αTRANSFER: COEFFICIENT TRANSFER FOR EFFICIENT MODEL MERGING

Anonymous authors
Paper under double-blind review

ABSTRACT

Model merging offers a promising solution for combining multiple fine-tuned checkpoints into a single model through parameter arithmetic. However, finding optimal merging coefficients requires an extensive search that becomes prohibitively expensive as models scale in both size and number, due to high memory requirements and combinatorial growth in the search space. We show that, within the same model family, models exhibit highly congruent performance distributions over merging coefficients across different model sizes. This distributional similarity enables a practical paradigm we call α*Transfer*: searching for optimal coefficients on a small proxy model, then directly transfer them to larger target models. We verify αTransfer across multiple merging methods, model families, and tasks. Experimental results demonstrate a 6$\times$ speedup and 70% memory reduction on vision transformers, and a 20$\times$ speedup and 85% memory reduction on large language models, while maintaining comparable performance. Our findings establish αTransfer as an efficient and generalizable approach to scaling model merging.

1 INTRODUCTION

The pre-train and fine-tune paradigm, where a powerful pre-trained model is fine-tuned on task-specific data to obtain specialized expert models, has become the dominant approach in deep learning (Howard & Ruder, 2018; Radford et al., 2019; Bommasani et al., 2021; Zhuang et al., 2020). This has led to a growing ecosystem of publicly available fine-tuned checkpoints (Wolf et al., 2020). However, real-world applications increasingly demand models capable of handling multiple tasks or integrating diverse skills simultaneously. Maintaining separate fine-tuned models for each capability thus becomes impractical: it incurs significant storage and deployment overhead (Fifty et al., 2021; Zhang & Yang, 2021), and the knowledge and skills of these isolated models cannot be easily composed to serve diverse applications (Ruder, 2017; Ni et al., 2023).

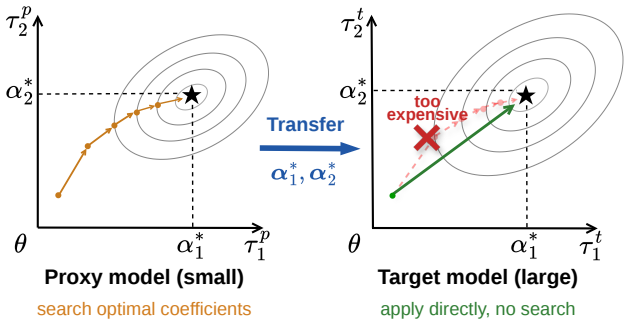


Figure 1: **Overview of αTransfer.** Axes represent task vectors (τ_1, τ_2); superscripts p, t denote proxy and target models. Contours show the performance landscape when merging with coefficients (α_1, α_2). Optimal coefficients found on a cheap proxy model transfer directly to the large target, skipping the expensive search.

Model merging has emerged as a promising solution to these limitations, directly combining multiple fine-tuned models in weight space into a single unified model capable of handling multiple tasks. Early work demonstrated the feasibility of this approach by merging task-specific models through weighted averaging (Jin et al., 2023). Merging was further reframed through the lens of task vectors, the parameter differences between fine-tuned and pre-trained models that capture task-specific knowledge (Ilharco et al., 2023). In this formulation, the merged model is obtained by adding a

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.

scaled combination of task vectors to the pre-trained model, where scaling coefficients determine each task's contribution. Subsequent works proposed various ways to process task vectors during merging, such as resolving sign conflicts (Yadav et al., 2023) or applying sparsity constraints (Yu et al., 2024; Davari & Belilovsky, 2024). Given the significant impact of scaling coefficients on merging performance, other works focus specifically on finding better coefficients, either by learning them adaptively (Yang et al., 2024; Touayouch et al., 2026) or searching over larger spaces (Akiba et al., 2025). This line of work has further extended to LLMs, demonstrating skill composition across diverse capabilities such as mathematical reasoning, code generation, and multilingual ability (Yu et al., 2024; Akiba et al., 2025; Yu et al., 2025; Huang et al., 2024).

Despite its promise, **searching for the merging coefficients** remains a central challenge that limits the scalability of model merging (He et al., 2026). Since each candidate set of coefficients requires a full model evaluation to assess, the search cost grows exponentially with the number of tasks, making exhaustive search infeasible. Existing methods address this through two broad strategies: evaluation-based search and gradient-based optimization. The first relies on approximate strategies such as assuming equal coefficients (Ilharco et al., 2023; Yadav et al., 2023; Yu et al., 2024), heuristic search (Akiba et al., 2025), or Bayesian optimization (Lee et al., 2025; Liu et al., 2025) to reduce the number of evaluations. However, each evaluation becomes increasingly costly as model size grows, and since these methods still require many evaluations to find good coefficients, the total cost can be substantial for large models. The second strategy avoids repeated evaluation by optimizing coefficients directly against a proxy objective (Yang et al., 2024; Touayouch et al., 2026), but requires computing gradients over multiple models during optimization. The memory demand thus scales with both the number of tasks and model size, rendering these methods infeasible when GPU memory is insufficient. Together, these two limitations pose a fundamental barrier to applying model merging in practical, large-scale settings.

The high computation cost of evaluation-based search and the prohibitive memory demand of gradient-based optimization together make coefficient search the primary bottleneck for applying model merging at large scale. A natural question then arises: can we perform this expensive search on a much smaller model instead? We propose α*Transfer* (see Figure 1), which searches for optimal merging coefficients on a small proxy model from the same family and directly applies them to the large target model. Our intuition is that optimal merging coefficients encode both the correlations among tasks and their relative importance, properties largely determined by the tasks themselves and the model family rather than by model scale, and are thus transferable across scales within the same family. We substantially reduce both computation and memory cost by performing coefficient search on a significantly smaller proxy model, making model merging practical even at large scale.

We validate αTransfer through extensive experiments on vision and language models, covering six merging methods (Task Arithmetic (Ilharco et al., 2023), TIES (Yadav et al., 2023), DARE (Yu et al., 2024), AdaMerging (Yang et al., 2024), AdaMerging++ (Yang et al., 2024), and DivMerge (Touayouch et al., 2026)) across CLIP-ViT (Radford et al., 2021), SigLIP (Zhai et al., 2023) , and Qwen3 (Yang et al., 2025) model families. Empirical analysis shows that models within the same family exhibit highly correlated performance landscapes across different coefficient combinations, supporting the feasibility of coefficient transfer. In many cases, transferred coefficients match direct search on the target model with negligible accuracy loss, while achieving up to **6**$\times$ speedup for search-based methods and **70%** memory reduction for gradient-based methods on vision transformers, directly addressing the two bottlenecks identified above, and up to **20**$\times$ speedup and **85%** memory reduction on large language models.

2 Related work

2.1 Coefficient search in model merging

While model merging encompasses a broad range of techniques (Yang et al., 2026), we focus on the core challenge of determining scaling coefficients. We systematically categorize coefficient search into two paradigms: evaluation-based and gradient-based methods, each constrained by distinct bottlenecks in computation and memory, respectively.

Evaluation-based methods. These methods search the coefficient space by repeatedly evaluating candidate merged models. The simplest variants use uniform coefficients across tasks (Ilharco et al.,

2023; Yadav et al., 2023; Yu et al., 2024), which is cheap but often suboptimal, as per-task or per-layer adjustment substantially improves performance (Yang et al., 2024). More sophisticated approaches reduce the number of evaluations through evolutionary search (Akiba et al., 2025) or Bayesian optimization (Lee et al., 2025; Liu et al., 2025). Despite these advances, the cost per evaluation grows with model size, making the total compute budget substantial at scale.

Gradient-based methods. These methods instead optimize coefficients directly against a proxy objective. AdaMerging (Yang et al., 2024) and AdaMerging++ minimize prediction entropy on unlabeled test data, while DivMerge (Touayouch et al., 2026) uses Jensen-Shannon divergence between merged and individual fine-tuned models. These methods avoid repeated evaluation, but require holding multiple fine-tuned models in memory simultaneously, with memory cost scaling jointly with task count and model size.

