SMAT: Simple and Efficient Merge-Aware Training

Anonymous authors
Paper under double-blind review

Abstract

Model merging integrates the capabilities of multiple experts without joint retraining, but standard expert training optimizes task loss alone and does not guarantee good performance after merging. Merge-aware training (MAT) aims to improve merged performance, but existing methods do not fully account for common merging operations and add training cost. We observe that, from an expert's perspective, common merging methods can be described by three operations: Scale reweights its own update, Mask removes selected coordinates, and Perturb adds updates from other experts. Based on this view, we introduce SMAT (Simple MAT), which jointly optimizes expert loss and expected loss at simulated merged parameters generated by sampling scaling coefficients, masks, and additive noise. We further introduce periodic scheduling, kernel fusion, and parameter storage switching to make SMAT efficient, with one forward and one backward pass per step. Across four language and vision-language backbones, SMAT improves the mean score across five merging methods by 1.07–2.16 points over the strongest baseline for each backbone, with less than 2% training-time overhead over standard fine-tuning. Our code is available at `https://anonymous.4open.science/r/smat-6a27/`.

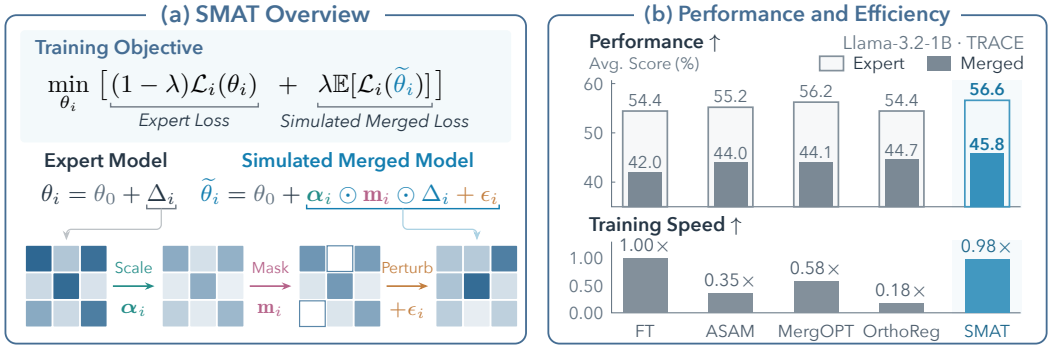


Figure 1: **Overview of SMAT.** (a) SMAT simulates merged parameter states during expert training through Scale, Mask, and Perturb. (b) This improves merged performance while retaining training speed close to standard fine-tuning.

1 Introduction

Model merging combines parameter updates from independently fine-tuned experts that share a pre-trained initialization, integrating their capabilities without joint retraining (Ilharco et al., 2023). Standard fine-tuning, however, optimizes only each expert's task loss, which does not guarantee good performance after merging. Overlapping updates and conflicting signs can reduce task performance after merging (Yadav et al., 2023).

Merge-aware training (MAT) trains experts to perform well after merging. SAFT (Lee et al., 2025) trains for low loss near the expert's weights; MergOPT (Yang et al., 2026) adds noise to simulate merging; and OrthoReg (Liu et al., 2026) encourages orthogonality within update matrices. However, these methods do not fully consider common merging operations, especially scaling and masking an

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.

expert's own update. They also increase training cost through extra passes, perturbation processing, or matrix regularization.

We observe that Task Arithmetic (Ilharco et al., 2023), TIES-Merging (Yadav et al., 2023), DARE (Yu et al., 2024), and DELLA (Deep et al., 2024) can be described by three operations from an expert's perspective: *Scale* reweights its own update, *Mask* removes selected coordinates, and *Perturb* adds updates from other experts. Let θ_0 be the pretrained parameters, θ_i the parameters of expert i, and $\Delta_i = \theta_i - \theta_0$ its task vector. From expert i's view, the merged parameters are

$$\theta_{\text{merge}} = \theta_0 + \boldsymbol{\alpha}_i \odot \mathbf{m}_i \odot \Delta_i + \epsilon_i, \qquad \epsilon_i = \sum_{j \neq i} \boldsymbol{\alpha}_j \odot \mathbf{m}_j \odot \Delta_j. \tag{1}$$

Here $\boldsymbol{\alpha}_i$ gives the merging weights, $\mathbf{m}_i$ is the mask that selects retained updates, and ϵ_i is the combined update from other experts; $\odot$ denotes coordinatewise multiplication.

Motivated by these three merging operations, we introduce SMAT (Simple MAT), illustrated in Figure 1. Its objective jointly optimizes the expert loss and the expected loss at simulated merged parameters to improve both expert and merged performance. Following Eq. (1), SMAT samples scaling coefficients and masks for its own update and noise for other experts' updates.

We further introduce three improvements for efficient SMAT training. *Periodic scheduling* applies the simulated merged loss every few steps and the expert loss otherwise, keeping one forward and one backward pass per step. *Kernel fusion* computes Scale, Mask, and Perturb together to reduce repeated parameter reads and writes. *Parameter storage switching* keeps the original and simulated weights in separate buffers and switches between them to avoid repeated weight copying.

Our experiments cover language and vision-language models ranging from 151M to 8B parameters, evaluated with five merging methods. Across all four backbones, SMAT achieves better merged performance and shorter training time than the compared MAT methods, with less than 2% overhead over standard fine-tuning. For Llama-3.2-1B-Instruct, SMAT improves merged performance by 1.12 points over the strongest baseline, OrthoReg, while using 82% less training time. Further analysis shows that SMAT broadens low-loss regions along merge-relevant directions, providing a loss-smoothing perspective on its improved robustness to merging.

We summarize our three contributions:

❶ We formulate merge-aware training through Scale, Mask, and Perturb, simulating these common merging operations during expert training.

❷ We implement this objective with one forward and one backward pass per step, using periodic scheduling and efficient parameter operations.

❸ We demonstrate improved merging across language and vision-language models with less than 2% training-time overhead, examine merging behavior, and extend SMAT to Muon.

