2024), thereby reusing the rapidly growing number of publicly available checkpoints (Goddard et al., 2024). Compared with conventional knowledge transfer approaches such as knowledge distillation (Ko et al., 2024; Fang et al., 2026), fine-tuning (Hu et al., 2022), and ensemble learning (Ganaie et al., 2022), model merging operates directly in weight space, allowing it to integrate knowledge into a single model at low computational cost and without access to the original training data.

However, most existing model merging methods assume homogeneous models (Yang et al., 2026), which severely limits their practical applicability, as language models (Grattafiori et al., 2024; Qwen Team, 2025) usually differ in architecture. Heterogeneous model merging lifts this restriction by allowing models that differ in architecture, size, and training objective to be combined, thereby enabling knowledge transfer across model families and scales. However, the main difficulty in merging heterogeneous models is that their architectural and parameter mismatches render simple weight combination either infeasible or detrimental to performance, especially in the absence of a reliable way to align their representational spaces. To address this, existing methods align heterogeneous models via intermediate representations (Luenam et al., 2026), activation-based transport (Cui et al., 2026), or learned latent weight spaces (Soro et al., 2026). Although these approaches make heterogeneous merging possible, they require model inputs, forward activations, or additional datasets and learned mappings, leaving the stricter case that only model weights are available open.

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

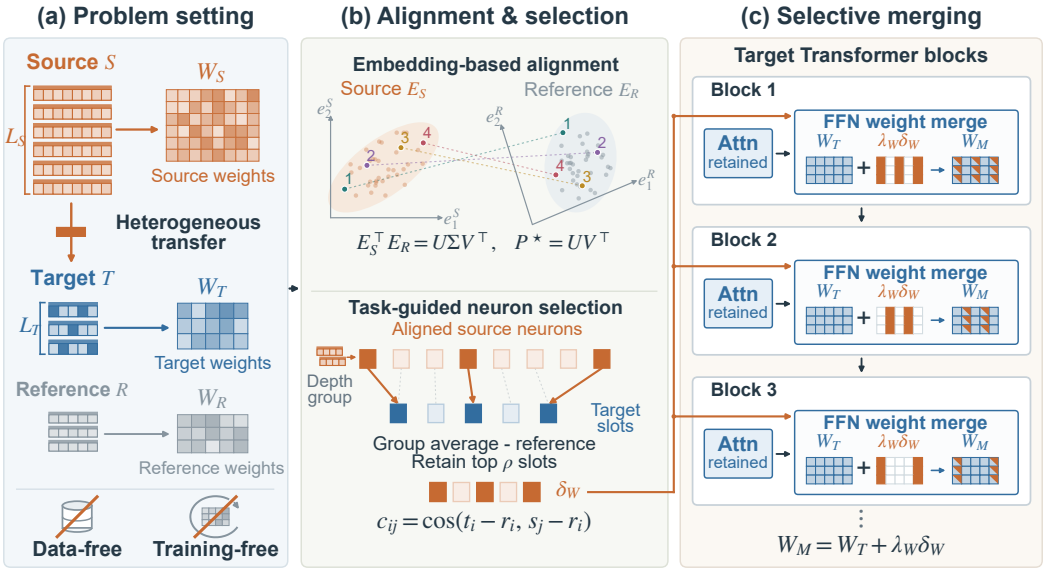

Figure 1: **Overview of Selective Modular Merging (SMM).** (a) Source S, target T, and reference R supply the checkpoint weights; merging is data-free and training-free for a fixed configuration. (b) Shared-token embeddings define a Procrustes map. Aligned source FFN units are matched using the target–reference task vector, aggregated within relative-depth groups, and selected under coverage ρ to form sparse updates δ_W; s_j denotes an aligned source unit. (c) SMM updates selected target FFN weights as $W_M = W_T + \lambda_W \delta_W$ with a per-matrix norm bound, while retaining attention and the target architecture. Orange marks selected updates, which are applied before inference. Grids and point clouds are schematic; normalization and residual connections are omitted.

To address these limitations, we propose **Selective Modular Merging (SMM)**, a training-free framework that transfers capabilities from a larger source model into a smaller heterogeneous target model using only model checkpoints. Instead of directly interpolating incompatible parameters, SMM aligns the hidden spaces of the two models through the embeddings of their shared tokens, and treats feed-forward network (FFN) neurons as the basic units of transfer. It then transports source neurons into the target parameter space, selects those whose changes agree with the target's task vector, and leaves the attention layers of the target unchanged to avoid context-dependent interference. Figure 1 summarizes the SMM workflow.

We evaluate SMM on heterogeneous Llama and Qwen model pairs at multiple scales, covering language understanding, mathematical reasoning, and domain-specific tasks. These transfers use only checkpoint weights and retain the target architecture, requiring neither additional training nor inference-time expansion. Our work offers a practical route to reusing larger pretrained models to extend smaller specialized models, potentially lowering the cost of capability adaptation and making heterogeneous model reuse more accessible to the community.

Our contributions are summarized as follows:

- **Training-free heterogeneous merging.** We propose SMM, which merges a larger source model into a smaller heterogeneous target without training, gradient updates, or forward passes on calibration data, and keeps the target architecture unchanged.
- **Selective modular design.** SMM aligns the two models by orthogonal Procrustes on shared token embeddings, merges FFN neurons selected by the target's task vector, and retains the target attention layers, a choice supported by a Jacobian analysis and module ablations.
- **Empirical effectiveness.** Across Qwen and Llama transfers in five domains, SMM improves the target on most benchmarks and is competitive with methods that require calibration data or additional training.

2 Related Work

Model Merging. Model merging combines existing checkpoints to integrate their capabilities into a single model. Model Soups (Wortsman et al., 2022) averages fine-tuned models sharing an initialization, while Task Arithmetic (Ilharco et al., 2023) represents adaptation as a parameter displacement from a reference checkpoint. To mitigate interference between task updates, TIES-Merging (Yadav et al., 2023) and Consensus Merging (Wang et al., 2024) filter conflicting updates, whereas AdaMerging (Yang et al., 2024) optimizes task- or layer-wise merging coefficients using unlabeled examples. Subspace-based approaches, including KnOTS (Stoica et al., 2025), TSV-Merge (Gargiulo et al., 2025), and WUDI-Merging (Cheng et al., 2025), exploit the geometry of task updates to improve alignment and reduce interference. These approaches primarily operate on shape-compatible parameters or task updates, rather than establishing parameter correspondence between models with different depths and widths.

Heterogeneous Model Merging. Heterogeneous merging additionally requires reconciling incompatible parameter shapes and identifying cross-model correspondences. Optimal-transport merging (Singh & Jaggi, 2020) establishes soft neuron correspondences between networks with matching layer structures but different widths, including a weight-only formulation. For heterogeneous language models, Transport and Merge (Cui et al., 2026) uses calibration activations to guide alignment and parameter transport. Other approaches accommodate heterogeneity through output supervision or learned weight representations: FuseLLM (Wan et al., 2024) combines source-model predictions and continually trains a target on the resulting distributions, while LS-Merge (Soro et al., 2026) learns a latent weight space for dimensional alignment and interpolation. These approaches enable heterogeneous knowledge transfer but introduce calibration or additional learning requirements. SMM combines checkpoint-based alignment with target-guided selection: shared-token embeddings establish a cross-model coordinate map, and the target's displacement from a compatible reference checkpoint guides the selection of source FFN units. For a fixed configuration, the merge requires neither calibration examples nor gradient updates, while preserving the target architecture and attention parameters.

3 Proposed Method

3.1 Problem Setting

We consider merging a source model S into a target model T, using a reference checkpoint R with the same architecture and compatible parameter coordinates as T. In contrast, the source may differ from the target in depth, hidden width, and FFN width. Given only these checkpoints, our goal is to construct a merged model that preserves the target architecture under a fixed merging configuration, without requiring additional data or training. Unlike conventional multi-task merging, which combines architecture-compatible models fine-tuned from a common checkpoint, the source and target here need not share parameter shapes or neuron indices, and their hidden spaces are generally misaligned due to independent training. Moreover, as the target is the model to be deployed, transferring source knowledge must resolve these mismatches without enlarging the target or overwriting its adaptation.

Following task arithmetic (Ilharco et al., 2023), we use the target–reference displacement as the task vector

$$\tau = \theta_T - \theta_R, \tag{1}$$

where θ_T and θ_R denote the parameters of the target and reference models, respectively. The task vector characterizes the target relative to R and provides a signal for selecting which source knowledge to merge. Its semantics depend on the reference.

Building on this formulation, we propose a selective modular merging principle, which treats the token embeddings, FFN layers, and attention layers according to their distinct roles and determines not only how but also whether each module should be merged. Specifically, the embeddings of shared tokens serve as anchors to establish the correspondence between the source and target (Section 3.2); the coupled FFN units of the source provide candidate updates for the target (Section 3.3); and the target attention layers are retained to preserve their learned token interactions (Section 3.4). Throughout the merging process, the task vector guides the selection of source updates, while an update budget bounds the perturbation to the target.

