Predicting Mergeability of Language Model Domain Experts

Ervin Dervishaj[*]
University of Copenhagen
erde@di.ku.dk

Quynh Do
Amazon
doquynh@amazon.de

Giannis Karamanolakis
Amazon
karamai@amazon.com

Anna Rumshisky
Amazon
Anna_Rumshisky@uml.edu

Salma Sohrabi-Jahromi
Amazon
salmasj@amazon.de

Judith Gaspers
Amazon
gaspers@amazon.de

Abstract

Model merging allows extending an expert model's capabilities by fusing its weights with another expert. However, task interference has been shown to negatively affect mergeability. To address this, prior work has mainly focused on introducing new merging methods. Despite reducing task interference, the effectiveness of these merging methods is only evident post-merging, requiring a full evaluation of the merged model, which can be expensive for contemporary LLMs. In this work, we study the overlooked predictive power of structural and functional properties of merged experts for mergeability. Our experiments with 8 expert LLMs spanning several programming languages, math, and knowledge show that activation L2 distance and activation cosine similarity emerge as the most important properties for predicting effective merging, contrasting with prior findings on vision models where gradient-based metrics dominated. We further show that task vector L2 distance, a cheaply computed structural metric, provides a mechanistic link between mergeability prediction and training dynamics: it explains why learning rate effects on mergeability are domain-dependent in LLMs, contradicting the universal "higher noise improves merging" principle from vision. This enables a practical intervention where mergeability metrics guide learning rate selection during expert training.

1 Introduction

Large language models (LLMs) are increasingly fine-tuned into domain-specific experts, for instance, for code generation in various programming languages, mathematical reasoning, or knowledge-intensive tasks. Model merging offers an attractive alternative to expensive multi-task training: by combining the weights of individually trained experts, one can produce a single model with multi-domain capabilities without additional training (Yang et al., 2026).

However, not all expert pairs merge well. Task interference, where combining task vectors degrades performance on one or both tasks, remains a fundamental challenge (Gargiulo et al., 2025; Fang et al., 2025; Hitit et al., 2026). So far, previous work has primarily focused on developing increasingly sophisticated merging algorithms, such as TIES-Merging (Yadav et al., 2023), DARE (Yu et al., 2024), Task Singular Vectors (Gargiulo et al., 2025), and NeuroMerging (Fang et al., 2025). While these methods reduce interference, a critical limitation persists: *their effectiveness can only be assessed after merging and fully evaluating the resulting model*. Especially for large-scale LLMs evaluated across dozens of benchmarks, this post-hoc evaluation is computationally expensive.

This motivates a complementary question: *can we predict whether two experts will merge well before performing the merge?* Two independent research threads have begun to address this. First, Zhou et al. (2026) introduced a framework for predicting mergeability from structural and functional metrics, however, only for ViT-B/16 models on image classification. Second, Zhang et al. (2026) showed that training dynamics (learning rate, batch size, weight decay) shape the merging loss landscape, however, they studied this only through post-merge performance, not through the lens of predictive metrics. These two threads have not been connected: it remains unknown whether the established metric importance hierarchy transfers across architectures and modalities, and how training dynamics interact with mergeability when experts are trained on heterogeneous domains.

Extending mergeability prediction to LLMs is not straightforward. Decoder-only transformers lack an encoder, requiring adaptation of the metric framework (e.g., decoder gradients replace encoder

[*] Work done during an internship with Amazon.

Proceedings of the 15th International Joint Conference on Natural Language Processing and the 5th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 5464–5477
November 6-10, 2026 ©2026 Association for Computational Linguistics

gradients). In addition, LLM domain experts can be created through continued pretraining over billions of tokens rather than few-epoch fine-tuning, operating in a fundamentally different optimization regime. The tasks themselves, i.e. autoregressive code generation and mathematical reasoning, differ qualitatively from image classification. These differences make it unclear a priori which (if any) findings from vision transfer to language.

We address these open questions with the following contributions:

1. We present the first study of mergeability prediction for LLM domain experts, using 8 experts (6 programming languages, math, and knowledge) trained from a 1B-parameter decoder-only model. Our results reveal that the metric importance hierarchy is **architecture- and modality-dependent**: activation-based metrics (L2 distance and cosine similarity) dominate for LLMs (accounting for $\sim$85% of predictive power), whereas gradient-based metrics dominated for vision models (Zhou et al., 2026).

2. We provide an initial joint investigation of mergeability metrics and training dynamics, bridging the frameworks of Zhou et al. (2026) and Zhang et al. (2026) for decoder-only LLMs. In particular, we show that task vector L2 distance can explain why learning rate effects on mergeability are **domain-dependent**, and demonstrate that this metric can serve as a training-time signal to guide learning rate selection for merge-friendly expert training.

2 Related Work

Model Merging Methods. Task arithmetic (Ilharco et al., 2023) combines models by adding scaled task vectors $\tau = \theta_{\text{ft}} - \theta_{\text{base}}$ to the base model, with θ_{ft} and θ_{base} denoting the fine-tuned and base checkpoint parameters, respectively. TIES-Merging (Yadav et al., 2023) addresses task vector sign conflicts by trimming, electing signs, and disjoint merging. DARE (Yu et al., 2024) applies random pruning to task vectors before merging. More recent approaches include Task Singular Vectors (Gargiulo et al., 2025) and NeuroMerging (Fang et al., 2025), which disentangle neuronal subspaces to reduce interference. Hitit et al. (2026) conducted a large-scale evaluation finding that Task Arithmetic remains the only method that reliably yields performance gains in realistic LLM merging scenarios. All these methods focus on *how* to merge; our work addresses the complementary question of *whether* a given pair of experts will benefit from merging.

Mergeability Prediction. Zhou et al. (2026) proposed a framework to predict merging success from 28 pairwise metrics spanning task vector geometry, effective rank, subspace overlap, activations, and gradients. Using ViT-B/16 models fine-tuned on 20 image classification tasks, they found that gradient-based metrics (encoder gradient L2 distance and input gradient L2 distance) showed the highest coefficients in a Lasso-regularized linear model. However, they noted high coefficient variance and studied only a single architecture under fixed training hyperparameters. Our work extends mergeability prediction to decoder-only LLMs, requiring metric adaptation (decoder gradients replace encoder gradients), and reveals a fundamentally different metric hierarchy where activations dominate over gradients.

Training Dynamics and Mergeability. Zhang et al. (2026) identified the *effective noise scale* $\tilde{S} = \eta/B(1-\mu)$ as a unifying quantity capturing how optimizer components affect merging outcomes. They showed that higher learning rates, smaller batch sizes, and weight decay improve mergeability in vision models and small language models (GPT-2 scale), arguing that increased effective noise leads to flatter minima which are more amenable to merging. Li et al. (2026) provided theoretical grounding, proving that over-trained experts cause merging failure. Critically, these works measure training dynamics' effect on mergeability only through post-merge evaluation—they do not study whether predictive metrics can capture these effects, nor whether their conclusions generalize to LLM continued pretraining. We bridge this gap by jointly investigating mergeability metrics and training dynamics, finding that the learning rate–mergeability relationship is domain-dependent in LLMs.

Model Merging at Scale. Wang et al. (2026) established power-law scaling relationships linking model size and expert count to merged model performance. Ueda et al. (2026) explored merging continual pre-training models for domain-specialized LLMs in finance, finding that model similarity correlates with merging success. Yao et al. (2026)

proposed activation-guided rotation-aware merging for streaming LLM updates, providing further evidence that activation subspaces carry important information for effective merging which is consistent with our finding that activation metrics are the strongest mergeability predictors for LLMs.

3 Methodology

3.1 Expert Training Setup

We train domain experts from a 1B-parameter decoder-only transformer with a LlaMA-like architecture (Touvron et al., 2023) (RoPE positional embeddings, SwiGLU activations, grouped-query attention, RMSNorm). We chose to train our own base model rather than use publicly available checkpoints because no single public model provides a sufficient number of domain experts of the same size trained from the same base checkpoint[1], which is a requirement for controlled mergeability analysis.