2 Related Work

Model Merging. Stochastic Weight Averaging (Izmailov et al., 2018), Model Stock (Jang et al., 2024), and Model Ratatouille (Ramé et al., 2023) show that weight averaging can improve generalization. Model Soups (Wortsman et al., 2022a) averages model weights, while Fisher Merging (Matena & Raffel, 2022) gives more important parameters more weight. Task Arithmetic (Ilharco et al., 2023) combines updates from a shared initialization, and AdaMerging (Yang et al., 2024) learns coefficients for tasks or layers. TIES-Merging (Yadav et al., 2023), DARE (Yu et al., 2024), DELLA (Deep et al., 2024), and Model Breadcrumbs (Davari & Belilovsky, 2024) use magnitude, sign agreement, or random deletion to select updates. Other methods learn shared masks or retain task-specific masks and rescalers (Tang et al., 2023; Wang et al., 2024; Huang et al., 2024). RegMean (Jin et al., 2023) uses linear regression to preserve layer outputs, while TSV-Merge (Gargiulo et al., 2025) uses subspaces defined by the singular vectors of task updates. These methods change task vectors during merging. SMAT simulates their scaling, masking, and additive changes during expert training.

Merge-Aware Training. Merge-aware training (MAT) changes expert training to improve performance after merging. Tangent-space Task Arithmetic (Ortiz-Jimenez et al., 2023) uses linearized

fine-tuning for this purpose. SAFT (Lee et al., 2025) uses sharpness-aware fine-tuning to find locally flatter solutions, building on SAM (Foret et al., 2021) and ASAM (Kwon et al., 2021). MergOPT (Yang et al., 2026) trains at sampled merge offsets, and OrthoReg (Liu et al., 2026) encourages orthogonality within update matrices. However, these methods do not fully consider common merging operations, particularly the scaling and masking of an expert's own task vector. They also incur extra computation through additional gradient evaluations, perturbation processing, or matrix regularization. SMAT trains experts under these changes by simulating Scale, Mask, and Perturb. Periodic scheduling and fused operations reduce overhead.

3 Preliminaries

3.1 Problem Setup

We consider N tasks with a shared pretrained initialization $\theta_0 \in \mathbb{R}^d$, where d is the number of trainable shared parameters that we merge and perturb. Expert i trains on its data distribution $\mathcal{D}_i$, without access to other experts' data or checkpoints.

Let x and y be an input and its target, $f_i(x;\theta)$ expert i's predictor with parameters θ, and ℓ_i its per-example loss. Its expected task loss is

$$\mathcal{L}_i(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}_i}\big[\ell_i(f_i(x;\theta), y)\big], \tag{2}$$

where $\mathbb{E}$ averages over the indicated distribution. Standard fine-tuning minimizes $\mathcal{L}_i$ to obtain θ_i for each expert. The shared parameters are then merged; any task-specific heads stay separate.

3.2 Task Vectors and Model Merging

Expert i's task vector is $\Delta_i = \theta_i - \theta_0$, where θ_i denotes its fine-tuned parameters. Equation (1) weights and masks task vectors before adding them to the shared initialization. The weights $\boldsymbol{\alpha}_j$ can vary by parameter, and both weights and masks can depend on all experts. This form covers Model Soups (Wortsman et al., 2022a), Fisher Merging (Matena & Raffel, 2022), and Task Arithmetic (Ilharco et al., 2023), as well as masking in TIES (Yadav et al., 2023), DARE (Yu et al., 2024), and DELLA (Deep et al., 2024).

We use γ for a scalar merging coefficient and $\mathbf{1}$ for the all-ones vector. Task Arithmetic sets $\boldsymbol{\alpha}_j = \gamma\mathbf{1}$ and $\mathbf{m}_j = \mathbf{1}$; weight averaging instead uses $\boldsymbol{\alpha}_j = \mathbf{1}/N$.

Relative to expert i, merging changes the parameters by

$$\theta_{\text{merge}} - \theta_i = (\boldsymbol{\alpha}_i \odot \mathbf{m}_i - \mathbf{1}) \odot \Delta_i + \epsilon_i. \tag{3}$$

Scale and Mask change the expert's own update, while Perturb represents the other experts' contribution ϵ_i. SMAT simulates these three changes during independent training.

3.3 The MAT Objective

Training should preserve each expert's performance while improving the merged model. We express this goal with the MAT objective

$$\mathcal{J}_i^{\text{MAT}} = (1-\lambda)\mathcal{L}_i(\theta_i) + \lambda\mathcal{L}_i(\theta_{\text{merge}}), \qquad 0 \le \lambda \le 1. \tag{4}$$

Here $\mathcal{J}_i^{\text{MAT}}$ is expert i's training objective, and λ weights the merged loss. The endpoints $\lambda = 0$ and $\lambda = 1$ train only for expert performance and merged performance, respectively.

4 Method

4.1 Simulating Merged Parameter States

Following Eq. (1), SMAT simulates merged parameters during expert training as

$$\widetilde{\theta}_i = \theta_0 + \boldsymbol{\alpha}_i \odot \mathbf{m}_i \odot \Delta_i + \epsilon_i. \tag{5}$$

We use $\boldsymbol{\alpha}_i = \alpha_i\mathbf{1}$, with one scalar coefficient shared by all coordinates. Here $\mathbf{m}_i$ is a sampled mask, and ϵ_i is sampled noise representing the combined updates from other experts. We sample α_i, $\mathbf{m}_i$, and ϵ_i independently. Figure 2 illustrates these operations.

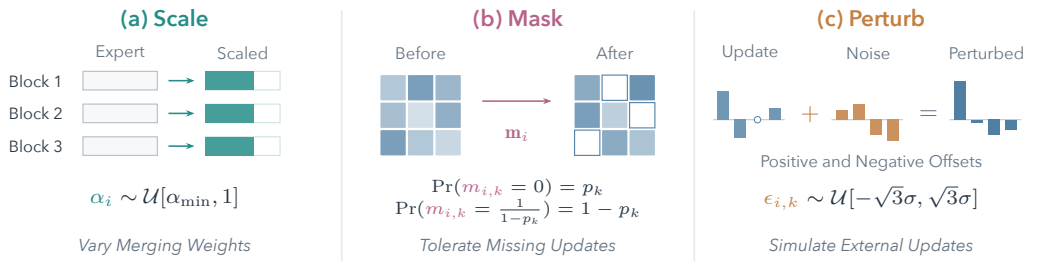


Figure 2: **Simulating merged updates.** Scale reweights the task vector, Mask drops and rescales its coordinates, and Perturb adds uniform noise to simulate other experts' updates.