3.2 Embedding Alignment for Cross-Model Transfer

As discussed in Section 3.1, the source and target differ in hidden width and are misaligned in their hidden spaces, yet aligning them is difficult without data. However, Lee et al. (2025) find that models from the same family share the relative geometry of their token embeddings, i.e., which tokens are closer together or farther apart, even when their embedding dimensions differ, and that embedding-based maps can transfer representations across model dimensions. Motivated by this, we align the embedding spaces of the source and target to express source parameters in target coordinates, enabling task-vector operations as in homogeneous merging. Since tokenizer compatibility is an important obstacle to such linear alignment (Gorbett & Jana, 2026), we focus on heterogeneous models within the same family that share a tokenizer.

Vocabulary correspondence and hidden-width alignment are two separate requirements, and we first address the former. A shared tokenizer provides direct token correspondences: embedding rows are paired by token identity, so that each token present in both vocabularies serves as an anchor representing the same token in both models. Extra embedding rows that exist in only one model are excluded. The target tokenizer, embeddings, and output head are retained, which preserves the input and output interface of the target, while the shared vocabulary is used only to align the internal FFN parameters.

For hidden-width alignment, let $E_S \in \mathbb{R}^{n \times d_S}$ and $E_R \in \mathbb{R}^{n \times d_R}$ contain the row-normalized source and reference embeddings of the n anchors, where d_S and d_R denote the hidden dimensions of the source and target, respectively. Following the correlation formulation of orthogonal Procrustes (Schönemann, 1966; Smith et al., 2017), we seek a semi-orthogonal map from source to reference coordinates:

$$P^\star = \arg\max_{P^\top P = I} \operatorname{tr}(P^\top E_S^\top E_R) = UV^\top, \quad \text{where } E_S^\top E_R = U\Sigma V^\top. \tag{2}$$

Here $P^\star \in \mathbb{R}^{d_S \times d_R}$, and the thin SVD gives a map that maximizes agreement between paired anchors. The reference and target share residual coordinates through their common pretraining lineage. The rectangular correlation objective and its relation to least squares are derived in Appendix B.2.

This map specifies how a source module reads from and writes to the target residual space. For a source FFN F_S and a row-vector target state h, transforming its input-facing and output-facing weights gives

$$\widetilde{F}_S(h) = F_S(hP^{\star\top})P^\star. \tag{3}$$

That is, the gate and up projections read the target input lifted into source coordinates, while the down projection maps the result back to target coordinates (see Appendix B.3 for the corresponding parameter transformations). As a result, the transported FFN operates on the target residual stream while retaining all units of the source FFN, so that its units can be directly compared with those of the target; Section 3.3 then determines which of them are merged. We apply the same map to FFNs at all depths, which is a modeling assumption supported by the finding of Lee et al. (2025) that embedding-based maps can transfer steering vectors between intermediate layers of different models.

3.3 Task-Guided FFN Merging

With source FFNs expressed in target coordinates, it remains to determine which of their units to merge into the target. We take a complete gated FFN unit as the basic element of merging, which consists of a gate row, an up row, and the corresponding down column (Shazeer, 2020), and which acts as a modular carrier of capability, since these components jointly define a nonlinear content transformation (Geva et al., 2021). We therefore assign the three components of each unit jointly, so as not to combine components that do not belong to the same source transformation.

Unlike existing methods that match neurons by the similarity of weights or activations (Cui et al., 2026), we use the target's task vector to select these units. Let t_i, r_i, and $\tilde{s}_j$ denote the gate, up, and down components of target slot i, its reference counterpart, and transported source unit j, respectively. Subtracting the reference yields the task vector of the target at slot i, $\tau_i = t_i - r_i$, and the candidate source increment $\tilde{s}_j - r_i$, whose directions we compare using

$$c_{ij} = \cos(\tau_i,\ \tilde{s}_j - r_i)\,. \tag{4}$$

Embedding alignment supplies the coordinate correspondence, while the reference gives both changes a common origin, so that the matching is conditioned on the target's existing adaptation without accessing task examples or intermediate activations.

To handle the depth mismatch, we group source layers by relative depth, match each of them to the corresponding target layer via a greedy one-to-one assignment over positive scores, and average the matched units on slots matched by every source layer in the group. To limit the perturbation to the target, a coverage budget ρ retains only the highest-scoring slots, and a norm budget β caps each matrix update at a fraction β of the target weight norm. The resulting update δ_{FFN}, i.e., the averaged source units minus the reference on the retained slots, is added to the target (Appendices C.1–C.3).

An Optimization Perspective. For a twice continuously differentiable task loss, a second-order expansion of the loss change caused by the update gives

$$\mathcal{L}_{\mathrm{task}}(\theta_T+\delta_{\mathrm{FFN}})-\mathcal{L}_{\mathrm{task}}(\theta_T) = \nabla\mathcal{L}_{\mathrm{task}}(\theta_T)^\top\delta_{\mathrm{FFN}}+\tfrac{1}{2}\,\delta_{\mathrm{FFN}}^\top H_{\mathrm{task}}(\theta_T)\,\delta_{\mathrm{FFN}}+o\big(\|\delta_{\mathrm{FFN}}\|_2^2\big)\,, \quad (5)$$

where the first-order term depends on the direction of the update, while the curvature term, governed by the Hessian H_{task}, grows quadratically with its magnitude and can reinforce or offset the first-order effect. Accordingly, SMM uses the task vector to guide the update direction and the two budgets to control its magnitude, without identifying τ with the current negative gradient.

3.4 Why Retain Target Attention?

Unlike FFNs, which act on each position independently, an attention update changes which information the target routes between positions. Its effect therefore cannot be judged from aligned weights alone, because the same parameter change acts differently on different input contexts. Consider an update δ_{attn} to one target attention block. For a context x, let y_x be the block output, ℓ_x the task loss on x, and $J_A(x) = \partial y_x/\partial\theta_A$ the Jacobian of the block at the target weights, with the incoming hidden states fixed. Applying the first-order term of Equation (5) to this update gives

$$\Delta y_x \approx J_A(x)\,\delta_{\mathrm{attn}}, \qquad \Delta\ell_x \approx (\nabla_{y_x}\ell_x)^\top J_A(x)\,\delta_{\mathrm{attn}}, \quad (6)$$

where the downstream gradient $\nabla_{y_x}\ell_x$ includes the effect of later layers. For query/key changes, the Jacobian passes through the softmax Jacobian $\mathrm{Diag}(a_x) - a_x a_x^\top$, where a_x is the attention weight vector, while value and output-projection changes depend on the content being routed. Both the Jacobian and the downstream gradient therefore vary with the context. For example, consider two positions with equal attention weights whose output-projected values are v and $-v$. A query/key update that shifts attention toward the first position moves the block output toward v; exchanging the two values leaves both the original output and the attention-weight update unchanged, yet reverses the output change, and hence the sign of $\Delta\ell_x$ whenever the downstream gradient has a nonzero component along v.

Hence, neither embedding alignment nor the task vector τ determines the sign of the loss change in equation 6 across task examples, and SMM neither accesses such examples nor evaluates this Jacobian during merging. In this data-free setting, we retain the original target attention parameters for stable performance, and transfer only the selected FFN increments. This preference does not imply that every attention merge is harmful. Moreover, retaining attention parameters does not freeze attention behavior, since FFN changes still alter later attention distributions through the hidden states.

3.5 Merging Procedure and Computational Cost

Algorithm 1 summarizes the complete procedure. The embedding map $P^\star$ is computed once and reused across all layers, and matching operates on low-dimensional sketches of coupled FFN units. With n anchors, computing $P^\star$ costs $O(nd_Sd_T + d_Sd_T^2)$ for the cross-product and its thin SVD, while scoring one layer pair with FFN widths m_T and m_S and sketch length k costs $O(m_Tm_Sk)$; blockwise scoring avoids storing the complete score matrix (Appendix C.1).

For a fixed configuration, SMM therefore requires no calibration forward passes, activation storage, or gradient-based fitting, whereas existing heterogeneous merging methods rely on activations computed on calibration data (Cui et al., 2026) or on learned mappings (Soro et al., 2026). Moreover, the merged checkpoint keeps the target architecture and parameter count, and thus introduces no additional inference cost. Measured runtimes, which depend on checkpoint sizes, matching budgets, and hardware, are reported in Section 4.1.