The base model is pretrained from scratch for 500B tokens on a mixture of web crawl, code, scientific, and mathematical data; details about the mixture including datasets and ratios can be found in Table 8 (appendix C) along with the mixtures for all of the experts. We pre-train the base model using a constant learning rate (lr $= 3 \times 10^{-4}$). From the base checkpoint, we create 8 domain experts by training for an additional 100B tokens each with annealed learning rate and data mixtures enriched for the target domain. The 8 experts cover: **Python**, **Java**, **JavaScript**, **C++**, **C#**, **PHP** (programming languages), **Math** (mathematical reasoning), and **Knowledge** (factual/encyclopedic content). Each expert's data mix maintains a base of general data while upsampling domain-specific sources (e.g., the Python expert trains on ∼42.5% Python code vs. ∼0% in the base pretrain mix).

3.2 Mergeability Metrics

Following Zhou et al. (2026), we compute pairwise metrics between experts spanning structural and functional properties. Given two experts with parameters θ_A and θ_B and their task vectors $\tau_A = \theta_A - \theta_0$ and $\tau_B = \theta_B - \theta_0$ (where θ_0 is the shared base model), we compute:

Structural Metrics (computed directly from parameters):

- **Task Vector L2 Distance**: $\|\tau_A - \tau_B\|_2$
- **Task Vector Cosine Similarity**: $\cos(\tau_A, \tau_B)$
- **Singular Value Overlap**: Cosine similarity between top-100 normalized singular value distributions of individual weight matrices, averaged across layers.

Functional Metrics (computed over a calibration set):

- **Activation L2 Distance / Cosine Similarity**: Mean of the representations of the last hidden state of the model.
- **Decoder Gradient L2 Distance / Cosine Similarity**: Mean of the gradient of the loss with respect to the decoder parameters, excluding the input embedding layer and language modeling head.
- **Input Gradient L2 Distance / Cosine Similarity**: Mean of the gradient of the loss with respect to the input embeddings.

Compared to Zhou et al. (2026) who used 28 metrics (including effective rank variants) over only 200 calibration samples *across all tasks*, we use a focused set of 9 metrics computed over a substantially larger calibration set (200 samples *per task* from the **validation** set of each task for a total of 7,140 samples), providing more stable functional metric estimates.

3.3 Normalized Post-Merge Performance

Following standard practice in the model merging literature (Ilharco et al., 2023; Yadav et al., 2023; Ortiz-Jimenez et al., 2023; Yadav et al., 2025; Zhou et al., 2026), we use normalized post-merge performance to account for differences in task difficulty and enable comparison across benchmarks. For a merged model M evaluated on task t, the normalized performance is:

$$p_t = \frac{s_{M,t}}{s_{\text{expert}_t,t}} \tag{1}$$

where $s_{M,t}$ is the merged model's score on task t and $s_{\text{expert}_t,t}$ is the score of the task-specific expert (i.e., the model fine-tuned on task t). A value of 1 indicates that the merged model matches the expert. The final mergeability score for a pair of experts is the average normalized performance across tasks.

[1]The shared-base-checkpoint is the standard experimental protocol in the model merging literature (Ilharco et al., 2023; Yadav et al., 2023; Yu et al., 2024).

3.4 Correlation Model

Following (Zhou et al., 2026), we address predicting whether two models merge well before performing the merge by optimizing a regularized linear model of mergeability metrics to maximize the correlation with the normalized post-merge performance. Given K mergeability metrics (see Section 3.2), we optimize:

$$\mathbf{w}^\star = \arg\max_{\mathbf{w}} \; r(\mathbf{X}\mathbf{w}, \mathbf{p}) \quad \text{s.t.} \quad \sum_{k=1}^{K} w_k = 1 \tag{2}$$

where $\mathbf{X} \in \mathbb{R}^{N \times K}$ is the matrix of normalized ($[-1, 1]$) mergeability metrics of N pairs of merged models, $\mathbf{p}$ is the vector of normalized post-merge performance, $\mathbf{w}$ is the set of weights to be optimized and $r(\cdot, \cdot)$ is the Pearson Correlation coefficient. We use Lasso-regularization (with λ tuned in $[0.01, 0.1, 1, 10]$) in order to retrieve a smaller subset of mergeability metrics with highest predictive power.

To optimize the linear model we use **Leave-One-Expert-Out** cross validation: the validation fold consists of all merged pairs involving a held-out expert with the remaining pairs forming the training fold, allowing for evaluation of the model on novel experts (Zhou et al., 2026). We validate statistical significance through a permutation test (1,000 permutations of the target variable).

4 Experiments

4.1 Experimental Setup

We merge all $\binom{8}{2} = 28$ expert pairs using **Task Arithmetic** (Ilharco et al., 2023) with a default scaling coefficient $\alpha = 0.5$. We choose Task Arithmetic based on recent findings that it is the most reliable merging method for LLMs (Hitit et al., 2026). Models were merged with MergeKit framework[2] (Goddard et al., 2024).

Evaluation Benchmarks. We evaluate individual experts and their merges on 25 benchmarks spanning: code generation in Python/Java/JavaScript/C#/PHP/Ruby/TypeScript (HumanEval (Chen et al., 2021) and MBXP (Athiwaratkun et al., 2023)), mathematical reasoning (GSM8K (Cobbe et al., 2021), MGSM (Shi et al., 2023), MATH (Hendrycks et al., 2021b), MATH-Verify[3]), commonsense reasoning (ARC-Challenge (Clark et al., 2018), CommonsenseQA (Talmor et al., 2019)), language understanding (MMLU (Hendrycks et al., 2021a), MMLU-Pro (Wang et al., 2024), TriviaQA (Joshi et al., 2017), BoolQ (Clark et al., 2019)), and general reasoning (BBH (Suzgun et al., 2023))). We use as final correlation target the average normalized post-merge performance across all benchmarks.

[2] https://github.com/arcee-ai/mergekit

[3] https://github.com/huggingface/Math-Verify

4.2 Pairwise Metric Correlations

We first examine potential correlations between the mergeability metrics, reported through the heatmap in Figure 1. We make the following observations: i) L2 distance and cosine similarity are highly correlated for task vector and activation metrics, separately, with input and decoder gradient showing limited and non-statistically significant correlations, ii) Singular value overlap shows high correlation with both task vector metrics, despite the former operating on 2D, layer-wise task vector weights instead of 1D and iii) decoder gradient metrics are strongly correlated with structural and input gradient metrics. The connection between model gradients and task vectors in particular, was first characterized by Zhou et al. (2025), albeit under stricter conditions like full batch gradient descent (GD) and only on feed-forward networks. The observed correlation can imply that this connection holds for transformer-based architectures and under stochastic GD.