Scale. Merging may assign different weights to an expert's task vector, as in WiSE-FT (Wortsman et al., 2022b) and Task Arithmetic (Ilharco et al., 2023). Scale samples a shared coefficient $\alpha_i \sim \mathcal{U}[\alpha_{\min}, 1]$, where $\mathcal{U}$ is the uniform distribution and $0 \leq \alpha_{\min} \leq 1$ is the smallest allowed coefficient.

Mask. Mask prepares the expert for coordinate removal using the random dropping and rescaling of DARE (Yu et al., 2024). This is related to stochastic masking in Dropout (Srivastava et al., 2014) and DropConnect (Wan et al., 2013), and to mixing weights with their pretrained values in Mixout (Lee et al., 2020). Let $\mathcal{S}$ be the set of coordinates to mask: by default, linear weights in Transformer attention and MLP blocks. Let p be their drop probability. For coordinate k, set $p_k = p$ inside $\mathcal{S}$ and $p_k = 0$ outside it, and draw

$$m_{i,k} = \begin{cases} 0, & \text{with probability } p_k, \\ \dfrac{1}{1-p_k}, & \text{with probability } 1 - p_k, \end{cases} \qquad 0 \leq p < 1. \tag{6}$$

Here $m_{i,k}$ is the k-th mask entry and $\mathbf{p}$ collects the drop probabilities. Masks are independent across coordinates; biases, normalization parameters, and embeddings are excluded. Since $\mathbb{E}[m_{i,k}] = 1$, rescaling preserves the expected scaled update. Mask acts only on the task vector, leaving pretrained weights and additive noise unchanged.

Perturb. Perturb simulates other experts' additive updates ϵ_i in Eq. (1). Their checkpoints and task vectors are unavailable during independent training, so we approximate the unknown update with random parameter noise.

With Perturb alone on Llama-3.2-1B-Instruct with AdamW, we compare uniform, Gaussian, and Laplace noise at matched root mean square (RMS) (Figure 3). Their trends are broadly similar, and uniform gives the highest merged performance when RMS $= 2 \times 10^{-3}$. Laplace, motivated by task-vector statistics in MergOPT (Yang et al., 2026), has no consistent advantage. This suggests that exposure to additive parameter variation may matter more than precise distribution matching in this setting.

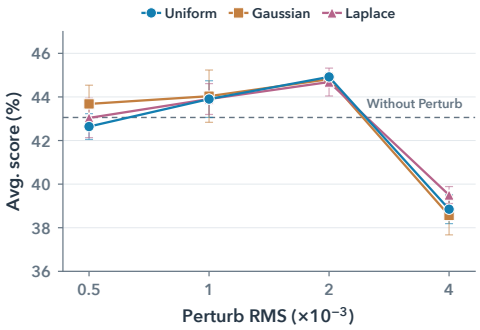


Figure 3: Perturb alone on Llama-1B (AdamW, $t = 4$). Five-merger means $\pm$ SD.

This observation motivates a robustness view through loss smoothing (Wen et al., 2018). With Scale and Mask fixed, the expected perturbed loss averages task loss over nearby parameters, encouraging tolerance to additive merge offsets. Appendix C expresses this as convolution; under its smoothness assumptions, the leading small-noise correction depends on local curvature. Perturb may therefore improve merge robustness without matching the full distribution of other experts' updates. We independently draw $\epsilon_{i,k} \sim \mathcal{U}[-\sqrt{3}\sigma, \sqrt{3}\sigma]$ at each trainable coordinate, with fixed RMS σ, zero mean, and variance σ^2.

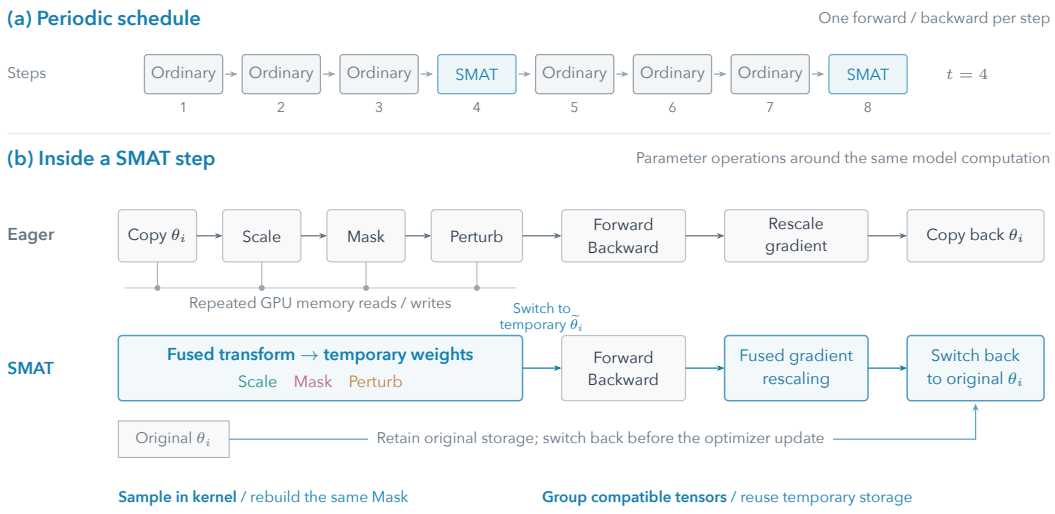


Figure 4: **Efficient SMAT training.** Each period of $t = 4$ steps includes one SMAT update. Fused kernels and storage switching reduce parameter overhead.