Algorithm 1 Selective Modular Merging (SMM)

Require: Source S, target T, reference R; coverage budget ρ, norm budget β, small constant $\epsilon > 0$
Ensure: Merged model M with the target architecture

Initialize $M \leftarrow T$; pair and row-normalize shared-token anchors E_S, E_R.
Compute the thin SVD $E_S^\top E_R = U\Sigma V^\top$ and set $P^\star \leftarrow UV^\top$.
for each target FFN layer **do**
    Group source layers by relative depth.
    for each source layer in the group **do**
        Transport coupled gate–up–down units into target coordinates using $P^\star$.
        Score candidates using sketches: $c_{ij} \approx \cos(\tau_i, \tilde{s}_j - r_i)$.
        Process target slots by descending best score; accept positive, unused source units.
    end for
    Keep slots matched by every source layer in the group; rank them by mean score.
    Retain at most $\lceil \rho m_T \rceil$ slots, where m_T is the target FFN width.
    Average matched units, subtract the reference, and zero the remaining slots to form δ_W.
    for each FFN matrix W with $\delta_W \neq 0$ **do**
        $$W_M \leftarrow W_T + \min\left\{1, \frac{\beta \max(\|W_T\|_F, \epsilon)}{\|\delta_W\|_F}\right\} \delta_W.$$
    end for
end for
return M with parameters $\theta_T + \delta_{\text{FFN}}$; all non-FFN parameters are retained from the target.

4 Experiments

4.1 Experimental Setup

Models and Benchmarks. We evaluate SMM on four Qwen2.5 capability transfers and a separate Llama-based Malay adaptation setting. The Qwen experiments share Qwen2.5-1.5B-Instruct as the target and Qwen2.5-1.5B as the reference, with each transfer producing a separate target-sized checkpoint. For mathematical reasoning and code generation, we use Qwen2.5-Math-7B-Instruct and Qwen2.5-Coder-7B-Instruct as the respective sources, evaluating on MATH-500 (Hendrycks et al., 2021; Lightman et al., 2024) and GSM8K (Cobbe et al., 2021), and on MBPP+ and HumanEval+ (Liu et al., 2023). For instruction following and multilingual understanding, we use Qwen2.5-7B-Instruct as the source and evaluate on IFEval (Zhou et al., 2023), and on Belebele (Bandarkar et al., 2024) and XNLI (Conneau et al., 2018), respectively. General-capability retention (*General*) is measured by equal-task macro accuracy over ARC-Easy (test) (Clark et al., 2018), PIQA (validation) (Bisk et al., 2020), and WinoGrande (validation) (Sakaguchi et al., 2020). The Llama experiment transfers an 8B source into Mesolitica's Malaysian-Llama-3.2-1B-Instruct-v0.1 target (Mesolitica, n.d.), evaluated on a 23,206-item within-study holdout of MalayMMLU (Poh et al., 2024). All benchmark scores are reported as percentages; checkpoint identities, dataset details, and scoring protocols are provided in Appendix D.

Baselines. We compare against the reference, target, and source checkpoints, as well as Transport and Merge (Cui et al., 2026), standard knowledge distillation (KD) (Hinton et al., 2015), MiniLLM (Gu et al., 2024), and FuseLLM (Wan et al., 2024). The larger source is excluded from best- and second-best rankings. In the tables, TF and DF denote training-free and data-free construction. Transport and Merge uses calibration activations to align and transport weights; we adopt its merging-only variant without distillation or post-merging training. Its MalayMMLU scores are taken from Table 1 of Cui et al. (2026) under the original evaluation protocol. Our KD baseline trains the target on domain-matched examples, combining forward KL divergence from the source's next-token distribution with next-token cross-entropy over concatenated prompts and responses. FuseLLM aligns and combines predictions from multiple source models and continually trains a target on the combined output distributions. Unlike SMM's checkpoint-only construction, these baselines require calibration activations or text-based training. Attention-only, FFN-only, and joint-transfer ablations are presented in Section 4.3.

Implementation and Evaluation Protocol. For a fixed configuration, SMM and its module ablations construct merged checkpoints without task examples, activation probes, or gradient updates. Qwen configuration selection is described in Appendix D.1. The subsequent evaluation is retrospective

rather than a prospectively frozen confirmatory study. For IFEval, we report the 433-item holdout excluding the 108 items previously used for development. SMM reuses a single embedding alignment across layers and employs 128-dimensional sketches with blocked scoring for neuron matching. On an NVIDIA A100-SXM4-80GB GPU, core merging takes 30.53, 23.87, 35.11, and 23.47 seconds for math, code, instruction-following, and multilingual transfer, respectively. These wall-clock times cover anchor alignment and weight transport/merging, excluding checkpoint loading and saving, configuration selection, and benchmark evaluation. All SMM checkpoints retain the target architecture and parameter count, requiring no additional modules or source-model calls at deployment.

4.2 RESULTS

Tables 1–3 report capability transfer and general-capability retention across the benchmarks described in Section 4.1. Score differences are computed from unrounded values and expressed in percentage points.

Table 1: SMM and baseline performance on math and code (%). General measures general-capability retention. Best and second-best available scores in each column, excluding Source, are **bold** and underlined. A dash denotes an unavailable score.

			Math			Code		
Method	TF	DF	MATH-500	GSM8K	General	MBPP+	Human-Eval+	General
Reference	–	–	42.40	47.16	71.37	**56.35**	47.56	71.37
Target	–	–	49.40	**72.48**	72.86	47.62	47.56	72.86
Source	–	–	84.20	95.22	60.27	66.14	80.49	79.08
Transport and Merge	✓	×	50.60	72.33	**73.20**	47.62	47.56	72.86
FuseLLM	×	×	40.60	65.43	72.28	46.30	40.24	72.07
KD	×	×	48.00	69.75	72.57	45.24	**50.00**	72.34
MiniLLM	×	×	50.80	68.01	68.82	51.59	46.34	**73.18**
Ours (SMM)	✓	✓	**53.00**	71.65	72.92	49.47	45.73	72.77

Role of the Target–Reference Direction. The contrast between the math and code results is consistent with SMM's design: the task vector $T - R$ guides the selection of source FFN units, so transfer quality depends on how well the chosen Target–Reference pair captures the intended capability. In math, the Target substantially outperforms the Reference on both MATH-500 and GSM8K, providing evidence that $T - R$ reflects useful task-specific changes. For the code pair used here, the Target scores lower than the Reference on MBPP+ and matches it on HumanEval+, providing less evidence that their difference represents a beneficial code-specific direction. SMM nevertheless improves MBPP+ over the Target, although it remains below the Reference, and does not improve HumanEval+. We interpret this pattern as consistent with weaker selection guidance from the chosen code Target–Reference pair. It highlights the importance of checkpoint selection when applying SMM: a stronger Source alone does not ensure effective transfer if $T - R$ provides an insufficiently informative task direction. This observation also supports the rationale for using three models in data-free merging: the Target and Reference jointly provide a task-specific selection direction, while the Source supplies candidate parameters for transfer.

Improvement over the Target. Across four separately constructed Qwen checkpoints, SMM improves five of seven capability benchmarks. The gains are 3.60 on MATH-500 and 1.85 on MBPP+ (Table 1), followed by 2.08 on IFEval, 0.33 on Belebele, and 1.25 on XNLI (Table 2). GSM8K and HumanEval+ decrease by 0.83 and 1.83, respectively, indicating that the gains depend on the benchmark. The exploratory Malay transfer extends the improvements to a language-adapted Target: overall accuracy rises from 41.91% to 47.98%, a gain of 6.08, with improvements in every subject category (Table 3).

Comparison with Trained Transfer. Beyond improving the Target, SMM is competitive with methods that train on text. On MATH-500, its 53.00% accuracy exceeds MiniLLM's 50.80%, although MiniLLM scores higher on both code benchmarks (Table 1). On Belebele, SMM achieves

Table 2: SMM and baseline performance on instruction following and multilingual understanding (%). General measures general-capability retention. Best and second-best available scores in each column, excluding Source, are **bold** and underlined. A dash denotes an unavailable score.

			Instruction following		Multilingual understanding		
Method	TF	DF	IFEval	General	Belebele	XNLI	General
Reference	–	–	26.33	71.37	59.99	49.32	71.37
Target	–	–	40.88	72.86	63.91	52.92	72.86
Source	–	–	70.21	83.75	79.71	66.39	83.75
Transport and Merge	✓	×	39.72	73.04	63.75	52.78	72.83
FuseLLM	×	×	39.03	72.57	63.02	**55.31**	71.82
KD	×	×	**44.80**	73.45	63.54	53.09	72.65
MiniLLM	×	×	42.96	**74.41**	62.33	54.27	71.53
Ours (SMM)	✓	✓	42.96	72.84	**64.25**	54.16	**72.88**

Table 3: SMM and baseline performance on MalayMMLU (%) with SMM using the exploratory follow-up configuration in Figure 2. Best and second-best scores under the shared evaluation protocol are **bold** and underlined, excluding Source and the published Transport and Merge row. A dash denotes an unavailable score.