Across both categories, all existing approaches conduct the search on the target model itself. We ask whether the search can instead be conducted on a smaller proxy model, enabling existing methods to be applied at scales where they would otherwise be impractical.

2.2 Searching on small proxies for large targets

Searching on a small proxy and applying the result to a large target has appeared in several settings outside model merging. μTransfer (Yang et al., 2021) transfers training hyperparameters such as learning rate from a small model to a large one via the μP parameterization. DoReMi (Xie et al., 2023) uses a 280M proxy to determine pretraining data mixture weights for training an 8B model. Khaddaj et al. (2025) characterize how training data influences model behavior consistently across scale, enabling proxy-based data attribution and dataset selection. These works share a common philosophy with ours: quantities optimized on a small model can remain meaningful when transferred to a larger one. However, to our knowledge, this paradigm has not been investigated in model merging. We are the first to show that merging coefficients exhibit such cross-scale transferability within model families, across architectures and merging methods.

3 αTRANSFER

3.1 Problem formulation

Consider a model merging problem where we are given a pre-trained model and T task-specific fine-tuned models derived from it. The goal is to combine these models into a single unified model that performs well across all T tasks, without access to the original training data.

Formally, let $\theta_0 \in \mathbb{R}^d$ denote the parameters of the pre-trained model, and let $\theta_1, \ldots, \theta_T$ denote the parameters of the T fine-tuned models. The *task vector* for task i is defined as the parameter difference between the fine-tuned and pre-trained models:

$$\tau_i = \theta_i - \theta_0. \tag{1}$$

Unified formulation. While existing model merging methods appear diverse in their mechanisms, we observe that they share a common underlying structure: they edit task vectors through various strategies (trimming, masking, etc.), then combine them with learnable or searched coefficients. This insight motivates us to propose a unified formulation that captures this shared structure, enabling systematic analysis and comparison across methods (see Table 1). We express a general family of model merging methods as:

$$\theta_{\text{merged}} = \mathcal{M}(\theta_0, \{\tau_i\}_{i=1}^{T}; \boldsymbol{\alpha}), \tag{2}$$

where $\mathcal{M}$ is a merging operator that encapsulates both the editing strategy and the combination mechanism, and $\boldsymbol{\alpha}$ denotes the merging coefficients. The key intuition is that most merging methods operate in two conceptual stages: an editing stage where a function $f(\cdot)$ (embedded within $\mathcal{M}$) transforms the task vectors into edited representations $\tilde{\tau}$, followed by a weighting stage where $\boldsymbol{\alpha}$ scales the edited representations and adds them back to θ_0. The granularity at which $\boldsymbol{\alpha}$ is defined gives rise to three common settings:

Table 1: **Representative model merging methods under our unified formulation.** Methods are characterized along three dimensions: editing function f, coefficient search strategy, and coefficient granularity.

Method	The Editing Function f	Search Strategy	Granularity
Task Arithmetic (Ilharco et al., 2023)	Identity	Grid search	Global
TIES (Yadav et al., 2023)	Trim + sign election	Grid search	Global
DARE (Yu et al., 2024)	Random masking + rescaling	Grid search	Global
AdaMerging (Yang et al., 2024)	Identity	Gradient-based	Task-wise/Layer-wise
AdaMerging++ (Yang et al., 2024)	Trim + sign election	Gradient-based	Task-wise/Layer-wise
DivMerge (Touayouch et al., 2026)	Identity	Gradient-based	Task-wise

- **Global:** The merging operator produces a single edited task vector $\tilde{\tau} = f(\{\tau_i\}_{i=1}^T)$, where $f : \mathbb{R}^{T\times d} \to \mathbb{R}^d$. A single scalar $\alpha \in \mathbb{R}$ is shared across all tasks:

$$\theta_{\text{merged}} = \theta_0 + \alpha \cdot \tilde{\tau}. \tag{3}$$

- **Task-wise:** The merging operator produces task-specific edited representations $\tilde{\tau}_i = f_i(\{\tau_j\}_{j=1}^T)$, where $f_i : \mathbb{R}^{T\times d} \to \mathbb{R}^d$ may leverage all task vectors to compute the representation for task i. A distinct scalar $\alpha_i \in \mathbb{R}$ is assigned to each task:

$$\theta_{\text{merged}} = \theta_0 + \sum_{i=1}^{T} \alpha_i \cdot \tilde{\tau}_i. \tag{4}$$

- **Layer-wise:** The merging operator produces task-specific and layer-specific edited representations $\tilde{\tau}_{i,l} = f_{i,l}(\{\tau_{j,l}\}_{j=1}^T)$, where $f_{i,l} : \mathbb{R}^{T\times d_l} \to \mathbb{R}^{d_l}$ may leverage all task vectors at layer l to compute the representation for task i. A distinct scalar $\alpha_{i,l} \in \mathbb{R}$ is assigned to each layer of each task, and the parameters at each layer l are merged independently:

$$\theta^{(l)}_{\text{merged}} = \theta^{(l)}_0 + \sum_{i=1}^{T} \alpha_{i,l} \cdot \tilde{\tau}_{i,l}, \quad \forall l = 1, \ldots, L. \tag{5}$$

3.2 THE αTRANSFER MECHANISM

We propose αTransfer, a simple yet efficient approach for obtaining merging coefficients for large target models by leveraging small proxy models from the same family. The core insight driving this approach is that merging coefficients encode the relative importance of tasks and their interactions, which are primarily determined by the task characteristics rather than model scale. Consequently, coefficients learned on a small proxy model can be transferred to larger target models within the same family, significantly reducing the computational bottleneck of coefficient search.

Global and task-wise coefficient transfer. Given a target model θ_0^t and a proxy model θ_0^p from the same family, along with their respective fine-tuned models $\{\theta_i^t\}_{i=1}^T$ and $\{\theta_i^p\}_{i=1}^T$, coefficient transfer proceeds in two steps. First, we apply any existing merging method to the proxy model to obtain the merging coefficients:

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \mathcal{L}^p \left(\theta_0^p + \sum_{i=1}^{T} \alpha_i \cdot \tilde{\tau}_i^p \right), \tag{6}$$

where $\tilde{\tau}_i^p = f(\tau_i^p)$ are the edited task vectors with $\tau_i^p = \theta_i^p - \theta_0^p$, and $\mathcal{L}^p$ denotes the objective used by the chosen merging method on the proxy. For global methods, all α_i are tied to a single scalar α. Note that the shape of $\boldsymbol{\alpha}^*$ depends only on the number of tasks T (or is a single scalar for global methods), not on model size. We can therefore directly apply this same $\boldsymbol{\alpha}^*$ to merge the target model:

$$\theta^t_{\text{merged}} = \theta_0^t + \sum_{i=1}^{T} \alpha_i^* \cdot \tilde{\tau}_i^t, \tag{7}$$

where $\tilde{\tau}_i^t = f(\tau_i^t)$ with $\tau_i^t = \theta_i^t - \theta_0^t$. Note that only the coefficients $\boldsymbol{\alpha}^*$ are transferred from the proxy; the editing function $f(\cdot)$ is still applied natively to the target model's task vectors.

Layer-wise coefficient transfer. In layer-wise merging, each parameter layer is assigned its own merging coefficient. Since proxy and target models often differ in depth (and thus total number of layers), a direct one-to-one coefficient transfer is not always possible.

Modern architectures such as transformers consist of repeating blocks, where certain modules appear in every block (e.g., attention, feed-forward) while others do not (e.g., embeddings, output layers). For layers that do not repeat across blocks, coefficients are directly transferred as they appear in the same positions in both models. For layers that repeat across blocks, recent work has shown that layers at similar relative depths follow a shared progression of representations across depth (Wolfram & Schein, 2025), suggesting that their merging coefficients might also be similar. Based on this intuition, we propose two strategies for transferring these coefficients.