4.2 A Practical Training Objective

We approximate the merged loss in Eq. (4) by an expectation over simulated parameters:

$$\mathcal{J}_i^{\text{SMAT}} = (1-\lambda)\mathcal{L}_i(\theta_i) + \lambda \mathbb{E}_{\alpha_i, \mathbf{m}_i, \epsilon_i}\left[\mathcal{L}_i(\widetilde{\theta}_i)\right]. \tag{7}$$

Here $\mathcal{J}_i^{\text{SMAT}}$ is the training objective; λ weights the simulated loss, and the expectation averages the three random draws. At each simulated-loss update, one sampled state provides a stochastic estimate of this expectation, so training does not enumerate merging configurations or expert combinations.

How closely this objective represents actual merging depends on the sampled states. Appendix C relates the loss difference to differences in the state means and second moments.

Gradient Analysis. Let B be a sampled minibatch and $\mathcal{L}_{i,B}$ its average task loss. With the three draws held fixed, the chain rule gives the gradient (∇) with respect to the expert parameters θ_i:

$$\nabla_{\theta_i} \mathcal{L}_{i,B}(\widetilde{\theta}_i) = \alpha_i \mathbf{m}_i \odot \nabla_{\widetilde{\theta}_i} \mathcal{L}_{i,B}(\widetilde{\theta}_i). \tag{8}$$

The two factors have a simple meaning: $\alpha_i \mathbf{m}_i$ is expert i's participation coefficient at each coordinate, and the second factor is the simulated merged model's gradient on task i. SMAT simulates merged weights with Scale, Mask, and Perturb, then uses the participation coefficient to weight the gradient passed back to the expert. If $\alpha_i = 0$ or $m_{i,k} = 0$, the expert's k-th task-vector coordinate does not contribute to the simulated merged model and receives zero gradient from the simulated loss.

4.3 Scheduling and Efficient Implementation

Directly optimizing Eq. (7) evaluates both losses at every step. These mixed-loss updates can instead be approximated by separate updates on each loss, with step counts proportional to their weights. Let $\mathcal{L}_A$ denote the expert loss and $\mathcal{L}_B$ the expected simulated loss.

Proposition 1 (Replacing mixed-loss updates) *Let $J = (1-k/n)\mathcal{L}_A + (k/n)\mathcal{L}_B$, with integers $n \geq 1$ and $0 \leq k \leq n$. Starting from the same parameters and using a constant step size $\eta > 0$, replace n full-gradient descent steps on J with $n-k$ steps on $\mathcal{L}_A$ and k steps on $\mathcal{L}_B$ in any order. If both loss gradients are L_g-Lipschitz and bounded in norm by G in a neighborhood containing both trajectories, the resulting parameters satisfy*

$$\|\theta_n^{\text{mixed}} - \theta_n^{\text{separate}}\|_2 \leq \eta^2 L_g G\, n(n-1). \tag{9}$$

Thus, mixed-loss training can be replaced to first order by a short sequence of single-loss steps. Appendix D gives the proof and SMAT's cycle-averaged gradient bound. This motivates periodic scheduling; kernel fusion and parameter storage switching further reduce its overhead (Figure 4).

Periodic scheduling. Each cycle has $t - 1$ expert-loss updates and one simulated-loss update, matching $\lambda = 1/t$ in Eq. (7). We use $t = 4$, corresponding to three expert-loss steps and one simulated-loss step. Each step computes only one loss, using one forward and one backward pass.

Kernel fusion. Separate Scale, Mask, and Perturb operations repeatedly access parameter-sized tensors. Two Triton kernels fuse weight simulation and gradient rescaling (Eq. (8)). They sample noise and masks and group compatible tensors, reducing memory traffic and kernel launches.

Parameter storage switching. SMAT uses simulated weights in a reusable buffer for the forward and backward pass, then restores the original storage for the optimizer update. Frozen base weights are prefetched from pinned CPU memory into this buffer during ordinary steps. This needs one additional GPU buffer the size of the trainable weights.

5 Experiments

5.1 Experimental Setup

Models and Tasks. We evaluate Llama-3.2-1B-Instruct and Llama-3.1-8B-Instruct from the Llama 3 family (Grattafiori et al., 2024) on seven TRACE (Wang et al., 2023) tasks: C-STANCE, FOMC, MeetingBank, ScienceQA, NumGLUE-cm, NumGLUE-ds, and 20Minuten. For vision, we evaluate CLIP (Radford et al., 2021) with ViT (Dosovitskiy et al., 2021) encoders (ViT-B/32 and ViT-L/14) on Cars, DTD, EuroSAT, GTSRB, MNIST, RESISC45, SUN397, and SVHN, following Task Arithmetic (Ilharco et al., 2023).

Compared Methods. We compare SMAT with standard fine-tuning (FT), ASAM (Kwon et al., 2021) implemented as in SAFT (Lee et al., 2025), MergOPT (Yang et al., 2026), and OrthoReg (Liu et al., 2026). Each expert set is merged using weight averaging (WA), Task Arithmetic (Ilharco et al., 2023) (TA), TIES (Yadav et al., 2023), DARE (Yu et al., 2024) followed by TA, and DELLA (Deep et al., 2024).

Training. Within each backbone, all methods share data, initialization, learning rate, batch size, and training budget. For language models, we follow the base training settings of MergOPT (Yang et al., 2026): AdamW (Loshchilov & Hutter, 2019) with learning rate 2×10^{-5}, batch size 8, and task-specific epoch counts. For vision models, we follow the public CLIP fine-tuning configuration of FusionBench (Tang et al., 2025): Adam (Kingma & Ba, 2015) with learning rate 10^{-5}, batch size 128, and 4,000 updates per task. We update all language-model parameters and only the CLIP vision encoder. SMAT uses fused operations and simulates every $t = 4$ steps, with Scale sampled from $\mathcal{U}[\alpha_{\text{min}}, 1]$, and Mask probability $p = 0.5$ on attention and MLP linear weights. Both language backbones use $(\alpha_{\text{min}}, \sigma) = (0.2, 2 \times 10^{-3})$; both vision backbones use $(0.1, 10^{-3})$, where α_{min} is the Scale lower bound and σ is the Perturb RMS. Each model with a MAT method is trained and evaluated three times using different random seeds, and the average performance is reported as the experimental result.