Method	TF	DF	Human-ities	Language	Others	STEM	Social sci.	Overall
Reference	–	–	43.80	38.63	39.94	40.24	40.91	40.61
Target	–	–	43.46	38.38	45.17	42.75	41.86	41.91
Source	–	–	67.28	65.48	61.86	63.90	62.70	64.23
Transport and Merge	✓	×	46.19	40.52	46.63	45.19	43.77	—
FuseLLM	×	×	48.01	40.21	47.42	45.40	46.62	45.22
KD	×	×	47.13	38.31	46.45	44.50	45.30	43.93
Ours (SMM)	✓	✓	**50.46**	**46.18**	**48.07**	**47.49**	**48.18**	**47.98**

the highest score among the target-sized models, 64.25%; on IFEval, it matches MiniLLM at 42.96%, behind KD's 44.80% (Table 2). In the Malay setting, SMM exceeds both KD and FuseLLM in all five subject categories (Table 3). These comparisons show that direct parameter transfer can compete with trained transfer across different capabilities, with the strongest method varying by benchmark.

General Capability Retention. We next examine whether these task-specific changes accompany changes in broader capabilities. Across the four Qwen transfers, SMM scores 72.77%–72.92%, within 0.09 points of the Target's 72.86% (Tables 1 and 2). Its capability gains therefore coexist with small aggregate changes on these general tasks, while the Target architecture and parameter count remain unchanged.

4.3 Analysis

We next examine which modules support the observed transfer and how the amount of transferred information affects performance.

Which Modules Should Be Transferred? Table 4 compares attention-only, FFN-only, and joint FFN–attention transfer in the archived Malay module study. SMM (FFN-only) achieves 45.99% overall, exceeding attention-only by 4.58 points and joint transfer by 0.41 points. It also leads in four subject categories, while joint transfer scores 0.21 points higher on STEM. These results support retaining Target attention in the evaluated configurations, consistent with Section 3.4. The variants use separately calibrated budgets, so the comparison does not isolate module choice at an identical perturbation magnitude. Their configurations are detailed in Appendix D.4.

Table 4: SMM module-transfer ablations on MalayMMLU (%). Ours (SMM) is the FFN-only variant; the other rows change its transferred modules. Protocol details are in Appendix D.4. Best and second-best scores in each column are **bold** and underlined.

Method	TF	DF	Human-ities	Language	Others	STEM	Social sci.	Overall
Attention-only	✓	✓	42.71	38.02	44.71	42.84	41.19	41.41
Ours (SMM)	✓	✓	**48.09**	**43.05**	**48.49**	46.71	**45.57**	**45.99**
FFN + Attention	✓	✓	47.28	42.44	48.02	**46.92**	45.42	45.58

Sensitivity to Update Budget and Coverage. Having examined SMM's module choice, we vary the norm budget β and the selected-slot coverage ρ in Figure 2. The reference SMM configuration, $\beta = 0.10$ and $\rho = 0.50$, gives the highest overall accuracy among the evaluated points in both sweeps and corresponds to the SMM row in Table 3. Individual categories peak at different settings, and increasing the update budget does not uniformly improve performance. Language shows the largest variation across the coverage sweep, with a range of 6.58 points. These patterns support controlling update magnitude and coverage separately. Appendix D.3 provides the sweep configurations and evaluation chronology.

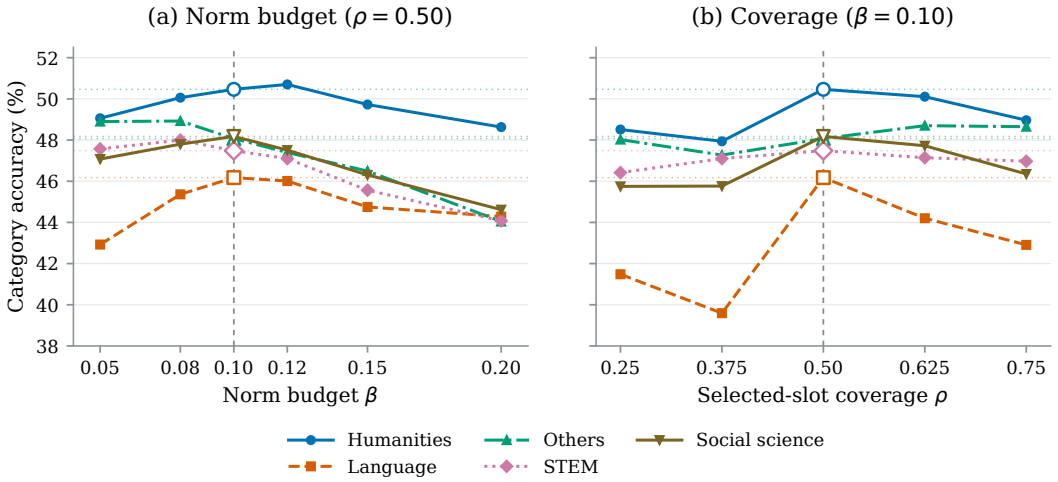

Figure 2: SMM sensitivity on MalayMMLU by subject category. (a) Varying β at fixed $\rho = 0.50$. (b) Varying ρ at fixed $\beta = 0.10$. Vertical dashed lines mark the reference configuration. Colored horizontal dotted lines indicate its category accuracies.

5 Conclusion and Future Work

In this paper, we present SMM, a framework for selective heterogeneous model merging. By aligning shared-token embeddings and selecting source FFN units using the target's task vector, SMM transfers parameters while preserving the target architecture and attention. For a fixed configuration, merging requires neither calibration data nor additional training. Experiments within the Qwen and Llama families show improvements over target models on several downstream benchmarks, with mixed results relative to distillation and training-based baselines. These findings support selective parameter transfer as a viable approach to reusing heterogeneous checkpoints.

Several limitations remain. Our evaluation is restricted to within-family transfers, leaving cross-family compatibility unverified. Future work will investigate alignment across model families with different tokenizers and architectures, and develop adaptive criteria for deciding whether and how strongly to transfer each attention layer. We will also explore using SMM as an initialization for limited-budget distillation when data are available, assessing whether these complementary approaches can improve the trade-off between transfer quality and computational cost.

AI USE STATEMENT

Generative AI tools were used to aid or polish writing, including improving wording and readability, organizing the presentation, and checking consistency in terminology, captions, and reference formatting. The authors are responsible for the final content of the paper, including all AI-assisted text.

ETHICS STATEMENT

This work studies model merging using existing language-model checkpoints and established benchmarks. The intended benefit is to reuse model capabilities with lower computational requirements for a fixed merging configuration. Use of the resulting models should respect the licenses and usage restrictions of the underlying checkpoints and datasets. As with other language models, deployment requires evaluation of potential bias, harmful outputs, and privacy risks in the intended application.

REPRODUCIBILITY STATEMENT

The SMM procedure is described in Section 3 and Algorithm 1. Appendices B and C provide the derivations and implementation details. Section 4.1 and Appendix D document the model checkpoints, evaluation data and protocols, configuration selection, and runtime measurement scope. Appendix E specifies the baseline training settings and objectives.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-Policy Distillation of Language Models: Learning from Self-Generated Mistakes. In *International Conference on Learning Representations*, pp. 21246–21263, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/5be69a584901a26c521c2b51e40a4c20-Abstract-Conference.html.

Lucas Bandarkar, Davis Liang, Benjamin Muller, Mikel Artetxe, Satya Narayan Shukla, Donald Husa, Naman Goyal, Abhinandan Krishnan, Luke Zettlemoyer, and Madian Khabsa. The Belebele benchmark: a parallel reading comprehension dataset in 122 language variants. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 749–775. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.44. URL https://aclanthology.org/2024.acl-long.44/.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020. doi: 10.1609/aaai.v34i05.6239. URL https://ojs.aaai.org/index.php/AAAI/article/view/6239.

Shilian Chen, Jie Zhou, Qin Chen, Wen Wu, Xin Li, Qi Feng, and Liang He. Can heterogeneous language models be fused? arXiv preprint arXiv:2604.01674, 2026. URL https://arxiv.org/abs/2604.01674.

Runxi Cheng, Feng Xiong, Yongxian Wei, Wanyun Zhu, and Chun Yuan. Whoever started the interference should end it: Guiding data-free model merging via task vectors. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 10121–10143. PMLR, 2025. URL https://proceedings.mlr.press/v267/cheng25h.html.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. arXiv preprint arXiv:1803.05457, 2018. URL https://arxiv.org/abs/1803.05457.

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

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2475–2485. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-1269. URL https://aclanthology.org/D18-1269/.

Chenhang Cui, Binyun Yang, Fei Shen, Yuxin Chen, Jingnan Zheng, Xiang Wang, An Zhang, and Tat-Seng Chua. Transport and merge: Cross-architecture merging for large language models. In *Proceedings of the 43rd International Conference on Machine Learning*, Seoul, South Korea, 2026. URL https://icml.cc/virtual/2026/poster/66026.