4.3 Individual Metric Correlations

To identify important mergeability metrics, we examine whether individual metrics correlate with post-merge performance (Table 1). Unlike Zhou et al. (2026), who reported weak correlations for Task Arithmetic merging of vision models ($|r| < 0.17$), we find that for LLMs, task vector L2 distance ($r = -0.409$, $p = 0.031$) and cosine similarity ($r = 0.408$, $p = 0.031$) show strong correlations. This indicates that experts whose fine-tuning trajectories diverge more from each other (larger L2 distance, lower cosine similarity) tend to merge less effectively. We further investigate this correlation at different Task Arithmetic scaling coefficient (appendix A). The task vector L2 distance correlation with performance **is stable, strong, and highly significant for** $\alpha \leq 0.6$ (the practically-relevant range for Task Arithmetic), then vanishes/flips at where the merge is dominated by full task vector addition. We provide additional results with

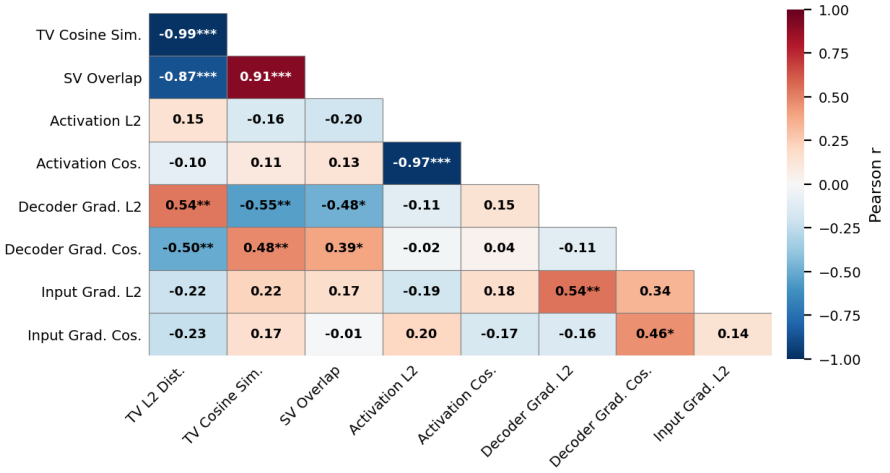


Figure 1: Pairwise metric correlations (* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$). L2 distance and cosine similarity are strongly correlated for task vector and activation metrics. Decoder gradient and task vector metrics are the only ones that correlate across functional and structural metrics.

Metric	Pearson r [95% CI]	p-value
Task Vector L2 Distance	-0.409 [-0.79, -0.06]	**0.031**
Task Vector Cosine Sim.	0.408 [+0.02, +0.81]	**0.031**
Singular Value Overlap	0.309 [-0.12, +0.72]	0.110
Activation L2 Distance	0.214 [-0.07, +0.49]	0.273
Activation Cosine Sim.	-0.128 [-0.45, +0.14]	0.517
Decoder Grad. L2 Dist.	-0.116 [-0.51, +0.14]	0.556
Decoder Grad. Cosine Sim.	0.373 [+0.01, +0.80]	0.051
Input Grad. L2 Dist.	0.192 [-0.15, +0.50]	0.328
Input Grad. Cosine Sim.	0.260 [-0.13, +0.52]	0.182

Table 1: Individual metric correlations with normalized post-merge performance. 95% bootstrap confidence interval is given by 5000 resamples over 28 merged pairs. Bold p-values indicate significance at $\alpha = 0.05$. Unlike Zhou et al. (2026) who found no individually significant metrics on vision models, task vector metrics show significant correlation for LLMs.

Held-out Expert	Train r	Val. r
C++	0.755	0.895
C#	0.589	0.492
Java	0.649	0.691
JavaScript	0.599	0.616
Knowledge	0.679	0.601
Math	0.491	0.110
PHP	0.671	0.527
Python	0.837	-0.012
Mean ($\pm$ std)	0.659 ($\pm$0.099)	0.490 ($\pm$0.281)

Table 2: Leave-one-expert-out CV results ($\lambda = 10$). The 26% relative gap between train and validation correlations is comparable to Zhou et al. (2026)'s findings on vision models.

TIES-Merging (Yadav et al., 2023) (appendix B) where mergeability metrics (in particular task vector L2 distance and cosine similarity) show strong correlation with performance, confirming that the mergeability prediction framework carries signal beyond Task Arithmetic.

4.4 Linear Model Results

Given the limited correlations between structural and functional mergeability metrics, we present in Table 2 the leave-one-expert-out results of the correlation model described in Section 3.4. The mean validation correlation is $r = 0.490$, an improvement over using the best individual metric alone (Section 4.3), showing that the combination of metrics provides additional predictive signal. Between the train and validation folds, we observe a relative decrease of 26%, consistent with vision models (Zhou et al., 2026). The mean validation correlation was corroborated by a permutation test ($p = 0.015$, $n = 1{,}000$). While some experts generalize quite well (e.g., C++, Java, JavaScript, Knowledge), the model struggles with Math and Python, potentially because i) the metric distributions of the Math expert are quite different to the other experts (Figure 5) with the majority of the 28 pairs consisting of code-code combinations, and ii) Python is syntactically different from the other languages. Training on all 28 pairs yields $r = 0.703$ ($p < 3 \times 10^{-4}$), with a permutation test p-value of 0.003.

4.5 Feature Importance

Table 3 shows the metric weights from the full model and the reduced model after removing metrics with zero weight and/or high variance.

The key finding is that **activation-based met-**

Metric	Full	Reduced
TV Cosine Similarity	0.191	0.092
Activation L2 Distance	0.416	**0.470**
Activation Cosine Sim.	0.180	**0.381**
Decoder Grad. L2 Dist.	0.109	0.051
Input Grad. Cosine Sim.	0.018	0.007
TV L2 Distance	0	—
SV Overlap	0	—
Input Grad. L2 Dist.	0	—
Decoder Grad. Cos. Sim.	0.085	—

Table 3: Metric weights from the correlation model. The reduced model (5 metrics) achieves $r = 0.703$ ($p < 3 \times 10^{-4}$). Activation metrics account for 85% of the total weight.

rics dominate for LLMs, with a combined weight of 0.85 in the reduced model ($\lambda = 10$). Sweeping Lasso-regularization coefficient λ across $[0.01, 0.1, 1]$ showed activation-based metrics' share of 0.49, 0.54 and 0.84 of the combined weight, confirming the importance of activation-based metrics. This contrasts sharply with Zhou et al. (2026)'s findings on ViT models (Dosovitskiy et al., 2021), where gradient-based metrics (encoder gradient L2 distance and input gradient L2 distance) had the largest coefficients. In our LLM setting, input gradient L2 distance receives zero weight.

We hypothesize that this difference stems from the scale and depth of LLMs: activations aggregate information across all layers and provide a more holistic functional similarity measure, whereas gradients computed with respect to the input or decoder parameters may be too local to capture the complex interactions within deep transformers.

4.6 Filtered-Benchmark Analysis

Averaging post-merge normalized performance across all benchmarks captures the realistic assumption that a merged model intended to replace multiple experts must not only preserve domain-specific capabilities but also maintain acceptable general-purpose performance; a merged code-math model that excels at Python but catastrophically degrades on language understanding would be considered a failed merge in practice.

However, a plausible second deployment scenario is the case when a practitioner values only the merged model's capabilities for which the constituent experts were explicitly fine-tuned for. To account for this scenario, we repeat the analysis considering only benchmarks relevant to each merged pair (e.g., for Python-Math merging, only Python and Math benchmarks). The model achieves $r = 0.578$ ($p = 0.001$) on the full data, with permutation test $p = 0.024$. In this setting (Table 4), task vector cosine similarity and decoder gradient L2 distance gain relative importance compared to the reduced full-benchmark case, while activation metrics remain dominant (combined weight 0.61).

Metric	Weight
TV Cosine Similarity	0.171
Activation L2 Distance	0.384
Activation Cosine Sim.	0.230
Decoder Grad. L2 Dist.	0.156
Input Grad. Cosine Sim.	0.059

Table 4: Metric weights from the reduced filtered-benchmark correlation model. The model achieves $r = 0.578$ ($p < 0.001$). Activation metrics account for 61% of the total weight.

Rank	Metric	Borda Score
1	TV Cosine Similarity	208 / 225
2	TV L2 Distance	205 / 225
3	Activation L2 Distance	129 / 225
4	Activation Cosine Sim.	120 / 225
5	Decoder Grad. L2 Dist.	109 / 225
6	SV Overlap	99 / 225
7	Decoder Grad. Cos. Sim.	97 / 225
8	Input Grad. L2 Dist.	89 / 225
9	Input Grad. Cos. Sim.	69 / 225

Table 5: Borda count ranking of metrics across 25 individual benchmarks. Task vector and activation metrics consistently rank highest.

4.7 Per-Benchmark Analysis

To provide an alternative view of metric importance, we train a separate correlation model per benchmark and rank metrics via Borda count across all benchmarks (Table 5).