Copy-based transfer. For a layer in block $b \in \{1, \ldots, B_t\}$ of the target model, we directly copy the coefficient from the corresponding layer type in the proxy block at the matched relative depth:

$$b_{\text{proxy}} = \left\lceil \frac{b}{B_t} \cdot B_p \right\rceil, \tag{8}$$

where B_p and B_t are the number of blocks in the proxy and target models. The ceiling ensures that target blocks 1 and B_t map to proxy blocks 1 and B_p, respectively.

Interpolation-based transfer. To obtain smoother coefficient transitions, we linearly interpolate between the coefficients from the two nearest proxy blocks based on the target block's continuous relative position.

4 Experiments

Experimental Setup. We evaluate αTransfer across three model families: CLIP-ViT (Radford et al., 2021) and SigLIP (Zhai et al., 2023) for vision, and Qwen3 (Yang et al., 2025) for language. Our evaluation covers 8 vision tasks (DTD (Cimpoi et al., 2014), EuroSAT (Helber et al., 2018), FER2013 (Goodfellow et al., 2013), Food101 (Bossard et al., 2014), GTSRB (Stallkamp et al., 2011), RESISC45 (Cheng et al., 2017), Stanford Cars (Krause et al., 2013), SUN397 (Xiao et al., 2016)) and 4 language tasks (Usefulness Judge (Chen et al., 2026), IFEval (Zhou et al., 2023), Banking77 (Loukas et al., 2023), DDXPlus (Fansi Tchango et al., 2022)). For each task, we fine-tune the corresponding base model to obtain an expert checkpoint. All experiments were conducted on a single NVIDIA RTX 4000 SFF Ada GPU, except for the Qwen3-4B model, which required two GPUs. Training and implementation details are provided in Appendix A.

4.1 Empirical analysis of coefficient transferability

We first investigate whether the performance landscapes over the merging coefficient space exhibit structural similarity across model scales. This perspective is crucial: if the overall landscapes align, optimization algorithms will navigate comparable objective surfaces and converge to corresponding regions, making coefficient transfer viable even when the absolute global optimum is not found.

To build intuition, we visualize the performance landscape of merged models under different coefficient combinations. For a given pair of tasks, we vary the two corresponding coefficients over a grid and record the average accuracy. Figure 2 shows these landscapes for two representative task pairs on SigLIP-Base and SigLIP-Large. Despite the difference in model scale, the two models exhibit highly similar performance distributions over the coefficient space. This observation suggests that the overall topography of the coefficient space remains relatively stable across scales.

To rigorously verify this beyond specific task pairs, we extend our analysis to all three model families in a many-task setting. By sampling multiple coefficient combinations, we compute the Spearman rank correlation (ρ) between the proxy and target mean accuracies (averaged over the 8 vision or 4 language tasks, respectively). As detailed in Table 2, all evaluated pairs exhibit consistently strong correlations (ρ up to 0.94). This confirms that *coefficient transferability* is a widespread and robust property in multi-task model merging, providing a solid empirical foundation for αTransfer.

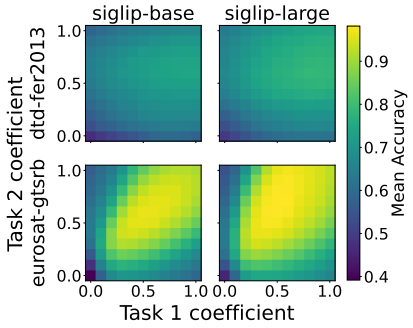


Figure 2: **Performance landscapes across model scales.** We visualize mean accuracy over a grid of merging coefficients. The heatmaps show highly similar distributions, indicating that the overall shape of the performance landscape is preserved across different model sizes.

Table 2: **Performance correlation between proxy and target models.** We sample multiple coefficient combinations, construct merged models accordingly, and compute the Spearman rank correlation (ρ) between their accuracies on the proxy and target models. The high correlations confirm that the relative performance of different coefficient combinations remains highly consistent across model sizes.

Proxy	Target	Spearman's ρ
ViT-B/32	ViT-L/14	0.58
ViT-B/16	ViT-L/14	0.84
SigLIP-B	SigLIP-L	0.79
Qwen3-0.6B	Qwen3-1.7B	0.94
Qwen3-0.6B	Qwen3-4B	0.90

4.2 Main results

4.2.1 Vision models

We evaluate αTransfer across six merging methods on the CLIP-ViT and SigLIP families across three proxy-target pairs: ViT-B/32 $\rightarrow$ ViT-L/14, ViT-B/16 $\rightarrow$ ViT-L/14, and SigLIP-Base $\rightarrow$ SigLIP-Large. For each method, we report the average accuracy under three settings: coefficients searched directly on the target model (origin), coefficients transferred from a proxy model (transfer), and simple averaging (SA) without any coefficient search, along with the computation time and peak memory cost of coefficient search.

As shown in Table 3, αTransfer achieves performance close to direct search across most methods and proxy-target pairs, while requiring only a fraction of the computational resources. In many cases the accuracy gap is negligible: on SigLIP-Base $\rightarrow$ SigLIP-Large, TIES and DARE drop by only 0.11% and 0.32%; on ViT-B/32 $\rightarrow$ ViT-L/14, Task Arithmetic and DARE drop by only 0.43% and 0.49%. Even in the most challenging cases (e.g., AdaMerging++ on ViT-B/32 $\rightarrow$ ViT-L/14, -7.02%), the transferred model still surpasses the simple average baseline by 7.12%, retaining most of the benefit of direct search.

The efficiency gains reflect the distinct resource bottlenecks of each method type. For evaluation-based searching methods, the bottleneck is computation, where a single run already exceeds 10 hours on the target model, and αTransfer reduces this by up to **6.07**$\times$ (Task Arithmetic, ViT-B/32 $\rightarrow$ ViT-L/14), bringing the search time to under 2 hours. For gradient-based methods, the bottleneck is GPU memory: running optimization on the proxy model allows a batch size of 16, whereas the target model requires reducing to 4 to avoid out-of-memory errors. Coefficient transfer eliminates this constraint, reducing GPU memory consumption by up to **70.4%** (AdaMerging/AdaMerging++, ViT-B/32 $\rightarrow$ ViT-L/14). Together, these results show that coefficient transfer enables efficient merging on large target models at a fraction of the cost, with minimal sacrifice in accuracy.

4.2.2 Large language models

To investigate coefficient transferability in LLMs, we evaluate Qwen3 models across two pairs: 0.6B $\rightarrow$ 1.7B and 0.6B $\rightarrow$ 4B. For TIES, we observe the DDXPlus expert is often filtered out due to conflicting update directions. To address this, we adopt a TIES-Pairwise variant: merging Usefulness Judge and IFEval in the first stage, then merging the result with Banking77 and DDXPlus in the second stage, using a consistent α throughout. This approach recovers DDXPlus performance while keeping the search space identical to standard TIES.

As shown in Table 4, αTransfer yields performance that closely tracks the exhaustive search baseline. While the accuracy gap on the 1.7B model is within a marginal 2%, αTransfer achieves zero

Table 3: **Coefficient transfer on vision models.** We compare direct search (Origin), αTransfer (Transfer) and simple average baseline (SA) across different proxy-target pairs. Transfer maintains accuracy comparable to the origin methods while significantly mitigating resource bottlenecks, yielding up to a 6.07× speedup and a 70.4% memory reduction.