Guodong Du, Zhuo Li, Xuanning Zhou, Junlin Li, Zesheng Shi, Wanyu Lin, Ho-Kin Tang, Xiucheng Li, Fangming Liu, Wenya Wang, Min Zhang, and Jing Li. Knowledge fusion of large language models via modular SkillPacks. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/32ebb6b560ee58abbdae834e5f37cb5d-Abstract-Conference.html.

Luyang Fang, Xiaowei Yu, Jiazhang Cai, Yongkai Chen, Shushan Wu, Zhengliang Liu, Zhenyuan Yang, Haoran Lu, Xilin Gong, Yufang Liu, Terry Ma, Wei Ruan, Ali Abbasi, Jing Zhang, Tao Wang, Ehsan Latif, Wei Liu, Wei Zhang, Soheil Kolouri, Xiaoming Zhai, Dajiang Zhu, Wenxuan Zhong, Tianming Liu, and Ping Ma. Knowledge distillation and dataset distillation of large language models: Emerging trends, challenges, and future directions. *Artificial Intelligence Review*, 59(1): 17, 2026. doi: 10.1007/s10462-025-11423-3. URL https://doi.org/10.1007/s10462-025-11423-3.

M. A. Ganaie, Minghui Hu, A. K. Malik, M. Tanveer, and P. N. Suganthan. Ensemble deep learning: A review. *Engineering Applications of Artificial Intelligence*, 115:105151, October 2022. doi: 10.1016/j.engappai.2022.105151. URL https://doi.org/10.1016/j.engappai.2022.105151.

Antonio Andrea Gargiulo, Donato Crisostomi, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, and Emanuele Rodolà. Task singular vectors: Reducing task interference in model merging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18695–18705, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Gargiulo_Task_Singular_Vectors_Reducing_Task_Interference_in_Model_Merging_CVPR_2025_paper.html.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.446. URL https://aclanthology.org/2021.emnlp-main.446/.

Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vladimir Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. Arcee's MergeKit: A toolkit for merging large language models. In Franck Dernoncourt, Daniel Preoţiuc-Pietro, and Anastasia Shimorina (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 477–485, Miami, Florida, US, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-industry.36. URL https://aclanthology.org/2024.emnlp-industry.36/.

Matt Gorbett and Suman Jana. Ventriloquist LLMs: Linear alignment of late-stage representations. arXiv preprint arXiv:2603.18908, 2026. URL https://arxiv.org/abs/2603.18908v5.

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

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/8ac015d409635f196f9e3e9dcfb9a94e-Abstract-Conference.html.

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. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*,

volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. URL https://research.google/pubs/distilling-the-knowledge-in-a-neural-network/.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

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

Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. DistiLLM: Towards Streamlined Distillation for Large Language Models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 24872–24895. PMLR, 2024. URL https://proceedings.mlr.press/v235/ko24c.html.

Jongwoo Ko, Tianyi Chen, Sungnyun Kim, Tianyu Ding, Luming Liang, Ilya Zharkov, and Se-Young Yun. DistiLLM-2: A Contrastive Approach Boosts the Distillation of LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 31044–31062. PMLR, 2025. URL https://proceedings.mlr.press/v267/ko25a.html.

Andrew Lee, Melanie Weber, Fernanda Viégas, and Martin Wattenberg. Shared global and local geometry of language model embeddings. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=aJDykpJAYF.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems*, volume 36, pp. 21558–21572, 2023. doi: 10.52202/075280-0943. URL https://papers.neurips.cc/paper_files/paper/2023/hash/43e9d647ccd3e4b7b5baab53f0368686-Abstract-Conference.html.

Zhenyi Lu, Chenghao Fan, Wei Wei, Xiaoye Qu, Dangyang Chen, and Yu Cheng. Twin-Merging: Dynamic Integration of Modular Expertise in Model Merging. In *Advances in Neural Information Processing Systems*, volume 37, pp. 78905–78935. Curran Associates, Inc., 2024. doi: 10.52202/079017-2505. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/8fcd17eb91bae20d9826786d7d6be799-Abstract-Conference.html.

Phoomraphee Luenam, Andreas Spanopoulos, Amit Sant, Sotiris Anagnostidis, Thomas Hofmann, and Sidak Pal Singh. Model fusion via retrofitting. In *Proceedings of the 43rd International Conference on Machine Learning*, Seoul, South Korea, 2026. URL https://icml.cc/virtual/2026/poster/63921.

Mesolitica. Malaysian-Llama-3.2-1B-Instruct-v0.1. Hugging Face model card, n.d. URL https://huggingface.co/mesolitica/Malaysian-Llama-3.2-1B-Instruct-v0.1. Accessed September 26, 2026.

Bowen Ping, Shuo Wang, Hanqing Wang, Xu Han, Yuzhuang Xu, Yukun Yan, Yun Chen, Baobao Chang, Zhiyuan Liu, and Maosong Sun. Delta-CoMe: Training-Free Delta-Compression with Mixed-Precision for Large Language Models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 31056–31077. Curran Associates, Inc., 2024. doi: 10.52202/079017-0978. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/37664246a1e07e212ddacea6e5a523f2-Abstract-Conference.html.

Soon Chang Poh, Sze Jue Yang, Jeraelyn Ming Li Tan, Lawrence Leroy Tze Yao Chieng, Jia Xuan Tan, Zhenyu Yu, Foong Chee Mun, and Chee Seng Chan. MalayMMLU: A multitask benchmark for the low-resource Malay language. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 650–669. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-emnlp.36. URL https://aclanthology.org/2024.findings-emnlp.36/.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025. URL https://arxiv.org/abs/2412.15115v2. Version 2, January 2025.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8732–8740, 2020. doi: 10.1609/aaai.v34i05.6399. URL https://ojs.aaai.org/index.php/AAAI/article/view/6399.

Peter H. Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966. doi: 10.1007/BF02289451. URL https://doi.org/10.1007/BF02289451.

Noam Shazeer. GLU variants improve transformer. arXiv preprint arXiv:2002.05202, 2020. URL https://arxiv.org/abs/2002.05202.

Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. In *Advances in Neural Information Processing Systems*, volume 33, pp. 22045–22055, 2020. URL https://papers.neurips.cc/paper/2020/hash/fb2697869f56484404c8ceee2985b01d-Abstract.html.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=r1Aab85gg.

Bedionita Soro, Aoxuan Zhang, Bruno Andreis, Jaehyeong Jo, Song Chong, and Sung Ju Hwang. LS-merge: Merging language models in latent space. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/ef0f1d04d0c43fd3e8f9fb9e56111d50-Abstract-Conference.html.

George Stoica, Pratik Ramesh, Boglarka Ecsedi, Leshem Choshen, and Judy Hoffman. Model merging with SVD to tie the Knots. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/0d4f8a5109c5083b5307fcd0bddae7a7-Abstract-Conference.html.

Fanqi Wan, Xinting Huang, Deng Cai, Xiaojun Quan, Wei Bi, and Shuming Shi. Knowledge fusion of large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=jiDsk12qcz.

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*, pp. 21618–21642. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.1096. URL https://aclanthology.org/2025.emnlp-main.1096/.

Ke Wang, Nikolaos Dimitriadis, Guillermo Ortiz-Jimenez, François Fleuret, and Pascal Frossard. Localizing task information for improved model merging and compression. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 50268–50287. PMLR, 2024. URL https://proceedings.mlr.press/v235/wang24k.html.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 23965–23998. PMLR, 2022. URL https://proceedings.mlr.press/v162/wortsman22a.html.

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*, volume 36, pp. 7093–7115, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1644c9af28ab7916874f6fd6228a9bcf-Abstract-Conference.html.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. AdaMerging: Adaptive model merging for multi-task learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/62868cc2fc1eb5cdf321d05b4b88510c-Abstract-Conference.html.

Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model merging in LLMs, MLLMs, and beyond: Methods, theories, applications, and opportunities. *ACM Computing Surveys*, 58(8):216:1–216:41, February 2026. ISSN 0360-0300. doi: 10.1145/3787849. URL https://doi.org/10.1145/3787849.

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 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 57755–57775. PMLR, 2024. URL https://proceedings.mlr.press/v235/yu24p.html.

Ming Zhong, Chenxin An, Weizhu Chen, Jiawei Han, and Pengcheng He. Seeking Neural Nuggets: Knowledge Transfer in Large Language Models from a Parametric Perspective. In *International Conference on Learning Representations*, pp. 25576–25596, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/6d6dc5c9ad19816d745535d352dc4295-Abstract-Conference.html.

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. URL https://arxiv.org/abs/2311.07911.