The per-benchmark analysis confirms that both structural (task vector) and functional (activation) metrics provide stable predictive signals. The average correlation across all benchmarks is 0.893 (train) and 0.685 (validation), demonstrating strong per-task predictive ability with a 23.5% relative gap. We note that task vector L2 distance metric ranks highly per-benchmark despite receiving zero weight in the joint correlation model. This reflects the near-perfect correlation between task vector L2 distance and cosine similarity: Lasso selects one representative from correlated feature groups with the specific choice between these two metrics

largely interchangeable.

4.8 Learning Rate and Mergeability

Motivated by Zhang et al. (2026)'s finding that higher learning rates improve mergeability in vision models (through the effective noise scale mechanism), we investigate this relationship for LLM experts. We fine-tune additional Math, Python and Java experts with increasing learning rates (lr $\in \{3 \times 10^{-5}, 1 \times 10^{-4}, 3 \times 10^{-4}, 9 \times 10^{-4}\}$) from the pretrained checkpoint for 100B tokens, whilst keeping other hyperparameters unchanged.

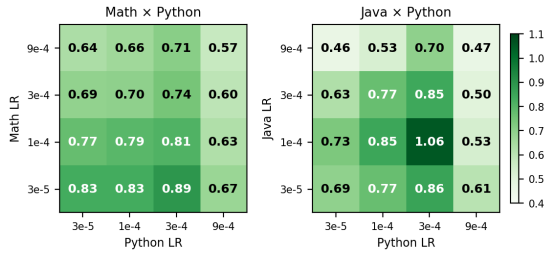


Figure 2: Normalized post-merge performance of Math-Python (left) and Java-Python (right) experts ($\alpha = 0.5$), fine-tuned with different learning rates.

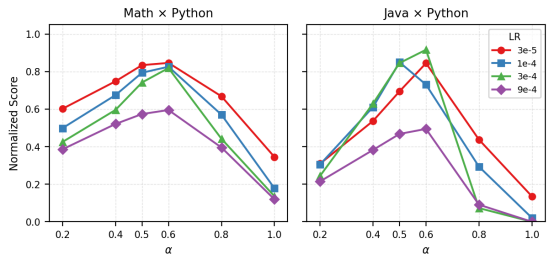


Figure 3: Normalized post-merge performance by Task Arithmetic scaling coefficient for Math-Python (left) and Java-Python (right) experts. While relatively larger learning rates improve Java-Python merging ($\alpha \leq 0.6$), Math-Python model benefits from lower learning rates.

Figure 2 shows the normalized post-merge performance of Math-Python and Java-Python experts fine-tuned with different learning rates ($\alpha = 0.5$). While Zhang et al. (2026) concluded that merging models fine-tuned with relatively high learning rates improves post-merge performance, we find this to be expert-dependent. For Java and Python experts, post-merge performance generally improves with increasing learning rate with the largest learning rate substantially deteriorating the performance, aligning with Zhang et al. (2026) findings in vision models. Math and Python experts on the other hand, merge better for smaller learning rates. We further investigate this finding by varying the Task Arithmetic scaling coefficient α (Figure 3) which can be considered a proxy measure for the loss flatness (Zhang et al., 2026). Figure 3 shows that for Math-Python merging, lower learning rates provide higher performance across all scaling coefficients.

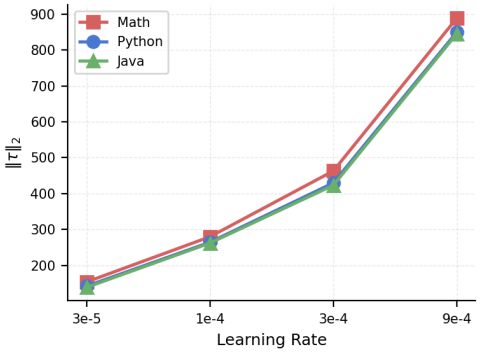


Figure 4: Task vector L2 norm ($\|\tau\|_2$) as a function of learning rate. Higher learning rates produce larger task vectors for all domains, with Math consistently deviating more from the base model than code experts.

To understand why the learning rate effect is domain-dependent, we examine the task vector magnitude of each expert as a function of learning rate (Figure 4). Higher learning rates produce monotonically larger task vectors for all domains, confirming that learning rate directly controls how far experts deviate from the shared base model. Crucially, the Math expert consistently exhibits larger $\|\tau\|_2$ than code experts at every learning rate, reflecting the greater distributional shift required for mathematical reasoning. Since task vector L2 distance is negatively correlated with mergeability ($r = -0.409$, Table 1), increasing learning rate for Math experts amplifies an already-large divergence, explaining their degraded mergeability at high learning rates. For code-code pairs (Java-Python), which start with smaller task vectors, moderate learning rate increases remain below the critical divergence threshold, allowing the flatness-promoting noise effect (Zhang et al., 2026) to dominate.

5 Discussion

Practical Implications. Our findings suggest a two-stage approach for mergeability prediction: (1) during training, monitor task vector L2 distance as an early signal—it is cheaply computed and individ-

ually significant; (2) post-training but pre-merging, compute activation metrics over a calibration set for more accurate prediction. This avoids the expensive full benchmark evaluation of merged models. Furthermore, the domain-dependence of the learning rate effect (Section 4.8) implies that practitioners should not universally increase learning rates to promote mergeability, but rather consider the specific domain pairing. To quantify the savings: in our experiments, full evaluation of all 28 merged models required $\sim$1,120 GPU-hours, while computing all 9 mergeability metrics required only $\sim$22 GPU-hours, a 50$\times$ reduction. This gap is expected to widen at larger model scales, where evaluation cost scales with both model size and generation length, while metric computation requires only a single forward pass over a fixed calibration set.

Why Activations Dominate for LLMs. The reversal of metric importance between vision models (gradients dominate) and LLMs (activations dominate) likely reflects architectural and training regime differences. In ViTs fine-tuned for few epochs on classification, gradients at the encoder and input carry concentrated task-discriminative information. In decoder-only LLMs trained for billions of tokens on autoregressive generation, activations aggregate information across all layers and positions, providing a holistic functional similarity measure. Gradients may be too local to capture the distributed representations that determine mergeability in deep transformers. This is corroborated by Yao et al. (2026), who independently found that activation subspaces are effective guides for LLM merging.

Bridging Metrics and Training Dynamics. Our joint investigation of mergeability metrics and learning rate effects reveals that these two research threads interact in non-trivial ways. The "higher noise $\rightarrow$ flatter minima $\rightarrow$ better merging" principle (Zhang et al., 2026) does not universally hold for LLM experts: while Java-Python (code-code) partially follows this pattern, cross-domain merges involving math experts show the opposite effect. This suggests that the effective noise scale framework, while useful for same-distribution merging, requires refinement for heterogeneous expert merging where the distance between task distributions matters as much as loss landscape geometry.

The individually significant task vector L2 distance ($r = -0.409$, Section 4.3) provides a mechanistic link between these two contributions. Math expert pairs already exhibit the largest task vector L2 distances among all pairs (Table 10), indicating greater deviation from the shared base model. Higher learning rates amplify this deviation, pushing task vectors further apart and degrading mergeability—precisely as the metric predicts. For code-code pairs like Java-Python, which start with smaller task vector distances ($\sim$430), moderate learning rate increases do not push them past a critical threshold, explaining why these pairs partially follow the vision-model pattern.

Learning Rate as a Mergeability Intervention. Beyond prediction, our findings suggest that mergeability metrics can *guide* training decisions. Task vector L2 distance, which is cheaply monitored during training, can inform learning rate selection for merge-friendly expert training. Specifically: when an expert's task vector naturally grows large, as happens for domains that diverge substantially from pretraining data (e.g., mathematical reasoning), practitioners should use lower learning rates to preserve proximity to the shared base model. Conversely, for domains that remain close in parameter space (e.g., related programming languages), moderately higher learning rates can be used to benefit from the flatness-promoting noise effect without exceeding the critical divergence threshold. This transforms mergeability metrics from passive post-hoc descriptors into active training signals, offering a principled alternative to the blanket "increase learning rate" recommendation of Zhang et al. (2026).