Dev Search. All methods share a development (dev) split held out from training data. The dev and final evaluation splits have no overlapping examples. We use dev data to select merging configurations. For each backbone and training method, TA, TIES, DARE, and DELLA select $\gamma \in \{0.1, 0.2, \ldots, 1.0\}$ by mean dev score, breaking ties in favor of smaller values. WA requires no search. On TRACE, the top two candidates are ranked by their mean over three decoding repeats. Sparsity is calibrated on FT dev results and shared within each model family (Appendix A).

Evaluation. Scores are task means on a 0–100 scale: ROUGE-L (Lin, 2004) for MeetingBank, SARI (Xu et al., 2016) for 20Minuten, and accuracy otherwise. We report final performance on the released eval split for language, with three decoding repeats per checkpoint, and on the test split for vision. Expert scores evaluate each expert on its own task; Avg. is the mean merged score over five mergers. Appendix A gives the data splits and configuration selection rules.

Table 1: TRACE results with AdamW. Bold marks the best score for each model.

Method	Expert ↑	Merged Model Performance ↑						Training cost ↓	
		WA	TA	TIES	DARE	DELLA	**Avg.**	Time (×FT)	Mem. (GiB)
Llama-3.2-1B-Instruct									
FT	54.43	39.88↑0.0	42.64↑0.0	42.93↑0.0	43.50↑0.0	41.17↑0.0	42.02↑0.0	1.000	20.0↑0.00%
ASAM	55.20	38.05↓1.8	**46.29**↑3.6	45.48↑2.5	45.71↑2.2	44.56↑3.4	44.02↑2.0	2.869	22.3↑11.5%
MergOPT	56.23	39.67↓0.2	45.62↑3.0	45.12↑2.2	45.59↑2.1	44.54↑3.4	44.11↑2.1	1.718	22.3↑11.5%
OrthoReg	54.44	**43.85**↑4.0	45.77↑3.1	45.93↑3.0	45.16↑1.7	42.57↑1.4	44.66↑2.6	5.693	28.2↑41.1%
SMAT	**56.61**	42.59↑2.7	46.11↑3.5	**47.17**↑4.2	**46.90**↑3.4	**46.09**↑4.9	**45.77**↑3.7	1.019	22.3↑11.5%
Llama-3.1-8B-Instruct									
FT	**62.88**	55.43↑0.0	56.03↑0.0	55.94↑0.0	56.46↑0.0	56.01↑0.0	55.97↑0.0	1.000	30.7↑0.00%
ASAM	62.05	57.09↑1.7	56.88↑0.9	57.61↑1.7	57.01↑0.6	56.60↑0.6	57.04↑1.1	2.676	43.3↑40.9%
MergOPT	61.24	56.23↑0.8	57.46↑1.4	57.97↑2.0	57.53↑1.1	56.63↑0.6	57.16↑1.2	1.543	38.2↑24.3%
OrthoReg	58.72	56.75↑1.3	58.19↑2.2	56.91↑1.0	57.90↑1.4	57.33↑1.3	57.42↑1.4	11.58	61.5↑100.%
SMAT	61.23	**57.99**↑2.6	**58.72**↑2.7	**58.44**↑2.5	**58.54**↑2.1	**58.75**↑2.7	**58.49**↑2.5	1.014	38.2↑24.3%

Expert: mean score before merging. Avg.: mean over five mergers. Mem.: peak GPU memory. Arrows: change from FT in score points or memory %.

Table 2: CLIP results with Adam. Bold marks the best score for each model.

Method	Expert ↑	Merged Model Performance ↑						Training cost ↓	
		WA	TA	TIES	DARE	DELLA	**Avg.**	Time (×FT)	Mem. (GiB)
CLIP ViT-B/32									
FT	89.26	67.22↑0.0	71.13↑0.0	73.05↑0.0	71.21↑0.0	71.75↑0.0	70.87↑0.0	1.000	5.8↑0.00%
ASAM	**90.61**	67.57↑0.4	72.78↑1.6	75.05↑2.0	72.78↑1.6	73.68↑1.9	72.37↑1.5	1.730	6.1↑5.67%
MergOPT	90.05	68.13↑0.9	73.15↑2.0	74.82↑1.8	72.91↑1.7	73.84↑2.1	72.57↑1.7	1.146	6.1↑5.67%
OrthoReg	89.36	70.74↑3.5	72.82↑1.7	74.37↑1.3	72.85↑1.6	70.11↓1.6	72.18↑1.3	1.125	6.5↑13.8%
SMAT	89.27	**73.83**↑6.6	**74.18**↑3.1	**75.31**↑2.3	**74.18**↑3.0	**74.81**↑3.1	**74.46**↑3.6	1.010	6.1↑5.65%
CLIP ViT-L/14									
FT	93.97	80.27↑0.0	84.97↑0.0	85.51↑0.0	84.95↑0.0	84.29↑0.0	84.00↑0.0	1.000	65.1↑0.00%
ASAM	**94.28**	81.58↑1.3	86.45↑1.5	87.18↑1.7	86.14↑1.2	85.85↑1.6	85.44↑1.4	2.000	66.2↑1.73%
MergOPT	94.02	80.86↑0.6	86.38↑1.4	87.29↑1.8	86.38↑1.4	85.77↑1.5	85.34↑1.3	1.025	66.2↑1.73%
OrthoReg	93.87	84.61↑4.3	86.67↑1.7	86.53↑1.0	86.29↑1.3	83.99↓0.3	85.62↑1.6	1.035	67.9↑4.32%
SMAT	93.66	**87.05**↑6.8	**87.72**↑2.7	**88.21**↑2.7	**87.69**↑2.7	**88.21**↑3.9	**87.78**↑3.8	1.002	66.2↑1.73%

Expert: mean score before merging. Avg.: mean over five mergers. Mem.: peak GPU memory. Arrows: change from FT in score points or memory %.

5.2 Main Results

Language Models. Table 1 shows that SMAT achieves Avg. scores across five mergers of 45.77 on Llama-1B and 58.49 on Llama-8B, exceeding the strongest baseline, OrthoReg, by 1.12 and 1.07 points, respectively. It leads three of five merging columns on 1B and all five on 8B. Expert performance improves over FT on 1B but decreases on 8B, showing that better merged performance need not coincide with a better independent expert.