Appendix Contents

A Novelty and Contribution 14
A.1 Comparison with Heterogeneous Model Merging 14
A.2 Comparison with Pairwise Distillation 15
B Coordinate Transport and Solution of P 15
C Algorithm and Implementation Details 17
D Experimental Setup 18
E Baseline Details 20

A Novelty and Contribution

SMM transfers capabilities from a larger Source into a smaller Target by establishing residual-coordinate correspondence, selecting complete gated FFN units using the Target–Reference displacement, and bounding their injected updates. For a fixed merging configuration, construction uses checkpoint weights without input examples, activation collection, or gradient updates. The resulting model retains the Target architecture and parameter count. The comparisons below explain how these design choices relate to heterogeneous merging and pairwise distillation.

A.1 Comparison with Heterogeneous Model Merging

Heterogeneous merging requires correspondence between parameters with different dimensions and computational roles. Transport and Merge (Cui et al., 2026) estimates neuron correspondences through optimal transport of activations collected from inputs. LS-Merge (Soro et al., 2026) trains a weight encoder and decoder to perform merging in a learned latent space. HeteroFusion (Chen et al., 2026) operates on heterogeneous LoRA adapters through structural alignment and conflict-aware merging. These approaches establish correspondence through activation evidence, learned weight representations, or adapter structure, respectively.

SMM obtains its residual-coordinate map from shared-token embeddings and applies it consistently to the input-facing and output-facing FFN weights. It then handles depth through layer grouping and

intermediate width through unit assignment. This separates coordinate transport from the choice of which Source units to transfer. The mapping and the Target–Reference displacement together provide the information used for selection, so no learned weight codec or fitted alignment module is required during model construction.

A related parameter-transfer approach, Seeking Neural Nuggets (Zhong et al., 2024), extracts knowledge-related parameters using sensitivity and injects them into smaller models through a LoRA intermediary. SMM instead compares complete Gate–Up–Down units in the transported coordinates and writes selected updates into existing Target FFNs. Its contribution combines cross-width transport with task-guided unit selection and bounded injection. Transport and Merge supplies the activation-based comparison in our experiments. Evaluating learned latent-space and adapter-based pipelines would additionally require constructing their learned weight representations or adapter inputs. We therefore compare their alignment mechanisms here, separately from the implemented full-checkpoint baselines.

A.2 Comparison with Pairwise Distillation

Pairwise distillation transfers a Source teacher's behavior by optimizing a Target student on text inputs. Recent methods address the choice of training sequences and distributional objectives: GKD (Agarwal et al., 2024) uses student-generated sequences, MiniLLM (Gu et al., 2024) optimizes reverse KL, and DistiLLM-2 (Ko et al., 2025) couples teacher- and student-generated responses with contrastive objectives. SMM obtains its candidate updates directly from Source weights and uses the Target's recorded adaptation to select them. It therefore transfers parameters without generating teacher responses or optimizing a student loss during construction.

Pairwise distillation can also prepare compatible checkpoints for a subsequent merging stage. FuseChat (Wan et al., 2025), including its 2.0 configuration, first transfers heterogeneous sources into Target-format models and then merges their parameters. This pipeline separates capability acquisition from parameter combination. SMM establishes correspondence directly between the available heterogeneous checkpoints, making an intermediate family of distilled Target-format models unnecessary. FuseLLM (Wan et al., 2024) offers another route by consolidating multiple teachers' output distributions through continued training; our experiments examine a single designated Source for each Target transfer setting.

Alternatively, modular methods preserve task-specific information for selective reuse. Twin-Merging (Lu et al., 2024) separates shared and exclusive knowledge and combines it according to the input. SMM selects its FFN units once during merging and produces one fixed checkpoint per transfer setting, without an inference-time router or a collection of task-specific modulators.

GraftLLM (Du et al., 2026) connects pairwise capability acquisition with modular storage: it trains the Target and compresses the resulting updates into SkillPacks. Delta-CoMe (Ping et al., 2024) likewise addresses the storage of acquired capabilities by compressing existing fine-tuning deltas with mixed precision. SMM's selection acts earlier: it constructs candidate increments from transported Source units relative to Reference, before any new Target training. Its coverage and norm budgets control which existing FFN slots change and by how much. Thus the reusable object is a selected heterogeneous parameter update, and deployment uses the original Target architecture. Our experimental comparisons use standard KD, MiniLLM, and FuseLLM for trained transfer; the modular pipelines above clarify the distinct roles of acquisition, combination, and storage.

B Coordinate Transport and Solution of P

B.1 The Coordinate Relation Defined by P

The role of P is to define the coordinate relation used to transport parameters between models. Shared-token embeddings provide the anchors from which this relation is computed. Let the Source residual width be $d_S \geq d_R$, with Reference and Target sharing width d_R and compatible parameter coordinates through their common pretraining lineage. Following the row-vector convention of the main text, let

$$P \in \mathbb{R}^{d_S \times d_R}, \qquad P^\top P = I_{d_R}.$$

A Target-width vector x is lifted into Source coordinates as $xP^\top$; a Source vector h_S is expressed in the selected Target-width coordinates as $h_S P$. The constraint makes the lift an isometry:

$$\langle xP^\top, yP^\top \rangle = \langle x, y \rangle, \qquad (xP^\top)P = x.$$

For unequal widths, $PP^\top$ is the orthogonal projector onto the selected d_R-dimensional Source subspace. Thus P supplies a shared coordinate convention for comparing and transporting residual-facing parameters.

Using this same map on both sides of a Source residual branch B_S gives

$$\big[xP^\top + B_S(xP^\top)\big]P = x + B_S(xP^\top)P. \tag{7}$$

The identity path is preserved because $P^\top P = I$. With different adjacent maps, this path would instead contain $xP_\ell^\top P_{\ell+1}$. Reusing one P across depths therefore avoids an additional inter-layer coordinate transformation and gives the transported modules a consistent residual coordinate system.

B.2 Computing P from Checkpoint Anchors

Embedding vectors are the inputs to the residual stream. We pair rows by shared token identity and normalize each selected row to unit length, forming $E_S \in \mathbb{R}^{n \times d_S}$ and $E_R \in \mathbb{R}^{n \times d_R}$. Their cross-product

$$C = E_S^\top E_R \in \mathbb{R}^{d_S \times d_R}$$

accumulates the correspondence supplied by these token pairs. We solve the rectangular correlation problem

$$P^\star = \arg \max_{P^\top P = I_{d_R}} \operatorname{tr}(P^\top C). \tag{8}$$

The objective equals $\langle E_S P, E_R \rangle_F$, while the orthonormal-column constraint prevents arbitrary scaling and preserves the Target-width geometry under lifting. For the thin singular value decomposition

$$C = U\Sigma V^\top, \qquad U \in \mathbb{R}^{d_S \times d_R}, \quad V \in \mathbb{R}^{d_R \times d_R},$$

the solution is

$$\boxed{P^\star = UV^\top.} \tag{9}$$

To see this, write

$$\operatorname{tr}(P^\top C) = \sum_{k=1}^{d_R} \sigma_k(C)\, u_k^\top P v_k \le \sum_{k=1}^{d_R} \sigma_k(C).$$

Each factor $u_k^\top P v_k \le 1$ because both u_k and Pv_k have unit norm. Substituting $P = UV^\top$ attains the bound and satisfies $P^\top P = I$. If C is rank deficient, an orthonormal completion gives an optimizer, which need not be unique.

The corresponding least-squares formulation is the reverse-map problem:

$$\|E_R P^\top - E_S\|_F^2 = \|E_R\|_F^2 + \|E_S\|_F^2 - 2\operatorname{tr}(P^\top C). \tag{10}$$

Its first two terms are constant under $P^\top P = I$. For rectangular P, the forward residual $\|E_S P - E_R\|_F^2$ instead includes the generally nonconstant term $\operatorname{tr}(P^\top E_S^\top E_S P)$, so it is not the objective in Equation (8). The computation consists of anchor pairing, row normalization, cross-product accumulation, and one thin SVD; it needs no model inputs, hidden-state collection, or optimization steps.

B.3 FFN Parameters Induced by the Coordinate Map

The coordinate relation determines how input-reading and output-writing weights are transported. Let $W_{S,g}, W_{S,u} \in \mathbb{R}^{m_S \times d_S}$ and $W_{S,d} \in \mathbb{R}^{d_S \times m_S}$ define the gated Source FFN

$$F_S(h) = \big[\mathrm{SiLU}(hW_{S,g}^\top) \odot (hW_{S,u}^\top)\big] W_{S,d}^\top.$$

The induced parameter map is

$$\widetilde{W}_g = W_{S,g}P, \qquad \widetilde{W}_u = W_{S,u}P, \qquad \widetilde{W}_d = P^\top W_{S,d}. \tag{11}$$

Proposition 1 (Transport of a gated FFN)**.** *The FFN defined by Equation (11) satisfies*

$$\widetilde{F}_S(x) = F_S(xP^\top)P \tag{12}$$

for every Target-width row vector x.