Proxy → Target	Method	Acc (%)			Time (h)			Memory (GB)		
		Origin	Transfer	SA	Origin	Transfer	Speedup	Origin	Transfer	Reduction
ViT-B/32 → ViT-L/14	Task Arithmetic	75.98	75.55	72.89	11.43	1.88	6.07×	2.12	0.67	68.7%
	TIES	75.17	71.65	64.96	12.12	2.05	5.92×	2.12	0.67	68.7%
	DARE	76.05	75.56	62.70	13.01	2.26	5.75×	2.12	0.67	68.7%
	AdaMerging	75.67	71.83	72.89	1.69	0.72	2.35×	17.84	5.27	70.4%
	AdaMerging++	79.10	72.08	64.96	1.46	0.83	1.77×	17.84	5.27	70.4%
	DivMerge	78.77	74.73	72.89	0.33	0.15	2.14×	17.84	5.27	70.4%
ViT-B/16 → ViT-L/14	Task Arithmetic	75.98	72.89	72.89	11.43	3.24	3.52×	2.12	0.87	58.8%
	TIES	75.17	69.53	64.96	12.12	3.36	3.61×	2.12	0.87	58.8%
	DARE	76.05	72.91	62.70	13.01	3.64	3.58×	2.12	0.87	58.8%
	AdaMerging	75.67	72.63	72.89	1.69	0.94	1.79×	17.84	6.87	61.5%
	AdaMerging++	79.10	74.82	64.96	1.46	0.84	1.75×	17.84	6.87	61.5%
	DivMerge	78.77	75.58	72.89	0.33	0.13	2.52×	17.84	6.87	61.5%
SigLIP-Base → SigLIP-Large	Task Arithmetic	80.22	78.83	75.05	10.40	3.46	3.01×	2.84	1.07	62.3%
	TIES	77.50	77.39	66.95	11.75	3.90	3.01×	2.84	1.07	62.3%
	DARE	79.84	79.52	64.79	13.41	4.00	3.35×	2.84	1.07	62.3%
	AdaMerging	81.48	80.27	75.05	1.58	1.02	1.54×	18.12	7.41	59.1%
	AdaMerging++	81.88	77.83	66.95	1.63	1.02	1.59×	18.12	7.41	59.1%
	DivMerge	82.98	80.01	75.05	0.32	0.14	2.33×	18.12	7.41	59.1%

Table 4: **Coefficient transfer on Qwen3 LLMs.** We compare direct search (Origin), coefficient transfer (Transfer) and simple average baseline (SA) across different proxy-target pairs. Transfer maintains accuracy comparable to the origin methods while significantly mitigating resource bottlenecks, yielding up to a 19.97× speedup and an 85.2% memory reduction. We include a TIES-Pairwise variant to address the poor performance of the original TIES method.

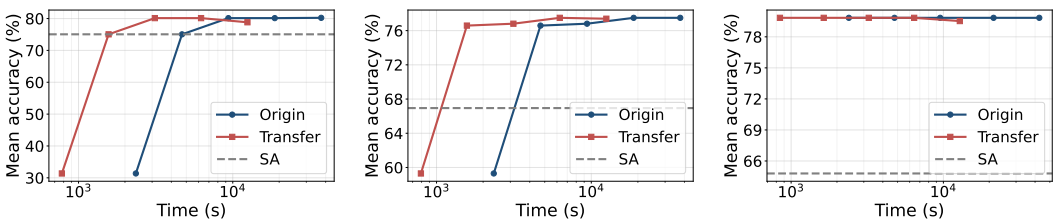
Proxy → Target	Method	Acc (%)			Time (h)			Memory (GB)		
		Origin	Transfer	SA	Origin	Transfer	Speedup	Origin	Transfer	Reduction
0.6B → 1.7B	Task Arithmetic	76.44	74.63	63.85	0.78	0.07	11.00×	17.21	5.96	65.4%
	TIES	58.35	58.26	32.65	1.86	0.10	18.96×	17.21	5.96	65.4%
	TIES-Pairwise	71.15	70.37	44.01	1.55	0.08	19.94×	10.32	3.58	65.3%
0.6B → 4B	Task Arithmetic	76.03	76.03	71.91	1.12	0.10	11.03×	40.22	5.96	85.2%
	TIES	66.04	65.26	41.04	3.80	0.20	18.98×	40.22	5.96	85.2%
	TIES-Pairwise	73.69	73.69	59.57	4.00	0.20	19.97×	24.13	3.58	85.2%

performance degradation for both Task Arithmetic and TIES-Pairwise on the 4B scale. More importantly, our method delivers a 20× speedup and an 85% memory reduction for the 4B model, compressing the memory requirement from over 40GB to under 6GB. These results demonstrate that αTransfer effectively bypasses the prohibitive cost of autoregressive evaluation, making the merging of large-scale models both accessible and scalable.

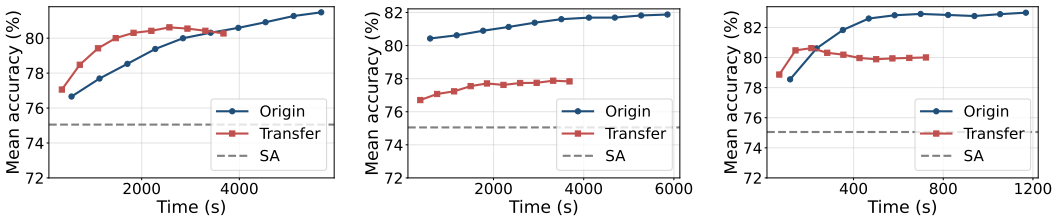
4.3 Constrained budget analysis

Beyond final accuracy, we examine whether coefficient transfer can match or exceed the performance of direct search given the same amount of computation spent on the target model.

Evaluation-based methods. We simulate increasing evaluation budgets by progressively refining the search granularity. At budget level k, we evaluate 2^k uniformly spaced points in the coefficient space ($k = 0, 1, \ldots, 4$), and report the best accuracy found so far. For transfer, we directly apply the proxy-optimal coefficient to the target model without any target-side evaluation. As shown in the top row of Figure 3, transferred coefficients achieve strong performance even at low budgets across all three methods. For Task Arithmetic and TIES, the origin curve requires substantially more evaluations to reach comparable accuracy. For DARE, both curves remain nearly flat, with transfer matching origin performance and substantially outperforming the simple average.



(a) Task Arithmetic (b) TIES (c) DARE



(d) AdaMerging (e) AdaMerging++ (f) DivMerge

Figure 3: **Budget analysis on SigLIP models.** We compare the best achievable accuracy under varying computational time budgets for direct search (Origin) and αTransfer (Transfer), with simple average (SA) as a static baseline. Transfer shifts the performance curves significantly to the left, reaching comparable accuracy in a fraction of the time for most methods. The performance gap in AdaMerging++ is further investigated in Appendix C.1, with full results provided in Appendix C.

Gradient-based methods. For gradient-based methods (AdaMerging, AdaMerging++, DivMerge), we independently optimize coefficients on the proxy and target models, evaluating all intermediate proxy coefficients directly on the target. As shown in the bottom row of Figure 3, αTransfer for AdaMerging and DivMerge converges faster initially, achieving competitive or higher accuracy than direct target optimization under limited budgets. While AdaMerging++ shows a performance gap, we find that it stems from overly high sparsity: lower sparsity slightly reduces in-model performance but substantially improves coefficient transferability Appendix C.1.

4.4 Layer-wise coefficient transfer

We evaluate the layer-wise coefficient transfer (detailed in Section 3.2) on AdaMerging and AdaMerging++ across three proxy-target pairs (Table 5). Both copy-based and interpolation mapping variants yield comparable performance, with the accuracy gap between the two variants remaining consistently below 0.5%. Notably, on ViT-B/16 $\rightarrow$ ViT-L/14 and SigLIP-Base $\rightarrow$ SigLIP-Large, layer-wise transfer matches or even outperforms direct target-model optimization. Remarkably, this higher accuracy is achieved while simultaneously reducing computation time by 1.53$\times$ to 1.71$\times$ and memory footprint by 59.1% to 61.5%.