6 Conclusion

We presented the first study of mergeability prediction for LLM domain experts, bridging two previously independent research threads: mergeability metrics (Zhou et al., 2026) and training dynamics (Zhang et al., 2026). Our experiments reveal that the metric importance hierarchy is architecture- and modality-dependent: activation-based metrics dominate for LLMs (85% of predictive weight) whereas gradient-based metrics dominated for vision models. We further show that the relationship between learning rate and mergeability is domain-dependent in LLMs, contradicting the universal effective noise scale principle established for vision. Crucially, we demonstrate that task vector L2 distance provides a mechanistic link between these findings: it both predicts mergeability and explains why the learning rate effect varies across domains.

This enables a practical intervention where mergeability metrics guide learning rate selection during expert training—transforming passive prediction into active control of merge compatibility.

Limitations

Our study uses a single model scale (1B parameters). Wang et al. (2026) showed that scaling laws govern merging; our findings may evolve at larger scales, which may be explored in future work.

The 28 expert pairs provide statistical significance (validated via permutation tests) but limit the diversity of domain combinations, particularly for the Math and Knowledge experts.

Activation metrics require running inference over a calibration set, adding computational cost relative to purely structural metrics. However, this is substantially cheaper than full benchmark evaluation.

Acknowledgments

Generative AI tools were used to assist with language polishing and improving the clarity of the manuscript. The authors reviewed and take full responsibility for the final content.

References

Ben Athiwaratkun, Sanjay Krishna Gouda, Zijian Wang, Xiaopeng Li, Yuchen Tian, Ming Tan, Wasi Uddin Ahmad, Shiqi Wang, Qing Sun, Mingyue Shang, Sujan Kumar Gonugondla, Hantian Ding, Varun Kumar, Nathan Fulton, Arash Farahani, Siddhartha Jain, Robert Giaquinto, Haifeng Qian, Murali Krishna Ramanathan, and 6 others. 2023. Multi-lingual evaluation of code generation models. In *The Eleventh International Conference on Learning Representations*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *Preprint*, arXiv:1803.05457.

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

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Zitao Fang, Guodong Du, Shuyang Yu, Yifei Guo, Yiwei Zhang, Yiyao Cao, Jing Li, Ho-Kin Tang, and Sim Kuan Goh. 2025. To see a world in a spark of neuron: Disentangling multi-task interference for training-free model merging. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 15720–15740, Suzhou, China. Association for Computational Linguistics.

Antonio Andrea Gargiulo, Donato Crisostomi, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, and Emanuele Rodolà. 2025. Task singular vectors: Reducing task interference in model merging. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18695–18705.

Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vladimir Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. 2024. Arcee's MergeKit: A toolkit for merging large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 477–485, Miami, Florida, US. Association for Computational Linguistics.

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

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Oğuz Kağan Hitit, Leander Girrbach, and Zeynep Akata. 2026. A systematic study of in-the-wild model merging for large language models. *Transactions on Machine Learning Research*.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Qinglun Li, Anke Tang, Miao Zhang, Mengzhu Wang, Quanjun Yin, and Li Shen. 2026. A unified generalization framework for model merging: Tradeoffs, non-linearity, and scaling laws. *Preprint*, arXiv:2601.21690.

Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. 2023. Task arithmetic in the tangent space: Improved editing of pre-trained models. In *Advances in Neural Information Processing Systems*, volume 36, pages 66727–66754. Curran Associates, Inc.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. 2023. Language models are multilingual chain-of-thought reasoners. In *The Eleventh International Conference on Learning Representations*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, Toronto, Canada. Association for Computational Linguistics.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Kentaro Ueda, François Portet, Hirohiko Suwa, and Keiichi Yasumoto. 2026. Merging continual pretraining models for domain-specialized llms: A case study in finance. In *Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026)*, pages 10114–10129, Palma, Mallorca, Spain. European Language Resources Association (ELRA).

Yuanyi Wang, Yanggan Gu, Yiming Zhang, Qi Zhou, Zhaoyi Yan, Congkai Xie, Xinyao Wang, Jianbo Yuan, and Hongxia Yang. 2026. Model merging scaling laws in large language models. In *Forty-third International Conference on Machine Learning*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, volume 37, pages 95266–95290. Curran Associates, Inc.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023. Ties-merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems*, volume 36, pages 7093–7115.

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

Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. 2026. Model merging in llms, mllms, and beyond: Methods, theories, applications, and opportunities. *ACM Comput. Surv.*, 58(8).

Yuxuan Yao, Haonan Sheng, Qingsong Lv, Han Wu, Shuqi Liu, Zehua Liu, Zengyan Liu, Jiahui Gao, Haochen Tan, Xiaojin Fu, Haoli Bai, Hing Cheung So, Zhijiang Guo, and Linqi Song. 2026. Merging beyond: Streaming llm updates via activation-guided rotations. *Preprint*, arXiv:2602.03237.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*.

Chenxiang Zhang, Alexander Theus, Damien Teney, Antonio Orvieto, Jun Pang, and Sjouke Mauw. 2026. How does the optimizer implicitly bias the model merging loss landscape? In *The Fourteenth International Conference on Learning Representations*.

Luca Zhou, Daniele Solombrino, Donato Crisostomi, Maria Sofia Bucarelli, Giuseppe Alessio D'Inverno, Fabrizio Silvestri, and Emanuele Rodolà. 2025. On task vectors and gradients. In *UniReps: 3rd Edition of the Workshop on Unifying Representations in Neural Models*.

Luca Zhou, Bo Zhao, Rose Yu, and Emanuele Rodolà. 2026. Demystifying mergeability: Interpretable properties to predict model merging success. In *Forty-third International Conference on Machine Learning*.

A α-stability of Task Vector Mergeability

Table 6 shows the correlation of our mergeability prediction model for Math-Python and Java-Python experts under different Task Arithmetic scaling coefficient α, performed using our learning rate sweep evaluation data. The task vector L2 distance vs. performance correlation coefficient r is **stable, strong, and highly significant for** $\alpha \leq 0.6$ (the practically-relevant range for Task Arithmetic), then vanishes/flips at where the merge is dominated by full task vector addition.

α	**Math-Python** r	**Java-Python** r
0.2	$-0.86^{\ddagger}$	-0.61^{*}
0.4	$-0.86^{\ddagger}$	-0.59^{*}
0.5	$-0.82^{\ddagger}$	-0.54^{*}
0.6	$-0.73^{\ddagger}$	-0.39
0.8	$+0.11$	$+0.33$
1.0	$+0.24$	$+0.22$

$^{*}p < 0.05$, $^{\dagger}p < 0.01$, $^{\ddagger}p < 0.001$

Table 6: α-stability of Task Vector mergeability for Math-Python and Java-Python experts.

B TIES-Merging Mergeability Prediction

We conducted a TIES-merging (Yadav et al., 2023) experiment (density $= 0.5$, weight $= 0.5$) across all 28 expert pairs. Due to compute constraints, we evaluated TIES merges using perplexity (PPL) on domain-specific calibration data. To validate PPL as an evaluation target, we compared it against benchmark-based mergeability on the 28 Task Arithmetic merges: PPL preserves the rank ordering of merge quality (Spearman $\rho = -0.43, p = 0.02$), though the linear relationship is weaker (Pearson $r = -0.24$, not significant). We therefore treat PPL as a coarse ordinal proxy. Table 7 shows the individual mergeability metric correlation with PPL for TIES-Merging. Consistent with 1, task vector geometry metrics are the strongest individual predictors. This indicates that the mergeability prediction framework carries signal beyond Task Arithmetic.

C Expert Training Data Composition

Table 8 shows the data mix ratios for the base pretrain and each domain expert. Each expert upsamples its target domain while maintaining a broad base of general data to prevent catastrophic forgetting.