Proof. The input projections obey $x\widetilde{W}_g^\top = (xP^\top)W_{S,g}^\top$ and $x\widetilde{W}_u^\top = (xP^\top)W_{S,u}^\top$, and the output projection obeys $\widetilde{W}_d^\top = W_{S,d}^\top P$. Substitution into the gated FFN gives Equation (12). □

For a complete FFN unit, let g_j^S, u_j^S be the transposes of Gate and Up row j, and d_j^S be Down column j. Their joint transport is

$$(g_j^S, u_j^S, d_j^S) \longmapsto (P^\top g_j^S, P^\top u_j^S, P^\top d_j^S).$$

The same map therefore transports all three components while preserving their pairing and the gated unit's input–output structure.

B.4 From Transported Units to Task-Guided Updates

After transport, the concatenated Target, Reference, and Source unit vectors $t_i, r_i, \tilde{s}_j \in \mathbb{R}^{3d_R}$ share parameter coordinates. Reference supplies the common origin for

$$\tau_i = t_i - r_i, \qquad \Delta_{ij}^S = \tilde{s}_j - r_i.$$

SMM compares these vectors using Equation (4). The Target–Reference displacement records the Target's existing adaptation and provides the selection direction. The donor increment is defined relative to the Reference unit at the candidate Target slot. Coordinate transport handles residual width, relative-depth grouping handles depth, and assignment of complete units handles intermediate FFN width. Selection, aggregation, and norm control then produce the update described in Appendix C.

C Algorithm and Implementation Details

C.1 Anchors and Sketched Matching

The implementation samples deterministic common token indices and normalizes each selected row. In addition to input-embedding anchors E_S, E_R, it uses paired output-head rows O_S, O_R as a second anchor set. Corresponding rows are paired by shared token identity. With n_E and n_O valid rows in the two sets, their empirical cross-products receive equal weight:

$$C = \tfrac{1}{2}\left(\frac{E_S^\top E_R}{n_E} + \frac{O_S^\top O_R}{n_O}\right).$$

The thin SVD $C = U\Sigma V^\top$ gives $P^\star = UV^\top$, extending the input-anchor construction in Equation (2). If a paired output head is unavailable, only input-embedding anchors are used. The Target embeddings and output head are retained unchanged; these anchors are used only to compute the coordinate map.

For matching, the implementation projects Gate, Up, and Down onto a common 128-dimensional residual basis and concatenates their sketches. Reference subtraction occurs after this linear projection. Scores approximate Equation (4); they are not exact full-coordinate cosines. Each Target slot retains a shortlist of Source candidates. Slots are processed in descending order of their best score and take the highest-scoring unused positive donor. If the shortlist is exhausted, the remaining candidates are searched. Zero task directions and slots without an unused positive donor are rejected.

Among the admissible slots, coverage $\rho \in (0, 1]$ retains at most $\lceil \rho m_T \rceil$ slots, ranked by mean matching score across the source group; stable ties follow target-slot order.

With n anchor rows, the cross-product costs $O(nd_S d_R)$, and its thin SVD costs $O(d_S d_R^2)$ for $d_S \geq d_R$. For one layer pair with FFN widths m_T, m_S and sketch length k, pairwise scoring costs $O(m_T m_S k)$. Blocking and top-candidate retention reduce memory but do not remove this pairwise work. These costs exclude checkpoint loading and selected full-weight transport.

C.2 Depth Aggregation

Contiguous Source groups are assigned by relative depth. Using zero-based indices, Target layer ℓ receives

$$\left\{ s : \left\lfloor \frac{\ell L_S}{L_T} \right\rfloor \leq s < \left\lfloor \frac{(\ell+1)L_S}{L_T} \right\rfloor \right\}.$$

Each Source layer is matched independently before the resulting tensors are averaged. A Target slot is eligible only if it has an admitted match in every group member. Gate, Up, and Down use the same assignments, with averaging performed separately for each matrix. This compresses parameters; it does not reproduce the sequential computation of the deeper Source.

C.3 Norm Control and Injection

For each transported and aggregated FFN matrix, subtract Reference and zero the rejected slots to form d_W. If $d_W \neq 0$, the implemented scale is

$$\lambda_W = \min\left\{1, \frac{\beta \max(\|W_T\|_F, \epsilon)}{\|d_W\|_F}\right\}, \qquad W_M = W_T + \lambda_W d_W.$$

A zero update remains zero, and $\beta = 0$ disables injection. The operation is performed in FP32 and cast back to the Target dtype. Separate caps preserve the direction within each matrix but can alter the concatenated complete-neuron direction. The resulting assembled update includes these matrix-specific scales.

Anchor count, sketch size, coverage, and β are recorded in each run configuration. The main exploratory Llama Malay configuration uses 50% coverage and $\beta = 0.10$. The archived module-study configurations for Table 4 are specified in Appendix D.4. The merge itself does not use task examples or activation probes. Reported performance should be tied to those run-specific values rather than to software defaults.

D Experimental Setup

D.1 Checkpoint Identities and Selection

Table 5 lists the model roles for both families. Checkpoint revisions are retained in the archived run configurations.

Table 5: Model checkpoints used for the Qwen and Llama transfers.

Role	Model
Qwen2.5 family (Qwen Team, 2025)	
Reference	Qwen2.5-1.5B
Target	Qwen2.5-1.5B-Instruct
Math Source	Qwen2.5-Math-7B-Instruct
Code Source	Qwen2.5-Coder-7B-Instruct
Instruction / multilingual Source	Qwen2.5-7B-Instruct
Llama family (Grattafiori et al., 2024)	
Reference	Llama-3.2-1B-Instruct
Target	Malaysian-Llama-3.2-1B-Instruct-v0.1 (Mesolitica, n.d.)
Source	Llama-3.1-8B-Instruct

Reference Lineage. For Qwen, R is the base checkpoint corresponding to the instruction-tuned target. The Mesolitica model card (Mesolitica, n.d.) reports continued tuning from Llama-3.2-1B, while the archived Malay runs use Llama-3.2-1B-Instruct as R. This instruction-tuned reference

shares the target's base-model lineage but is not claimed to be its direct initialization. Thus, the Malay task vector is a target–reference contrast and need not isolate only Malay adaptation.

Source, Target, and Reference use frozen public weights. The Qwen configurations were selected from multiple candidates using domain development scores and a general-retention constraint. Each domain has its own SMM checkpoint. The saved run configurations, predictions, score files, and checkpoint-rebuild identities should accompany the released artifact; scores alone are insufficient to reproduce selection. The four Qwen merges share one Target but use domain-specific Sources; they do not form a single checkpoint with all four abilities. The 7B-to-1.5B comparisons vary model width within the Qwen2.5 family. Attention-only is an SMM ablation. Transport and Merge (Cui et al., 2026) uses activation-based transport; FuseLLM (Wan et al., 2024), KD (Hinton et al., 2015), and MiniLLM (Gu et al., 2024) train the target. Score differences reflect each method's recorded information, tuning, and training budgets.

D.2 Datasets and Evaluation Protocol

Benchmarks and Metrics. The Qwen evaluation covers mathematical reasoning, code generation, instruction following, and multilingual understanding. We report accuracy on MATH-500 and GSM8K, and enhanced-test pass@1 on MBPP+ and HumanEval+ with one generated completion per problem. IFEval is evaluated using strict prompt-level accuracy. For Belebele and XNLI, we compute accuracy within each language and report an equal-language macro average, giving each evaluated language the same weight.

General-Capability Retention. We measure retention using the equal-task macro accuracy of ARC-Easy (Clark et al., 2018), PIQA (Bisk et al., 2020), and WinoGrande (Sakaguchi et al., 2020). This evaluation uses the ARC-Easy test split and the public validation splits of PIQA and WinoGrande. A development-disjoint variant is also computed to check the effect of PIQA overlap with development data.

Evaluation Protocol. Within each benchmark, our evaluations use the same items, prompts, and scoring procedure; the published Malay Transport and Merge row follows the original paper's protocol. For IFEval, we consistently report results on the held-out subset that excludes examples previously used for development. Each SMM configuration remains fixed during its reported benchmark evaluation. Once a configuration is fixed, constructing the merged checkpoint uses model weights without calibration examples, activation collection, or gradient updates.

Result Reporting. Tables 1 and 2 report descriptive benchmark point estimates for all evaluated methods; unavailable results remain unreported. Baseline information requirements, tuning budgets, and implementation details are provided in Appendix E.