The only exception is ViT-B/32 $\rightarrow$ ViT-L/14, where transfer underperforms direct search by 1–3%. This slight gap aligns with the lower coefficient correlation observed for this pair (Table 2, $\rho = 0.58$ vs. 0.79–0.84). We hypothesize that the larger architectural gap between the patch size of proxy model and target model also plays a role; specifically, the difference in input resolution may alter the optimization landscape, making direct coefficient mapping less effective.

4.5 Extending αTransfer via hybrid search

While αTransfer provides massive efficiency gains through direct coefficient mapping, we explore a hybrid extension for scenarios where maximizing absolute performance is prioritized over extreme efficiency. In this setting, rather than treating the transferred coefficients as the final solution, we leverage them as a high-quality initialization, allocating a small fraction of the computational budget to the target model for refinement. Specifically, for evaluation-based methods, we conduct a coarse search on the proxy followed by a localized refinement on the target; for gradient-based methods,

Table 5: **Layer-wise coefficient transfer.** We compare direct search (Origin) against two relative-depth mapping variants: Copy and Interpolation (Interp). Remarkably, layer-wise transfer frequently outperforms direct search—achieving higher accuracy on the target model while simultaneously reducing computational time and memory footprint.

Proxy → Target	Method	Acc (%)			Time Speedup	Memory Reduction
		Origin	Copy	Interp		
ViT-B/32 → ViT-L/14	AdaMerging	**80.57**	77.85	78.21	2.17×	70.4%
	AdaMerging++	**82.80**	81.35	81.55	2.17×	70.4%
ViT-B/16 → ViT-L/14	AdaMerging	80.57	**81.21**	81.13	1.68×	61.5%
	AdaMerging++	82.80	**83.15**	83.07	1.71×	61.5%
SigLIP-Base → SigLIP-Large	AdaMerging	83.57	**84.25**	84.21	1.53×	59.1%
	AdaMerging++	83.13	83.55	**83.57**	1.61×	59.1%

Table 6: **Hybrid search results.** We compare direct search (Origin), pure coefficient transfer (Transfer), and hybrid search (Hybrid) that refines transferred coefficients on the target model. Hybrid search effectively recovers accuracy drops from pure transfer—frequently matching or even outperforming direct search—while still maintaining a steady computational speedup.

Proxy → Target	Method	Acc (%)			Speedup	
		Origin	Transfer	Hybrid	Transfer	Hybrid
ViT-B/32 → ViT-L/14	Task Arithmetic	**75.98**	75.55	**75.98**	6.07×	1.86×
	AdaMerging	75.67	71.83	**76.00**	2.35×	1.40×
	DivMerge	**78.77**	74.73	78.51	2.14×	1.38×
ViT-B/16 → ViT-L/14	Task Arithmetic	**75.98**	72.89	**75.98**	3.52×	1.68×
	AdaMerging	75.67	72.63	**76.41**	1.79×	1.28×
	DivMerge	78.77	75.58	**78.78**	2.52×	1.29×
SigLIP-Base → SigLIP-Large	Task Arithmetic	**80.22**	78.83	78.96	3.01×	1.46×
	AdaMerging	81.48	80.27	**81.59**	1.54×	1.21×
	DivMerge	**82.98**	80.01	82.92	2.33×	1.20×

we perform half of the optimization steps on the proxy and complete the remaining steps directly on the target using the transferred coefficients as the starting point.

As detailed in Table 6, this hybrid approach effectively closes any marginal accuracy gaps associated with direct transfer across the evaluated proxy-target pairs. More importantly, it consistently matches—and frequently exceeds—the accuracy of exhaustive target optimization, while still delivering a 1.20× to 1.86× computational speedup. On LLMs, hybrid search similarly recovers the original merging performance while achieving 2.29× to 4.35× speedups (Table 8). Overall, these results show that αTransfer serves not only as a zero-cost merging technique, but also as an effective accelerator for full-scale merging, offering a flexible trade-off between efficiency and performance.

5 Conclusion

We introduced αTransfer, a highly efficient framework that decouples the cost of coefficient search from target model size. By optimizing coefficients on a small proxy and transferring them via direct copying or relative-depth mapping, we achieve accuracy comparable to—and sometimes surpassing—exhaustive direct search. Across six merging methods, this approach yields a 6× speedup and 70% memory reduction on vision transformers, and up to a 20× speedup and 85% memory reduction on large language models.

These findings suggest that optimal merging coefficients are primarily properties of the tasks and model family, rather than scale. This insight opens exciting directions for future work, such as investigating the transferability of other merging hyperparameters and exploring cross-recipe or cross-architecture coefficient mapping. Ultimately, we hope αTransfer serves as a crucial step toward making model merging practical and highly scalable for modern foundation models.

AI USE STATEMENT

In this work, we used generative AI tools for designing or providing feedback on experiments, implementing methods, literature survey. We have not used generative AI tools to develop the main idea, the conceptual framework, support qualitative and thematic data analysis, design or provide feedback on research methodology, and interpret results. Helping develop theoretical models, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, generating synthetic data sets, proposing or refining hypotheses, assisting with translation, and cleaning and reformatting datasets are not applicable to this work. Additionally, we used generative AI tools to make the sentences more fluent and for grammar checking. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness by 2 authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Experiment details are clearly described in the main paper and Appendix A to reproduce the empirical results. The datasets used are all publicly available.

REFERENCES

Takuya Akiba, Makoto Shing, Yujin Tang, Qi Sun, and David Ha. Evolutionary optimization of model merging recipes. *Nature Machine Intelligence*, 7(2):195–204, 2025.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *European conference on computer vision*, pp. 446–461. Springer, 2014.

Yen-Shan Chen, Zhi Rui Tam, Cheng-Kuang Wu, and Yun-Nung Chen. Expected harm: Rethinking safety evaluation of (mis) aligned llms. *arXiv preprint arXiv:2602.01600*, 2026.

Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3606–3613, 2014.

MohammadReza Davari and Eugene Belilovsky. Model breadcrumbs: Scaling multi-task model merging with sparse masks. In *European Conference on Computer Vision*, pp. 270–287. Springer, 2024.

Arsene Fansi Tchango, Rishab Goel, Zhi Wen, Julien Martel, and Joumana Ghosn. Ddxplus: A new dataset for automatic medical diagnosis. *Advances in neural information processing systems*, 35: 31306–31318, 2022.

Chris Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. Efficiently identifying task groupings for multi-task learning. *Advances in Neural Information Processing Systems*, 34:27503–27516, 2021.

Ian J Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, et al. Challenges in representation learning: A report on three machine learning contests. In *International conference on neural information processing*, pp. 117–124. Springer, 2013.

Yifei He, Siqi Zeng, Yuzheng Hu, Rui Yang, Tong Zhang, and Han Zhao. Mergebench: A benchmark for merging domain-specialized LLMs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2026. URL https://openreview.net/forum?id=rw50iUoyLu.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Introducing eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. In *IGARSS 2018-2018 IEEE international geoscience and remote sensing symposium*, pp. 204–207. IEEE, 2018.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 328–339, 2018.

Shih-Cheng Huang, Pin-Zu Li, Yu-Chi Hsu, Kuang-Ming Chen, Yu Tung Lin, Shih-Kai Hsiao, Richard Tsai, and Hung-Yi Lee. Chat vector: A simple approach to equip llms with instruction following and model alignment in new languages. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10943–10959, 2024.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, 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.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=FCnohuR6AnM.

Alaa Khaddaj, Logan Engstrom, and Aleksander Madry. Small-to-large generalization: Training data influences models consistently across scale. In *The Thirteenth International Conference on Learning Representations*, 2025.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pp. 554–561, 2013.

Sanwoo Lee, Jiahao Liu, Qifan Wang, Jingang Wang, Xunliang Cai, and Yunfang Wu. Dynamic fisher-weighted model merging via bayesian optimization. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4923–4935, 2025.