Vision-Language Models. In Table 2, SMAT leads all five merging columns on both CLIP backbones. Its Avg. across five mergers is 74.46 on ViT-B/32, 1.89 points above MergOPT (72.57), and 87.78 on ViT-L/14, 2.16 points above OrthoReg (85.62), the strongest baseline in each setting. We use the same SMAT hyperparameters for both model sizes.

Training Efficiency. With frozen-base offload, SMAT adds about 0.2–1.9% training-time overhead across the four backbones, while increasing peak GPU memory by 1.7–24.3%, as shown in Tables 1 and 2. We measure expert training loops; the speed in Figure 1 is FT time divided by each method's time. Appendix E gives absolute costs and the software difference affecting ASAM timing.

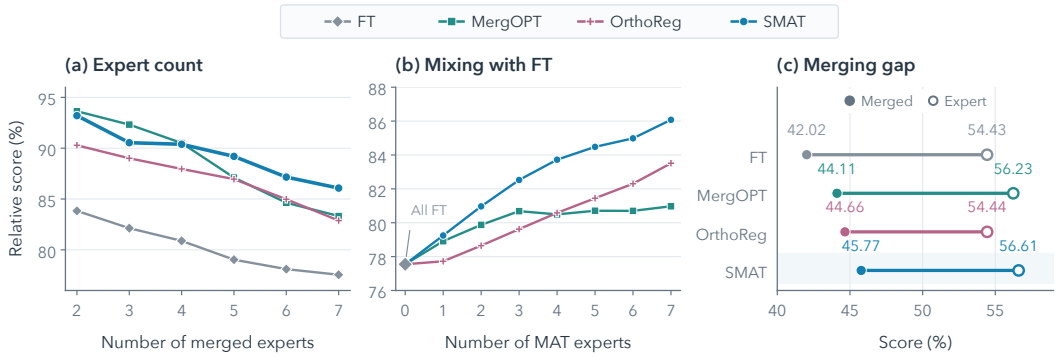


Figure 5: Llama-1B expert merging with AdamW: (a) varying expert count, (b) mixing MAT and FT experts, and (c) expert scores (open) versus mean merged scores (filled). Relative scores in (a,b) are normalized to FT experts.

5.3 Analysis

We study SMAT's generalization and underlying mechanisms on Llama-1B using TRACE.

Generalization to Muon. To examine whether SMAT's gains extend beyond AdamW, we evaluate it with Muon on Llama-1B. SMAT achieves an average score of 44.15 across five merging methods, outperforming OrthoReg by 0.74 points and standard fine-tuning by 5.90 points, while leading on four of the five mergers. Its independent-expert performance remains close to FT (55.07 vs. 55.31), and its measured training time is $1.00\times$ that of FT. These results show that SMAT's merging gains and low training overhead extend to Muon in this setting. Full settings and results are provided in B.2.

Table 3: TRACE with Muon on Llama-1B. Bold marks the best score.

Method	Expert ↑	**Avg.** ↑	Time ↓ (×FT)
FT	**55.31**	38.25↑0.0	1.00
ASAM	54.69	40.34↑2.1	1.66
MergOPT	54.94	40.22↑2.0	1.25
OrthoReg	50.89	43.41↑5.2	2.87
SMAT	55.07	**44.15**↑5.9	1.00

Avg.: mean over five mergers. Arrows: change from FT. Time: training time relative to FT.

Effect of the Number of Experts. We merge $K = 2, \ldots, 7$ experts with TA for FT, MergOPT, OrthoReg, and SMAT, keeping each method's Table 1 TA coefficient. Seven cyclic subsets balance task inclusion for $K < 7$; $K = 7$ uses all tasks. We normalize scores by the corresponding FT expert scores, then average over tasks and subsets (Appendix B.1). SMAT performs similarly to MergOPT with fewer experts and leads all three baselines at $K = 5, 6, 7$ in Figure 5(a). Its normalized score decreases from 93.20% with two experts to 86.08% with all seven. SMAT's advantage becomes clearer with more experts, suggesting greater robustness to the combined updates in larger expert sets.

Compatibility with Standard Experts. Figure 5(b) replaces q of seven FT experts with MergOPT, OrthoReg, or SMAT experts for the same tasks. We use identical balanced replacement sets and Table 1 training configurations. All curves start from all-FT and fix the TA coefficient at $\gamma = 0.3$. The normalized score rises from 77.55% for all-FT to 80.97%, 83.51%, and 86.08% for all-MergOPT, all-OrthoReg, and all-SMAT, respectively. SMAT's curve rises with each replacement count and remains above the other curves for $q > 0$. Thus, SMAT can improve an existing FT expert set through partial replacement, with larger gains than the other methods under this common TA coefficient.

Sources of Merging Gains. Figure 5(c) compares expert scores with five-merger averages from Table 1; their difference is the merging drop. Relative to FT and MergOPT, SMAT has both a higher expert score and a smaller drop. Relative to OrthoReg, its higher expert score offsets a larger drop. A smaller merging drop alone therefore does not determine the value of a training method: SMAT achieves the best mean merged performance by balancing expert quality and retention after merging.