Metric	**Pearson** r	p**-value**
Task Vector L2 Distance	-0.733	< 0.001
Task Vector Cosine Sim.	0.793	< 0.001
Singular Value Overlap	0.669	< 0.00
Decoder Grad. L2 Distance	-0.515	0.005

Table 7: TIES-merging individual mergeability metric correlation with post-merge performance.

D Full Benchmark Results

Table 9 reports absolute scores across all benchmarks for the base model, individual experts, and all 28 merged pairs.

E Full Mergeability Metrics

Table 10 reports all 9 mergeability metrics computed for each of the 28 expert pairs.

F Per-Benchmark Correlation Results

Table 11 reports per-benchmark cross-validation results, ordered by validation correlation.

G Learning Rate Fine-tuning Results

Figure 6 shows the post-merge performance of Math-Python and Java-Python experts fine-tuned with increasing learning rate.

Category	Source	PT	Math	Know.	Python	Java	JS	PHP	C++	C#
Factual	Wikipedia	8.0	5.1	40.0	4.1	4.1	4.1	4.1	4.1	4.1
Scientific	Arxiv	7.9	8.4	12.5	4.5	4.5	4.5	4.5	4.5	4.5
Technical QA	StackExch.	6.2	4.4	12.3	5.3	5.3	5.3	5.3	5.3	5.3
Web Crawl	CommonCrawl	23.7	6.3	8.5	6.8	6.8	6.8	6.8	6.8	6.8
Web Crawl	C4	24.3	7.3	8.3	6.6	6.6	6.6	6.6	6.6	6.6
Math	Open-Web-Math	10.0	14.6	3.3	8.8	8.8	8.8	8.8	8.8	8.8
Code	GitHub	20.0	4.1	2.7	13.5	13.5	13.5	13.5	13.5	13.4
Code (Synth.)	Target Lang.	0.0	3.5	2.5	42.5	42.5	42.5	42.1	42.1	42.0
Code (SFT)	Synthetic	0.0	3.8	2.7	7.5	7.5	7.5	7.5	7.5	7.5
Math (Synth.)	Math CC	0.0	35.2	2.6	0.0	0.0	0.0	0.0	0.0	0.0

Table 8: Data mix ratios (%) for pretraining (PT) and each domain expert. Target language-specific synthetic code is upsampled to ∼42% for programming language experts.

	Code Generation HumanEval								Code Generation MBXP						Math Reasoning				General Reasoning			Language Understanding			
Model	C#	Python (greedy)	Python (pass@1)	Java	JS	PHP	Ruby	TS	Python (greedy)	Java	JS	PHP	Ruby	TS	GSM8K	MGSM	MATH	MATH Verify	ARC-C	CommonsenseQA	BBH	MMLU	MMLU-Pro	TriviaQA	BoolQ
PT	6.8	9.8	9.5	0.0	6.2	6.8	1.2	7.5	10.0	0.2	9.4	8.8	1.4	11.5	1.4	0.6	2.6	1.2	26.2	19.3	27.2	24.3	11.6	22.8	61.4
C++	11.8	28.1	28.5	8.7	18.6	26.1	10.6	31.7	30.4	18.3	31.1	32.9	13.7	33.1	7.4	1.2	4.7	1.7	26.9	20.0	29.3	25.5	11.2	24.8	62.7
C#	11.8	26.2	26.8	0.6	22.4	22.4	5.0	29.2	27.4	2.6	29.3	28.9	5.4	30.5	7.6	2.1	4.8	2.0	26.4	19.7	29.0	25.8	11.2	25.1	62.1
Java	13.0	26.2	28.3	15.5	21.7	24.2	8.7	28.0	28.0	24.9	25.9	26.5	7.6	31.9	6.8	1.8	5.0	1.8	26.5	20.9	29.1	25.0	11.2	24.7	62.2
JS	13.7	26.2	25.6	2.5	26.1	22.4	13.7	32.3	27.2	8.6	38.0	24.1	19.9	33.9	7.7	1.7	4.9	1.7	24.3	19.6	29.8	25.9	11.6	24.6	62.2
PHP	11.8	26.2	26.3	3.1	19.3	33.5	18.0	25.5	25.4	3.6	27.7	37.0	21.7	26.5	6.0	1.6	4.7	2.3	26.3	20.3	30.0	25.4	11.3	23.9	62.3
Python	12.4	28.7	30.5	8.7	20.5	23.6	14.9	27.3	27.6	26.5	29.3	27.9	25.1	30.3	7.5	1.7	5.4	1.8	25.5	19.6	30.1	25.2	11.7	24.2	62.1
Know.	8.1	14.0	16.2	0.0	14.3	13.7	0.6	9.3	15.4	0.0	23.3	21.5	0.0	21.9	7.0	1.7	6.5	2.6	23.6	20.0	29.5	25.3	12.0	29.8	60.6
Math	7.5	22.6	22.7	1.2	15.5	21.1	4.4	18.6	24.0	2.8	28.9	27.1	10.2	23.9	24.2	4.7	18.0	12.7	35.8	40.5	34.0	32.1	12.8	26.2	62.8
C++-JS	10.6	28.7	27.9	7.5	26.1	21.7	13.0	32.3	30.8	18.9	35.5	32.3	16.9	33.9	8.1	1.7	4.9	1.8	26.8	19.6	30.2	25.3	11.4	25.3	63.0
C++-Know.	6.2	22.0	22.1	0.6	16.8	16.8	1.2	21.7	24.6	0.0	27.7	27.7	0.8	28.9	8.7	1.9	5.4	2.7	26.1	21.1	30.2	25.3	11.7	28.7	62.8
C++-PHP	8.1	26.2	27.6	7.5	22.4	29.2	12.4	27.3	28.2	14.5	30.9	35.3	17.1	28.7	6.8	1.6	4.2	1.8	26.4	20.5	30.3	25.1	11.2	24.8	62.8
C#-C++	13.7	27.4	27.3	8.1	23.6	23.6	9.9	27.3	29.2	12.1	30.5	29.7	11.0	29.9	8.6	1.6	5.1	2.0	27.6	21.0	29.9	25.6	11.4	25.6	62.3
C#-Java	14.9	27.4	26.2	6.2	21.1	23.0	6.8	26.1	26.6	7.8	29.1	28.1	7.0	31.9	8.0	1.5	4.9	2.1	25.0	20.1	29.5	25.2	11.3	25.6	62.3
C#-JS	15.5	25.6	25.6	3.7	26.1	24.2	10.6	33.5	27.8	9.6	34.7	28.7	14.3	32.3	7.7	2.0	5.1	2.0	24.2	19.1	30.1	25.2	11.6	25.5	62.2
C#-Know.	12.4	18.3	20.6	0.0	14.3	18.0	1.2	19.9	21.4	0.2	27.1	24.5	0.0	29.1	6.8	1.9	5.8	3.0	24.2	20.0	29.4	25.5	11.7	28.6	61.6
C#-Math	8.1	23.8	24.4	3.7	21.7	16.8	5.6	28.0	29.6	9.8	29.1	27.3	6.4	29.3	15.6	2.8	8.6	5.1	25.8	22.5	31.7	26.4	11.9	26.8	62.5
C#-PHP	11.2	23.2	25.2	2.5	25.5	26.7	13.0	24.8	25.8	3.4	28.5	34.9	12.5	31.3	6.4	2.0	5.1	2.0	26.9	19.7	29.8	25.7	11.3	25.5	62.0
C#-Python	15.5	27.4	27.5	5.0	25.5	22.4	9.9	31.7	27.8	17.7	28.9	28.3	10.6	30.3	7.6	1.9	5.1	2.1	25.1	19.2	30.5	25.7	11.6	25.0	62.1
Java-C++	11.2	26.8	29.0	15.5	25.5	21.7	7.5	29.8	29.6	21.1	33.1	29.7	7.6	31.9	6.6	1.5	4.7	2.0	25.7	21.7	29.3	24.8	11.2	25.4	62.1
Java-JS	16.2	26.2	27.6	11.8	29.2	19.9	8.7	31.7	30.0	16.9	33.9	25.7	15.1	34.7	7.1	1.9	5.0	1.8	26.3	19.0	30.1	25.3	11.5	25.5	62.2
Java-Know.	9.3	20.1	21.3	0.0	14.9	17.4	1.2	21.7	22.8	0.0	26.9	25.9	0.0	27.3	8.2	1.7	5.9	2.7	25.6	20.3	29.8	25.6	11.6	28.6	62.2
Java-PHP	12.4	23.2	24.7	8.7	26.1	29.8	10.6	25.5	26.2	18.1	29.7	35.1	13.1	31.3	6.7	1.7	4.7	1.9	25.3	20.2	29.2	25.5	11.3	24.9	62.1
Java-Python	14.9	28.1	28.5	11.8	24.2	23.0	13.7	31.1	30.0	23.3	29.5	28.3	12.9	32.5	6.4	1.8	5.1	2.0	24.8	19.1	30.1	24.9	11.6	25.1	62.2
JS-Know.	12.4	20.1	21.2	1.2	19.9	18.0	1.9	19.3	23.2	0.2	29.7	24.7	1.2	28.5	8.8	1.8	5.8	2.8	24.9	19.8	30.2	25.4	12.1	28.5	62.8
JS-PHP	12.4	22.0	24.7	4.4	24.8	28.6	16.8	31.7	27.4	8.4	35.1	32.7	20.3	31.7	6.2	1.8	4.8	1.8	26.3	20.5	30.2	25.6	11.5	24.8	62.4
Math-C++	13.0	23.8	26.6	3.7	19.9	19.3	5.0	24.8	28.4	8.8	30.7	29.5	9.6	29.9	16.9	2.9	8.6	4.8	27.0	22.2	32.1	26.5	11.5	26.7	63.2
Math-Java	13.0	25.6	27.0	6.2	18.0	19.9	4.4	23.6	27.4	13.9	29.7	28.3	3.4	30.3	15.9	2.9	9.1	5.0	27.3	22.9	30.9	25.7	11.5	27.0	62.3
Math-JS	14.3	24.4	26.2	1.9	25.5	19.9	6.2	24.2	28.2	8.6	32.1	27.3	9.4	33.3	16.8	3.4	8.5	5.0	27.0	24.7	32.0	27.0	11.7	26.6	62.6
Math-Know.	8.7	18.9	20.7	0.6	14.3	19.9	0.6	13.0	20.0	0.6	28.3	25.3	2.4	23.7	17.6	2.5	10.2	5.9	24.1	22.9	31.3	25.8	12.1	29.4	62.7
Math-PHP	6.8	20.7	22.9	5.0	15.5	22.4	5.0	23.6	24.6	10.0	29.1	32.1	7.2	28.3	14.6	2.7	8.4	5.2	27.3	22.4	31.6	26.4	11.8	26.4	62.7
Math-Python	13.0	25.6	25.8	2.5	21.1	19.3	3.7	25.5	28.4	6.4	31.3	26.7	9.0	29.5	16.1	2.8	9.0	5.3	26.6	27.0	32.7	26.7	11.9	26.7	63.0
PHP-Know.	8.1	18.3	20.1	1.2	17.4	19.3	2.5	21.1	18.4	0.2	26.1	30.7	0.8	25.3	7.4	1.9	5.6	2.9	26.7	19.9	30.2	25.3	11.7	27.9	61.4
Python-C++	13.0	27.4	28.2	9.9	25.5	24.2	12.4	29.8	29.8	18.9	31.1	29.3	15.9	30.1	8.7	1.3	5.3	1.9	25.3	19.9	29.8	25.4	11.2	25.3	62.7
Python-JS	15.5	26.2	27.0	4.4	24.8	19.9	17.4	35.4	30.2	22.5	34.9	27.5	21.1	32.7	7.9	1.8	5.1	1.9	23.4	20.2	30.7	25.2	11.5	25.0	62.5
Python-Know.	11.8	22.6	22.9	0.6	16.2	18.0	3.7	19.9	26.0	0.8	27.9	24.3	0.4	27.7	8.6	2.1	5.5	2.6	24.0	20.5	30.4	25.6	12.0	27.9	62.4
Python-PHP	12.4	26.8	28.0	5.6	21.1	26.1	18.0	26.7	27.6	15.5	29.5	34.5	19.3	30.3	7.9	1.5	5.1	2.2	26.1	19.6	30.5	25.5	11.5	24.7	62.4