Deyuan Liu, Zecheng Wang, Bingning Wang, Weipeng Chen, Chunshan Li, Zhiying Tu, Dianhui Chu, and Dianbo Sui. Maximizing intermediate checkpoint value in LLM pretraining with bayesian optimization. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=UvwWrUV1JV.

Lefteris Loukas, Ilias Stogiannidis, Odysseas Diamantopoulos, Prodromos Malakasiotis, and Stavros Vassos. Making llms worth every penny: Resource-limited text classification in banking. In *Proceedings of the Fourth ACM International Conference on AI in Finance*, pp. 392–400, 2023.

Jingwei Ni, Zhijing Jin, Qian Wang, Mrinmaya Sachan, and Markus Leippold. When does aggregating multiple skills with multi-task learning work? a case study in financial nlp. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7465–7488, 2023.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: a multi-class classification competition. In *The 2011 international joint conference on neural networks*, pp. 1453–1460. IEEE, 2011.

Brahim Touayouch, Loïc Fosse, Géraldine Damnati, and Gwénolé Lecorvé. Divmerge: A divergence-based model merging method for multi-tasking. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7157–7180, 2026.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45, 2020.

Christopher Wolfram and Aaron Schein. Layers at similar depths generate similar activations across LLM architectures. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=8wKec6faAT.

Jianxiong Xiao, Krista A Ehinger, James Hays, Antonio Torralba, and Aude Oliva. Sun database: Exploring a large collection of scene categories. *International Journal of Computer Vision*, 119 (1):3–22, 2016.

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.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in neural information processing systems*, 36:7093–7115, 2023.

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.

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://openreview.net/forum?id=nZP6NgD3QY.

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):1–41, 2026.

Ge Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning large neural networks via zero-shot hyperparameter transfer. *Advances in Neural Information Processing Systems*, 34:17084–17097, 2021.

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 *Forty-first International Conference on Machine Learning*, 2024.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Extend model merging from fine-tuned to pre-trained large language models via weight disentanglement, 2025. URL https://openreview.net/forum?id=2pvMZKGYDR.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 11975–11986, 2023.

Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE transactions on knowledge and data engineering*, 34(12):5586–5609, 2021.

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.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1): 43–76, 2020.

APPENDIX

Table of Contents

A Training details **14**
- A.1 Vision models 14
- A.2 Large language models 14

B Evaluation details **15**
- B.1 Model merging hyperparameters 15

C Efficiency analysis complete results **16**
- C.1 Effect of k in AdaMerging++ 16

D Hybrid search results for LLMs **17**

E Per-task accuracy breakdown **18**
- E.1 Vision models 18
- E.2 Large language models 18

A TRAINING DETAILS

In this section, we provide the training details for the models used in our experiments. We separate the discussion into vision models (Appendix A.1) and large language models (Appendix A.2).

A.1 VISION MODELS

We adopt the pre-trained CLIP vision encoders released by OpenAI, specifically ViT-B/32, ViT-B/16, and ViT-L/14, as well as the SigLIP vision encoders released by Google (SigLIP-Base/16-256 and SigLIP-Large/16-256). Each model is fine-tuned independently on eight downstream image classification tasks: DTD, EuroSAT, FER2013, Food101, GTSRB, RESISC45, Stanford Cars, and SUN397.

Hyperparameter search. For each (model, dataset) pair, we perform a grid search over the following hyperparameters:

- Batch size: $\{32, 48\}$
- Number of epochs: $\{5, 8, 10, 20\}$
- Learning rate: $\{1\text{e}{-5},\ 3\text{e}{-5}\}$

We train each configuration for up to the maximum number of epochs specified in the search space, evaluate the model on the validation set at the end of every epoch, and select the checkpoint that achieves the highest validation accuracy. The chosen hyperparameters used for each (model, dataset) combination are reported in Table 7. Notably, ViT-B/32 and ViT-L/14 share nearly identical optimal hyperparameters across all tasks (differing only on GTSRB), while the optimal configurations for ViT-B/16 and the SigLIP variants vary more substantially across datasets.

Optimization. All vision models are fine-tuned with the AdamW optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$) and a linear learning-rate scheduler with a warmup ratio of 0.1.

A.2 LARGE LANGUAGE MODELS

We fine-tune Qwen3 models at three scales (0.6B, 1.7B, and 4B parameters) on four downstream tasks: Banking77, DDXPlus, instruction following, and usefulness judging.

Table 7: Selected hyperparameters for each vision model and dataset. Each cell reports the number of epochs (E), batch size (B), and learning rate (LR) chosen by the validation-accuracy criterion described in Appendix A.1.

	ViT-B/32			ViT-B/16			ViT-L/14			SigLIP-B			SigLIP-L		
Dataset	E	B	LR	E	B	LR	E	B	LR	E	B	LR	E	B	LR
DTD	10	32	1e-5	5	32	1e-5	10	32	1e-5	5	48	3e-5	5	48	3e-5
EuroSAT	10	32	1e-5	10	48	3e-5	10	32	1e-5	10	48	3e-5	10	48	3e-5
FER2013	10	32	1e-5	5	32	1e-5	10	32	1e-5	20	48	3e-5	20	48	3e-5
Food101	10	32	1e-5	5	32	1e-5	10	32	1e-5	20	48	3e-5	10	48	1e-5
GTSRB	10	48	3e-5	5	32	1e-5	10	32	1e-5	5	48	3e-5	5	48	3e-5
RESISC45	10	32	1e-5	8	32	3e-5	10	32	1e-5	5	48	3e-5	10	48	1e-5
Stanford Cars	10	32	1e-5	10	48	3e-5	10	32	1e-5	20	48	3e-5	10	48	1e-5
SUN397	10	32	1e-5	5	32	1e-5	10	32	1e-5	20	48	3e-5	10	48	1e-5

Training data. For Banking77 and DDXPlus, we fine-tune directly on the training split of the corresponding dataset. For instruction following, we train on a 15,000-example subsample drawn from the Nemotron-SFT-Instruction-Following-Chat-v2 dataset[1]. For the usefulness-judging task, we train on the usefulness-judge dataset[2] proposed by Chen et al. (2026).

Hyperparameter search. For each (model, task) pair, we perform a grid search over batch sizes $\{36, 48\}$ and learning rates $\{1\mathrm{e}{-5},\ 4\mathrm{e}{-5}\}$. We train each configuration for up to 6 epochs, evaluate on the validation set at the end of every epoch, and select the checkpoint with the highest validation accuracy. The selected configuration is the same across all (model, task) pairs: batch size 36 and learning rate $4\mathrm{e}{-5}$.

Optimization. All LLMs are fully fine-tuned using the AdamW optimizer with a cosine learning-rate scheduler.

B Evaluation details

B.1 Model merging hyperparameters

For optimization-based methods (AdaMerging, AdaMerging++, and DivMerge), we dynamically adjust the batch size based on the model scale to accommodate memory constraints: we use a batch size of 16 for base-scale models (SigLIP-Base, ViT-B/32, ViT-B/16) and a batch size of 4 for large-scale models (SigLIP-Large, ViT-L/14).

Search Range for Coefficients. For evaluation-based methods that require a grid search over the merging coefficient α (as reported in Table 3), we define the search space following the recommendations in their respective original works:

- **Task Arithmetic:** We search within the range $(0, 1]$ using 16 equidistant points (an interval of 0.0625).
- **DARE and TIES:** We search within the range $(0, 1.8]$ using 18 points (an interval of 0.1).