D.3 Merging Configurations

The main Malay result and the centered sweeps use a 128-dimensional matching sketch. We center the sweeps at $\beta = 0.10$, $\rho = 0.50$. The norm-budget sweep uses $\beta \in \{0.05, 0.08, 0.10, 0.12, 0.15, 0.20\}$ at fixed $\rho = 0.50$; the coverage sweep uses $\rho \in \{0.25, 0.375, 0.50, 0.625, 0.75\}$ at fixed $\beta = 0.10$. These form ten distinct configurations, including one shared center. Eight are newly evaluated and two reuse complete evaluations with the same model identities, prompts, and 23,206-item within-study holdout. The center and initial scan ranges were fixed after inspecting the preceding exploratory sweep. The $\beta = 0.12$, $\rho = 0.50$ point was added after the initial centered sweep; the reference lines remain fixed at $\beta = 0.10$, $\rho = 0.50$.

D.4 Malay Module-Study Configurations

Table 4 uses the archived module-study checkpoints with the same Llama Source, Reference, and Malay Target identities. Attention-only uses an attention budget of 0.005 and no FFN update; FFN-only uses an FFN budget of 0.1 and no attention update; joint transfer uses FFN and attention budgets of 0.1 and 0.01, respectively. These configurations were selected by calibration accuracy in the original module study. The reported scores use its common 23,710-item holdout, excluding 503 calibration items, and are computed from archived per-item predictions. These module-study results use their own archived configurations and split; they are distinct from the main Malay comparison.

D.5 Llama Malay Evaluation Protocol

Models and Scoring. The source is Llama-3.1-8B-Instruct, the reference is Llama-3.2-1B-Instruct, and the target is Malaysian-Llama-3.2-1B-Instruct-v0.1 (Mesolitica, n.d.). MalayMMLU and its five-category taxonomy follow Poh et al. (2024) and the authors' official benchmark repository. Overall accuracy pools held-out items rather than averaging the five category scores. Category accuracies are computed from archived per-item predictions using the benchmark taxonomy.

Data Split and Prompt Consistency. Questions were partitioned by deterministic prompt groups, keeping repeated prompts together. The development partition contains 1,000 groups (1,007 items), and the reported within-study holdout contains 23,206 items. Because the benchmark had been inspected previously, this holdout is not treated as a previously unseen external test set. Evaluation uses fixed-date chat prompts with the explicit date 15 September 2026; prompt hashes were checked for consistency across compared models. The merging configurations and sensitivity sweeps are specified in Appendix D.3.

E Baseline Details

E.1 MiniLLM Training Settings

We evaluate MiniLLM (Gu et al., 2024) using its no-PT configuration with the Qwen models and the fixed schedule below. The student is initialized from Qwen2.5-1.5B-Instruct in all four domains. Teachers are Qwen2.5-Math-7B-Instruct for mathematics, Qwen2.5-Coder-7B-Instruct for code, and Qwen2.5-7B-Instruct for instruction following and multilingual transfer. Training prompts come from the same domain-specific corpora used by our standard KD baseline. Reference responses are not used by this no-PT variant. The tokenizer vocabularies and token identities are checked for exact equality; padding-only model output slots are excluded from both distributions.

The implementation retains teacher-mixture sampling, per-token importance weights, length-normalized reward-to-go, clipped policy optimization, and single-step reverse-KL regularization from the official implementation. We use the pinned Microsoft LMOps MiniLLM implementation. The Qwen configuration uses chat formatting, explicit response masks, whitening on valid response positions, single-GPU rollout buffers, and the fixed training schedule in Table 6.

Table 6: Fixed MiniLLM configuration shared by the four domain runs.

Setting	Value
Unique training prompts per domain	256
Maximum prompt / response length	256 / 256 tokens
Rollout buffer / generation batch	32 / 4
PPO epochs per buffer	4
Optimization microbatch / accumulation	1 / 8
Optimizer steps per domain	128
Learning rate	5×10^{-6}
Warmup	12 steps
AdamW betas	$(0.9, 0.95)$
Weight decay	0
Gradient norm limit	1.0
Teacher-mixture coefficient	0.2
Sampling temperature / top-p	1.0 / 1.0
PPO clipping parameter	0.2
Reward scaling divisor / clipping	0.5 / 100
Discount factor	1.0
Single-step reverse-KL coefficient	1.0
Auxiliary pretraining loss	None
Random seed	42

Prompts are shuffled using the fixed seed; overlength prompts are skipped rather than truncated, and the first 256 eligible prompts are used. Each run processes at most 524,288 non-padding

input tokens during student optimization, including repeated PPO epochs. This bound applies to student optimization tokens; rollout generation is additional to this budget. Student weights and optimizer states use FP32, forward computation uses BF16 autocasting, and saved checkpoints use BF16. Gradient checkpointing is enabled. The fixed configuration is selected before these runs' test evaluations; no checkpoint or hyperparameter is selected using the resulting test scores. A four-prompt smoke run checks numerical stability and serialization and is not used as a reported baseline.

Evaluation reuses the existing prompts, splits, generation limits, and scoring implementations for each domain, together with the general-capability suite. These are within-study evaluations on previously inspected benchmarks, not new external test sets. All four training runs and their final evaluations completed before the MiniLLM results were added to the main comparison tables.

E.2 Standard KD versus MiniLLM

Let p denote the frozen source (teacher) distribution and q_θ the target (student) next-token distribution. Our standard KD baseline combines forward KL divergence, $D_{\mathrm{KL}}(p\|q_\theta)$, with next-token cross-entropy on the concatenated prompt and reference response. It uses fixed domain-matched examples and teacher predictions at their prefixes. DistiLLM (Ko et al., 2024) studies forward KL alongside alternative distillation divergences for language models.

MiniLLM instead targets reverse KL, $D_{\mathrm{KL}}(q_\theta\|p)$. The evaluated no-PT configuration samples responses from a student–teacher mixture, uses importance-weighted clipped policy updates with length-normalized reward-to-go, and includes a single-step reverse-KL term. The two KL orientations weight disagreements differently: forward KL averages under the teacher distribution, whereas reverse KL averages under the student distribution. MiniLLM does not merely reverse the KL arguments on the same fixed training examples. It changes how responses and prefixes are obtained and how gradients are estimated. This variant uses training prompts but no reference-response cross-entropy or auxiliary pretraining loss. Both KD and MiniLLM require teacher inference, text inputs, and student gradient updates. Their configurations have different token exposure and computational costs; the reported runs are not a matched-compute comparison.

E.3 Standard KD Training Settings

The archived Qwen KD runs use a teacher weight of 0.9, a token budget of 524,288, a block length of 512, gradient accumulation over eight microbatches, and seed 42. The objective is

$$\mathcal{L}_{\mathrm{KD}} = 0.9\,\vartheta^2 D_{\mathrm{KL}}(p_\vartheta\|q_\vartheta) + 0.1\,\mathcal{L}_{\mathrm{CE}},$$

where ϑ is the distillation temperature and both distributions use the exact shared semantic vocabulary. Cross-entropy covers concatenated prompt and response tokens. The registered learning-rate grid is $\{10^{-6}, 3\times10^{-6}, 10^{-5}, 3\times10^{-5}\}$ and the temperature grid is $\{1, 2\}$. Selected math, code, instruction, and multilingual runs use learning rates 10^{-5}, 3×10^{-6}, 3×10^{-6}, and 3×10^{-6}, with temperatures 1, 1, 1, and 2, respectively. Their registered corpora contain 928, 1,000, 1,002, and 939 examples; these are corpus sizes rather than claims that every example is consumed under the token budget. MiniLLM draws its 256 eligible prompts per domain from these same corpora.

E.4 Malay KD and Other Evaluated Baselines

The Malay KD run uses the same forward-KL-plus-cross-entropy objective, teacher weight 0.9, temperature 1, learning rate 10^{-5}, gradient accumulation of eight, and seed 42. Training consumes 1,024 blocks of 512 tokens from the archived Malay training corpus. Its configuration is fixed before training. Evaluation uses the same 23,206-item holdout and fixed-date prompts as the main Malay comparison.

For the Qwen FuseLLM runs (Wan et al., 2024), the two prediction providers are the domain Source and a frozen copy of the initial Target. The implementation selects a teacher distribution by minimum sequence cross-entropy, retains its top ten vocabulary entries, and uses a language modeling loss weight of 0.9. All four runs use learning rate 10^{-5} and seed 42, FP32 master weights, and BF16 forward computation. The shared semantic vocabulary is checked before combining distributions.

The Qwen Transport and Merge runs (Cui et al., 2026) use 128 calibration samples, retain 128 candidate correspondences, and set inner and outer transport regularization to 0.03 and 0.1. The registered mixing grid is $\{0.0005, 0.001, 0.0025, 0.005, 0.01, 0.02, 0.03, 0.05, 0.08, 0.1, 0.15, 0.25, 0.4, 0.6, 0.8, 1.0\}$. These runs use masked merging and no post-merging adaptation. The Malay row reports the original paper's published category scores and is excluded from within-protocol rankings in Table 3.