Table 9: Full benchmark results (%) of individual experts and their pairwise merging (28 pairs).

		Task Vector			Activation		Decoder Grad.		Input Grad.	
Model A	**Model B**	L2	Cos	SV Ov.	L2	Cos	L2	Cos	L2	Cos
Math	C++	584.3	0.493	0.997	0.016	−0.561	8.69	0.832	20.87	−0.005
Math	C#	586.2	0.491	0.996	0.051	−0.870	5.77	0.925	27.96	−0.023
Math	Java	574.7	0.504	0.996	0.014	−0.761	6.70	0.898	18.36	−0.010
Math	JS	579.6	0.502	0.996	0.002	−0.869	4.89	0.946	17.71	−0.008
Math	Know.	606.7	0.446	0.999	0.002	−0.912	4.92	0.950	25.25	0.010
Math	PHP	588.1	0.488	0.996	0.001	0.925	5.02	0.943	21.64	0.006
Math	Python	572.7	0.513	0.997	0.012	−0.841	8.46	0.861	20.62	0.001
C++	C#	444.5	0.688	1.000	0.042	0.671	7.15	0.878	23.37	0.242
C++	Java	433.9	0.698	1.000	0.011	0.689	7.83	0.858	15.03	0.149
C++	JS	458.5	0.667	1.000	0.015	0.554	7.92	0.854	15.44	0.005
C++	Know.	592.0	0.436	0.994	0.014	0.555	9.30	0.818	23.69	0.020
C++	PHP	466.7	0.656	1.000	0.015	−0.516	7.95	0.854	19.94	0.005
C++	Python	431.3	0.705	1.000	0.012	0.637	9.39	0.826	17.90	0.094
C#	Java	415.1	0.724	1.000	0.039	0.897	4.22	0.957	19.81	0.494
C#	JS	441.8	0.692	1.000	0.050	0.901	3.52	0.970	23.56	0.049
C#	Know.	591.6	0.438	0.994	0.050	0.889	6.32	0.919	29.78	0.020
C#	PHP	445.4	0.687	1.000	0.051	−0.777	3.56	0.970	27.01	0.003
C#	Python	442.0	0.691	1.000	0.040	0.940	6.96	0.912	22.04	0.336
Java	JS	434.6	0.697	1.000	0.013	0.832	5.13	0.937	11.61	0.041
Java	Know.	581.5	0.448	0.994	0.013	0.807	6.90	0.900	21.72	0.002
Java	PHP	438.7	0.692	1.000	0.014	−0.672	5.14	0.937	17.34	−0.002
Java	Python	430.3	0.702	1.000	0.006	0.885	7.77	0.883	14.43	0.195
JS	Know.	587.3	0.445	0.994	0.000	0.965	5.71	0.933	21.18	0.005
JS	PHP	426.4	0.713	1.000	0.001	−0.749	2.11	0.989	16.50	0.023
JS	Python	431.6	0.705	1.000	0.011	0.893	7.24	0.900	15.07	0.023
Know.	PHP	592.1	0.437	0.994	0.001	−0.815	5.82	0.930	24.61	0.007
Know.	Python	583.8	0.451	0.994	0.010	0.873	8.72	0.856	23.39	0.033
PHP	Python	440.9	0.692	1.000	0.011	−0.750	7.26	0.899	19.57	0.022

Table 10: Full mergeability metrics for all 28 expert pairs. Activation, decoder gradient, and input gradient metrics are computed over 7,140 calibration instances.