Method-specific Settings. The structural and optimization hyperparameters for each method are as follows:

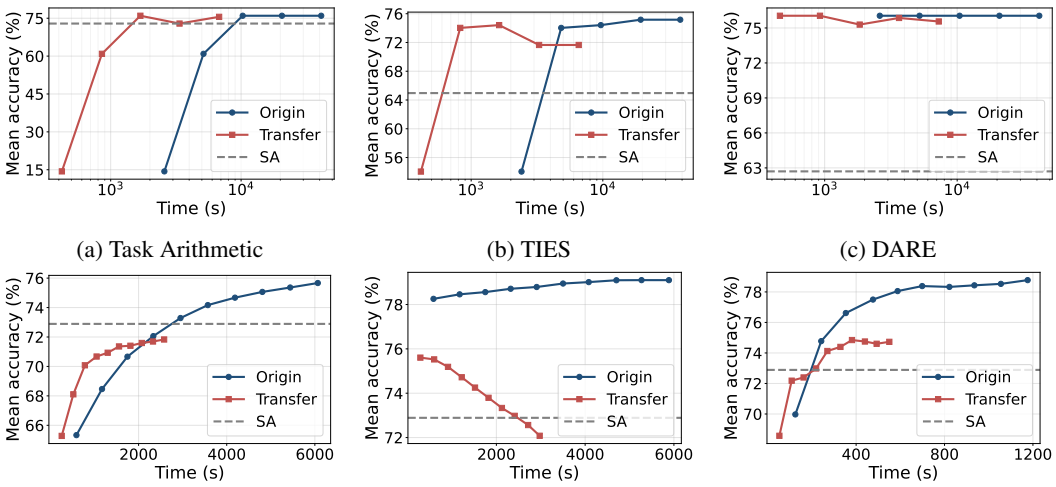
- **AdaMerging:** We optimize coefficients for 500 steps using the Adam optimizer with a learning rate of 1×10^{-3}. We use 1,000 samples to compute the merging objective and set the initial coefficient to 0.5.

[1] https://huggingface.co/datasets/nvidia/Nemotron-SFT-Instruction-Following-Chat-v2
[2] https://huggingface.co/datasets/miulab/usefulness-judge

- **AdaMerging++:** We adopt the identical optimization settings as AdaMerging. Additionally, we incorporate TIES-style masking with a top-k retention ratio of 20% ($k = 20$) and use summation as the merging function.
- **DivMerge:** The coefficients are optimized for 100 steps using the Adam optimizer with a learning rate of 1×10^{-2}. We set the initial coefficient to 0.5, use 200 samples for the optimization process, and employ the Jensen-Shannon (JS) divergence as the metric.
- **TIES:** We apply a top-k filtering ratio of 20% ($k = 20$) and use summation to merge the weight updates.
- **DARE:** We apply a random drop rate of 0.5 to the model weights before merging.

C Efficiency analysis complete results

In this section, we provide the complete budget analysis curves for the two proxy→target pairs not shown in the main paper: ViT-B/32 → ViT-L/14 (Figure 4) and ViT-B/16 → ViT-L/14 (Figure 5). The trends are consistent with those reported for SigLIP-Base → SigLIP-Large in the main paper: across all six merging methods, transferring search/training results from a smaller proxy model achieves accuracy comparable to (or surpassing) the simple-average baseline at a substantially lower computational budget.



(a) Task Arithmetic (b) TIES (c) DARE

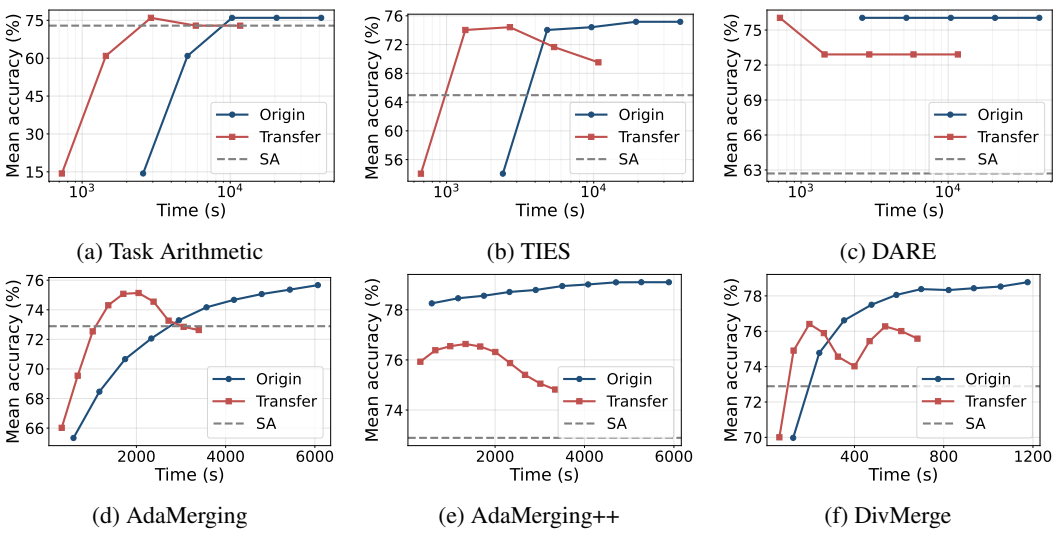
(d) AdaMerging (e) AdaMerging++ (f) DivMerge

Figure 4: **Budget analysis on ViT-B/32 → ViT-L/14.** We compare direct search (Origin) against coefficient transfer (αTransfer) under varying time budgets. Dashed lines indicate the simple average (SA) baseline. Top and bottom rows show evaluation- and gradient-based methods, respectively. Consistent with the SigLIP results, αTransfer shifts the performance curves to the left, reaching comparable accuracy in a fraction of the time.

C.1 Effect of k in AdaMerging++

Since AdaMerging++ incorporates TIES-style masking, its optimization dynamics are intrinsically tied to the sparsity parameter k (the percentage of weights retained). To investigate the performance gap observed in our primary evaluation, we conduct an ablation study varying $k \in \{20, 40, 60, 80\}$ for the SigLIP Base-to-Large transfer, comparing direct target search (*Origin*) against α**Transfer** (Transfer) (Figure 6).

The results indicate that the effectiveness of αTransfer in AdaMerging++ is influenced by the sparsity level. Under higher sparsity settings ($k = 20, 40$), heavy masking can enhance structural differences between proxy and target models, which moderately limits transferability and allows direct target search to achieve a higher performance ceiling.



(a) Task Arithmetic (b) TIES (c) DARE

(d) AdaMerging (e) AdaMerging++ (f) DivMerge

Figure 5: **Budget analysis on ViT-B/16 → ViT-L/14.** We compare direct search (Origin) against coefficient transfer (αTransfer) under varying time budgets. Dashed lines indicate the simple average (SA) baseline. Top and bottom rows show evaluation- and gradient-based methods, respectively. Consistent with previous observations, αTransfer shifts the performance curves to the left, reaching comparable accuracy in a fraction of the time.

Conversely, under lower sparsity ($k = 60, 80$), αTransfer yields strong results. It demonstrates faster initial convergence and achieves comparable or higher final accuracy than direct target optimization. This observation suggests that αTransfer remains a viable and efficient strategy for AdaMerging++ when a sufficient proportion of weights is retained during the optimization process.

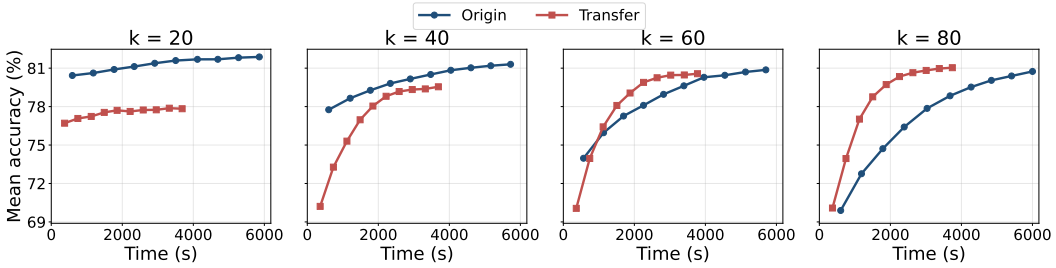


Figure 6: **Impact of sparsity on αTransfer in AdaMerging++.** We compare exhaustive target optimization (Origin) against αTransfer across varying TIES retention ratios $k \in \{20, 40, 60, 80\}$ on the SigLIP-Base → SigLIP-Large scaling chain. The x-axis represents the wall-clock training time. The effectiveness of αTransfer is highly sensitive to the sparsity level: at extreme sparsity ($k = 20$), direct target search achieves a higher performance ceiling. However, as the retention ratio increases ($k \geq 60$), αTransfer dominates the compute-performance trade-off, achieving massive speedups while converging to a higher final accuracy than direct target search.