Benchmark	**Train** r	**Val.** r	**Rel. Gap**	λ	**Top Metric**
MATH Verify	0.986	0.928	5.9%	0.1	TV L2 Distance
HE Python (pass@1)	0.969	0.894	7.7%	0.01	TV Cosine Sim.
BBH (CoT)	0.833	0.821	1.4%	1	TV L2 Distance
MBXP Ruby	0.962	0.815	15.3%	0.01	TV Cosine Sim.
MATH (CoT)	0.964	0.813	15.7%	0.1	TV L2 Distance
MBXP Java	0.941	0.802	14.8%	0.1	TV Cosine Sim.
TriviaQA	0.921	0.792	14.0%	1	TV L2 Distance
MMLU-Pro	0.899	0.759	15.6%	0.01	TV L2 Distance
GSM8K (CoT)	0.945	0.752	20.4%	0.1	TV L2 Distance
HE Python (greedy)	0.916	0.725	20.9%	10	TV Cosine Sim.
CommonsenseQA	0.774	0.723	6.6%	1	TV L2 Distance
MGSM (CoT)	0.944	0.713	24.5%	0.01	TV L2 Distance
MBXP JavaScript	0.921	0.708	23.1%	0.01	TV Cosine Sim.
HE PHP	0.900	0.696	22.7%	0.1	TV Cosine Sim.
MBXP Python	0.949	0.696	26.7%	0.01	TV Cosine Sim.
HE Ruby	0.954	0.691	27.6%	0.01	TV Cosine Sim.
MMLU	0.872	0.677	22.4%	0.1	TV L2 Distance
HE Java	0.871	0.656	24.7%	0.01	TV Cosine Sim.
HE TypeScript	0.931	0.630	32.3%	0.01	TV Cosine Sim.
MBXP TypeScript	0.937	0.572	39.0%	0.01	TV Cosine Sim.
HE C#	0.683	0.511	25.2%	1	TV Cosine Sim.
HE JavaScript	0.820	0.500	39.0%	1	TV Cosine Sim.
BoolQ	0.708	0.496	29.9%	1	TV L2 Distance
MBXP PHP	0.900	0.401	55.4%	0.01	TV Cosine Sim.
ARC-C (25-shot)	0.814	0.357	56.1%	0.01	TV Cosine Sim.
Mean	0.893	0.685	23.5%	—	—

Table 11: Per-benchmark CV results with the most important metric per benchmark.

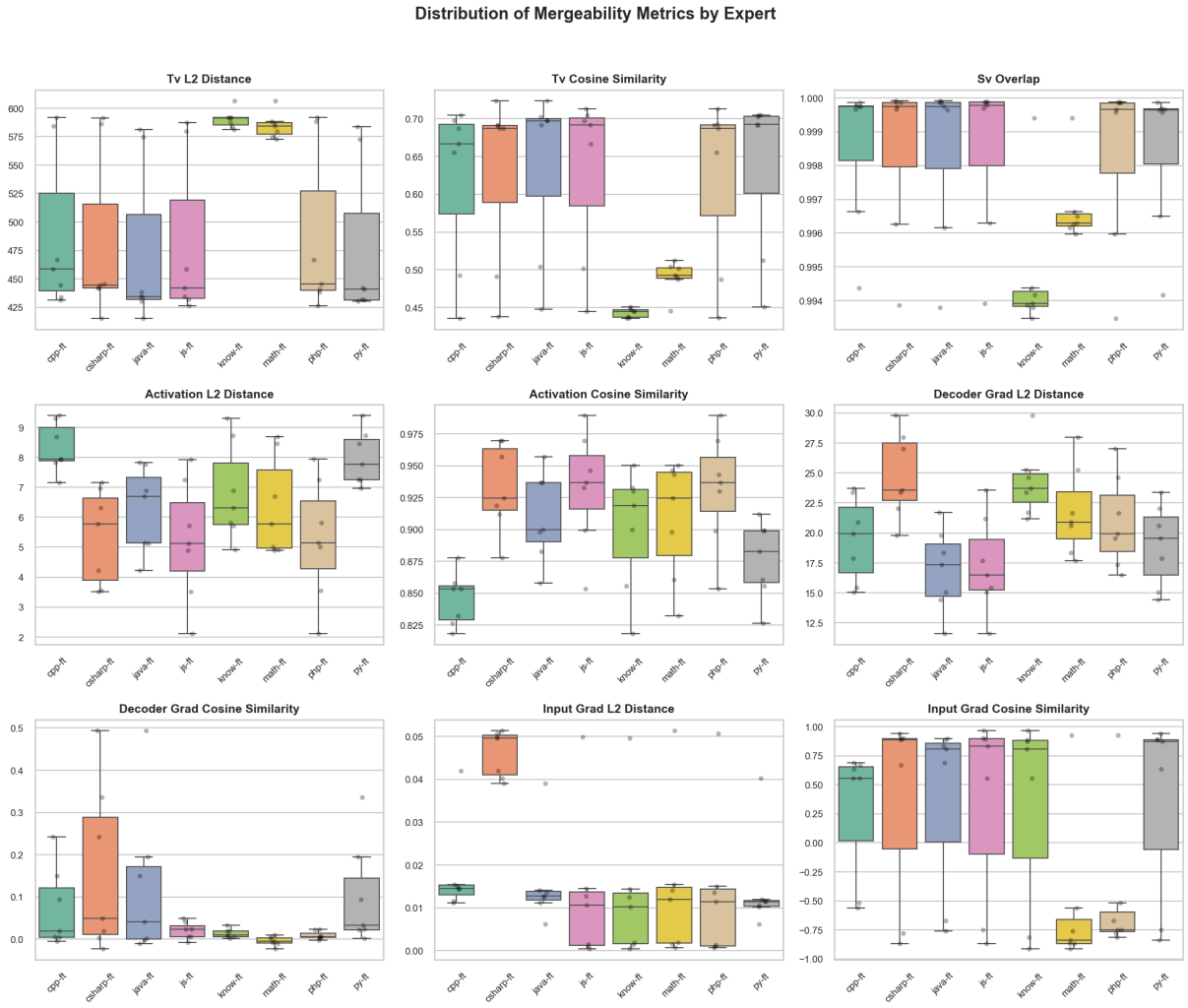


Figure 5: Distributions of mergeability metrics by expert.

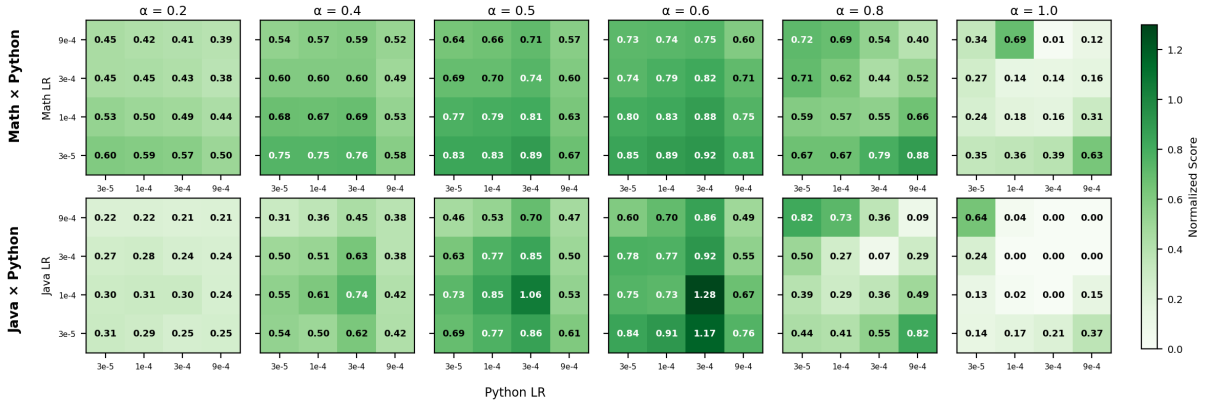


Figure 6: Normalized post-merge performance of Math-Python (top) and Java-Python (bottom) experts merged with different scaling coefficients α and fine-tuned with different learning rates.