D HYBRID SEARCH RESULTS FOR LLMS

We additionally evaluate the hybrid search strategy on LLMs. Following the same procedure described in Section 4.5, we use the transferred coefficients as initialization and allocate a small target-side budget for refinement. As shown in Table 8, hybrid search recovers the original merging performance for all evaluated proxy-target pairs, while achieving substantial computational speedups over exhaustive target optimization.

Table 8: **Hybrid search results on LLMs.** Hybrid search recovers the accuracy of direct search while substantially reducing the computational cost.

Proxy → Target	Method	Acc (%)			Speedup	
		Origin	Transfer	Hybrid	Transfer	Hybrid
Qwen3-0.6B → Qwen3-1.7B	Task Arithmetic	**76.44**	74.63	**76.44**	11.00×	2.29×
	TIES	**58.35**	58.26	**58.35**	18.96×	4.34×
	TIES-Pairwise	**71.15**	70.37	**71.15**	19.94×	3.05×
Qwen3-0.6B → Qwen3-4B	Task Arithmetic	**76.03**	**76.03**	**76.03**	11.03×	2.30×
	TIES	**66.04**	65.26	**66.04**	18.98×	4.35×
	TIES-Pairwise	**73.69**	**73.69**	**73.69**	19.97×	3.06×

E PER-TASK ACCURACY BREAKDOWN

E.1 VISION MODELS

Table 9 reports the per-task accuracy for each proxy → target pair, complementing the average scores reported in the main paper.

E.2 LARGE LANGUAGE MODELS

Table 10 reports the per-task accuracy for each LLM proxy → target pair, complementing the average scores reported in the main paper.

Table 9: Per-task accuracy (%) for each proxy→target pair. **Or** = origin (target's best coefficients evaluated on target); **Tr** = transfer (proxy-best coefficients evaluated on target).

Method		DTD	EuroSAT	FER	Food	GTSRB	RESISC	Cars	SUN	Avg
ViT-B/32 → ViT-L/14										
Task Arithmetic	Or	59.63	89.07	51.67	91.50	77.53	85.40	79.59	73.48	75.98
	Tr	60.53	87.11	50.35	92.62	73.34	85.59	81.10	73.77	75.55
TIES	Or	54.41	85.04	55.92	93.02	74.35	86.25	77.29	75.06	75.17
	Tr	57.23	75.85	49.96	94.13	58.12	84.03	80.20	73.70	71.65
DARE	Or	60.05	88.74	51.11	92.04	76.67	85.65	80.39	73.72	76.05
	Tr	60.59	87.07	50.32	92.63	73.33	85.56	81.18	73.77	75.56
AdaMerging	Or	59.68	89.93	49.39	85.90	91.83	81.24	75.96	71.41	75.67
	Tr	58.14	97.67	47.41	87.80	63.49	72.79	76.05	71.30	71.83
AdaMerging++	Or	62.98	88.04	56.09	89.87	92.94	86.03	82.48	74.38	79.10
	Tr	56.54	97.44	49.68	85.52	64.09	77.03	76.36	69.97	72.08
DivMerge	Or	72.55	89.70	60.64	86.75	89.90	78.43	82.34	69.86	78.77
	Tr	69.10	97.33	40.50	89.59	68.54	81.89	79.39	71.50	74.73
ViT-B/16 → ViT-L/14										
Task Arithmetic	Or	59.63	89.07	51.67	91.50	77.53	85.40	79.59	73.48	75.98
	Tr	60.53	81.93	47.21	93.31	62.05	83.78	81.36	72.94	72.89
TIES	Or	54.41	85.04	55.92	93.02	74.35	86.25	77.29	75.06	75.17
	Tr	56.76	70.07	47.37	94.11	53.25	81.63	80.14	72.88	69.53
DARE	Or	60.05	88.74	51.11	92.04	76.67	85.65	80.39	73.72	76.05
	Tr	60.69	81.81	47.19	93.31	62.10	83.83	81.40	72.95	72.91
AdaMerging	Or	59.68	89.93	49.39	85.90	91.83	81.24	75.96	71.41	75.67
	Tr	60.21	66.04	45.37	90.51	97.89	70.89	74.06	76.07	72.63
AdaMerging++	Or	62.98	88.04	56.09	89.87	92.94	86.03	82.48	74.38	79.10
	Tr	62.07	71.89	50.49	89.51	97.88	75.14	75.64	75.92	74.82
DivMerge	Or	72.55	89.70	60.64	86.75	89.90	78.43	82.34	69.86	78.77
	Tr	78.03	68.33	49.29	89.52	96.44	73.92	75.29	73.84	75.58
SigLIP-Base → SigLIP-Large										
Task Arithmetic	Or	72.82	96.37	57.26	90.19	89.27	68.05	93.97	73.87	80.22
	Tr	71.38	91.15	51.98	93.22	82.15	72.25	94.13	74.36	78.83
TIES	Or	66.91	96.67	61.94	90.25	82.95	58.05	90.47	72.73	77.50
	Tr	67.39	95.59	59.15	91.53	81.37	59.87	91.17	73.05	77.39
DARE	Or	72.66	94.48	53.61	92.61	85.08	71.75	94.07	74.43	79.84
	Tr	72.39	93.33	53.05	92.83	84.24	71.89	94.07	74.39	79.52
AdaMerging	Or	75.74	93.89	52.84	88.45	93.48	77.70	93.98	75.75	81.48
	Tr	81.91	95.19	45.75	88.11	93.77	71.94	93.56	71.95	80.27
AdaMerging++	Or	77.02	95.96	50.33	90.96	94.01	76.02	94.48	76.23	81.88
	Tr	74.36	96.93	49.53	80.28	96.79	63.89	92.33	68.57	77.83
DivMerge	Or	76.49	90.22	58.85	88.88	90.77	87.17	94.12	77.35	82.98
	Tr	81.76	94.78	40.25	89.94	95.20	74.16	92.34	71.63	80.01

Table 10: Per-task accuracy (%) for each LLM proxy→target pair. **Origin** = target's best coefficients evaluated on target; **Transfer** = proxy-best coefficients evaluated on target.

Method		DDXPlus	Banking77	Usefulness Judge	IFEval	Avg
Qwen3-0.6B → Qwen3-1.7B						
Task Arithmetic	Origin	85.71	95.70	79.20	45.13	76.44
	Transfer	80.78	93.50	77.60	46.64	74.63
TIES	Origin	8.56	77.10	82.40	65.34	58.35
	Transfer	7.94	79.60	79.60	63.28	57.61
TIES-Pairwise	Origin	75.62	92.30	72.00	44.67	71.15
	Transfer	72.17	90.40	72.80	46.09	70.37
Qwen3-0.6B → Qwen3-4B						
Task Arithmetic	Origin	91.21	92.00	78.40	42.51	76.03
	Transfer	91.21	92.00	78.40	42.51	76.03
TIES	Origin	36.17	93.00	80.00	54.99	66.04
	Transfer	28.29	91.20	77.20	61.31	64.50
TIES-Pairwise	Origin	54.82	89.30	68.00	29.57	60.42
	Transfer	53.91	85.40	67.60	34.26	60.29