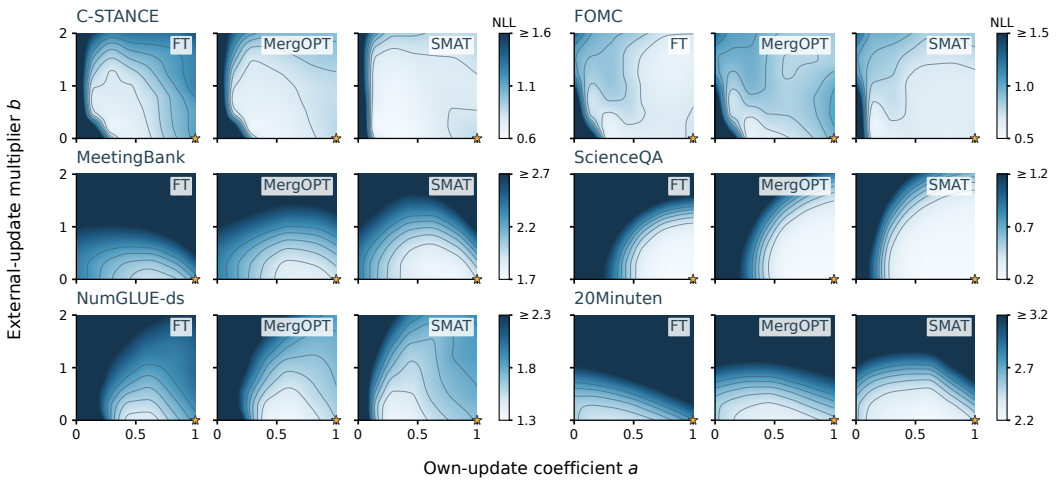


Figure 6: Loss slices for six tasks using the experts in Table 1. Columns are FT, MergOPT, and SMAT; stars mark independent experts. Each task shares one NLL color scale across methods; scales differ across tasks, and contours are spaced by 0.1.

Loss Along Merge Directions. Figure 6 probes answer-token negative log-likelihood (NLL) along two merge directions: rescaling an expert's own update and adding updates from other tasks. For target task i and training method r, let $\Delta_i^{(r)}$ be its update and u_i^{FT} the shared FT partner update. We scale them by a and b, respectively, to define

$$\theta_i^{(r)}(a,b) = \theta_0 + a\Delta_i^{(r)} + bu_i^{\mathrm{FT}}, \qquad u_i^{\mathrm{FT}} = \gamma \sum_{j \neq i} \Delta_j^{\mathrm{FT}}, \quad \gamma = 0.3. \tag{10}$$

We use the Table 1 FT, MergOPT, and SMAT expert sets, with identical FT partner updates and evaluation samples. The point $(1, 0)$ recovers the independent expert. SMAT shows broader low-loss regions than FT and MergOPT across the six tasks. These regions are consistent with Perturb's smoothing of the conditional training loss (Appendix C), supporting loss smoothing as a mechanism for improved merging along the tested directions.

Ablation Studies. We ablate Scale, Mask, and Perturb individually with AdamW in Table 4, keeping the training budget and merger settings fixed. Removing Scale, Mask, or Perturb lowers the Avg. across five mergers by 1.25, 1.07, and 2.10 points, respectively. All three operations contribute, with Perturb having the largest effect. Its importance is consistent with the loss-smoothing interpretation above, while Scale and Mask provide further gains.

Table 4: Component ablation on Llama-1B with AdamW. Avg. averages five mergers; Change is relative to Full SMAT.

Variant	**Avg.** ↑	Change
Full SMAT	45.77	–
w/o Scale	44.52	−1.25
w/o Mask	44.70	−1.07
w/o Perturb	43.67	−2.10

6 Conclusion

We introduced SMAT, a simple and efficient merge-aware training method that prepares independently trained experts for model merging by simulating three common operations, i.e., Scale, Mask, and Perturb, with efficient periodic training. Across language and vision-language models, SMAT consistently improves merged performance across multiple merging methods with less than 2% training-time overhead. Our analyses further reveal broader low-loss regions along merge-relevant directions, supporting loss smoothing as a contributing mechanism to SMAT's merge robustness, while Scale and Mask provide complementary gains beyond Perturb. Overall, our results highlight merge-induced parameter variation as a useful training signal for building experts that remain effective after merging.

AI USE STATEMENT

In this work, we used generative AI tools to assist with language editing of the manuscript, including polishing wording, improving readability, and correcting grammar. We also used generative AI tools to assist in drafting code for functional testing. All AI-assisted text was reviewed and revised by the authors, and all AI-assisted code was manually inspected and tested for correctness before use. The authors take full responsibility for the final content of the manuscript and the correctness of the code used in this work.

REPRODUCIBILITY STATEMENT

Our code is available at `https://anonymous.4open.science/r/smat-6a27/`. Section 4 describes SMAT and its efficient implementation. Section 5.1 and Appendix A provide the experimental settings, while Appendix E describes the implementation and measurement procedures. Proofs and their assumptions are provided in Appendices C and D.

REFERENCES

MohammadReza Davari and Eugene Belilovsky. Model breadcrumbs: Scaling multi-task model merging with sparse masks. In *ECCV*, 2024.

Pala Tej Deep, Rishabh Bhardwaj, and Soujanya Poria. DELLA-Merging: Reducing interference in model merging through magnitude-based sampling. *arXiv*, 2024.

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. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*, 2021.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-Aware Minimization for Efficiently Improving Generalization. In *ICLR*, 2021.

Antonio Andrea Gargiulo, Donato Crisostomi, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, and Emanuele Rodolà. Task singular vectors: Reducing task interference in model merging. In *CVPR*, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The Llama 3 Herd of Models. *arXiv*, 2024.

Chenyu Huang, Peng Ye, Tao Chen, Tong He, Xiangyu Yue, and Wanli Ouyang. EMR-Merging: Tuning-Free High-Performance Model Merging. In *NeurIPS*, 2024.

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

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging Weights Leads to Wider Optima and Better Generalization. In *UAI*, 2018.

Dong-Hwan Jang, Sangdoo Yun, and Dongyoon Han. Model Stock: All we need is just a few fine-tuned models. In *ECCV*, 2024.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. In *ICLR*, 2023.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015.

Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. ASAM: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In *ICML*, 2021.

Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. Mixout: Effective Regularization to Finetune Large-scale Pretrained Language Models. In *ICLR*, 2020.

Yeoreum Lee, Jinwook Jung, and Sungyong Baik. Mitigating parameter interference in model merging via sharpness-aware fine-tuning. In *ICLR*, 2025.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, 2004.

Shangge Liu, Yuehan Yin, Lei Wang, Qi Fan, Yinghuan Shi, Wenbin Li, Yang Gao, and Dacheng Tao. Understanding and enforcing weight disentanglement in task arithmetic. In *CVPR*, 2026.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2019.

Michael S Matena and Colin A Raffel. Merging models with Fisher-weighted averaging. In *NeurIPS*, 2022.

Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. Task Arithmetic in the Tangent Space: Improved Editing of Pre-Trained Models. In *NeurIPS*, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

Alexandre Ramé, Kartik Ahuja, Jianyu Zhang, Matthieu Cord, Léon Bottou, and David Lopez-Paz. Model Ratatouille: Recycling Diverse Models for Out-of-Distribution Generalization. In *ICML*, 2023.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 2014.

Anke Tang, Xianglin Luo, Li Shen, Yong Luo, Liang Ding, Han Hu, Bo Du, and Dacheng Tao. Concrete Subspace Learning based Interference Elimination for Multi-task Model Fusion. *arXiv*, 2023.

Anke Tang, Li Shen, Yong Luo, Enneng Yang, Han Hu, Lefei Zhang, Bo Du, and Dacheng Tao. FusionBench: A Unified Library and Comprehensive Benchmark for Deep Model Fusion. *JMLR*, 2025.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using DropConnect. In *ICML*, 2013.

Ke Wang, Nikolaos Dimitriadis, Guillermo Ortiz-Jimenez, François Fleuret, and Pascal Frossard. Localizing Task Information for Improved Model Merging and Compression. In *ICML*, 2024.

Xiao Wang, Yuansen Zhang, Tianze Chen, Songyang Gao, Senjie Jin, Xianjun Yang, Zhiheng Xi, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, and Xuanjing Huang. TRACE: A comprehensive benchmark for continual learning in large language models. *arXiv*, 2023.

Wei Wen, Yandan Wang, Feng Yan, Cong Xu, Chunpeng Wu, Yiran Chen, and Hai Li. SmoothOut: Smoothing out sharp minima to improve generalization in deep learning. *arXiv*, 2018.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *ICML*, 2022a.

Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo-Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig Schmidt. Robust fine-tuning of zero-shot models. In *CVPR*, 2022b.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. Optimizing statistical machine translation for text simplification. *TACL*, 2016.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A. Raffel, and Mohit Bansal. TIES-Merging: Resolving interference when merging models. In *NeurIPS*, 2023.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. AdaMerging: Adaptive model merging for multi-task learning. In *ICLR*, 2024.

Enneng Yang, Qun Yang, Peng Wang, Anke Tang, Guibing Guo, Xiaochun Cao, and Li Shen. MergOPT: A merge-aware optimizer for robust model merging. In *ICLR*, 2026.

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 *ICML*, 2024.

Table 5: Baseline training hyperparameters for the four backbones.

Backbone	ASAM radius ρ	MergOPT Laplace scale b_{MergOPT}	OrthoReg λ_{reg}
Llama-1B	2	5×10^{-4}	0.03
Llama-8B	0.25	2.5×10^{-4}	0.01
ViT-B/32	0.25	2×10^{-4}	0.1
ViT-L/14	0.125	5×10^{-5}	0.05

A Additional Experimental Settings

This appendix specifies the settings for the language and vision experiments in Tables 1 and 2.

Training. For both language backbones, we follow MergOPT (Yang et al., 2026) for the optimizer, learning rate, batch size, weight decay, and task-specific epoch counts. We use AdamW with batch size 8, learning rate 2×10^{-5}, a constant schedule, weight decay 10^{-3}, gradient clipping at 1, and a maximum length of 512.

The epoch counts for C-STANCE, FOMC, MeetingBank, ScienceQA, NumGLUE-cm, NumGLUE-ds, and 20Minuten are 5, 3, 7, 3, 5, 5, and 7, respectively. We append EOS before truncation and supervise it when it is retained.

For both ViT backbones, we use the public CLIP fine-tuning configuration from FusionBench (Tang et al., 2025). Vision models use Adam with learning rate 10^{-5}, cosine decay without warmup, zero weight decay, batch size 128, and 4,000 updates per task. The text encoder, projection, and classification heads remain frozen.

Table 5 lists the ASAM (Kwon et al., 2021) radius ρ, the MergOPT (Yang et al., 2026) Laplace scale b_{MergOPT}, and the OrthoReg (Liu et al., 2026) regularization weight λ_{reg}.

SMAT settings. For task i, SMAT samples one scalar Scale coefficient $\alpha_i \sim \mathcal{U}[\alpha_{\min}, 1]$ for all trainable coordinates. It uses a block-linear Mask with probability $p = 0.5$, excludes embeddings from masking, and simulates merging every four steps. Both language backbones use $(\alpha_{\min}, \sigma) = (0.2, 0.002)$, and both vision backbones use $(0.1, 0.001)$, where $\alpha_{\min}$ is the Scale lower bound and σ is the Perturb RMS.

Data splits. To construct training, dev, and final evaluation splits, we group examples by normalized prompt or RGB-image identity, remove groups present in the final evaluation split from training and dev, and remove dev groups from training. The three splits therefore have no overlapping examples under this exact-identity definition.

Language dev sizes are 200, 190, 200, 200, 50, 102, and 200 in the task order above. Vision uses 200 examples per task, including 200 of the 397 SUN397 classes. We evaluate language tasks on their released eval splits and vision tasks on their test splits; NumGLUE-cm has 41 evaluation examples.

Sparse-merger calibration. We calibrate sparsity on dev data using FT on Llama-1B and ViT-B/32. For TIES (Yadav et al., 2023), retention is selected from $\{0.1, 0.2, 0.3, 0.5\}$. For DARE (Yu et al., 2024) and DELLA (Deep et al., 2024), drop probability is selected from $\{0.1, 0.3, 0.5, 0.7, 0.9\}$; the DELLA window is fixed at 0.14.

Reference merging coefficients are 0.3 for language TIES/DARE, 0.2 for vision TIES/DARE, and 1.0 for DELLA; ties favor less pruning. Each selected sparse setting transfers across methods and model sizes within its family, after which each expert set selects its own merging coefficient.

TIES retention is 0.5 and DARE drop probability is 0.3 for both families. DELLA drop probability is 0.5 for language and 0.1 for vision.

Table 6: Muon results on Llama-3.2-1B-Instruct. Avg. is the mean over five merger methods.

Method	Expert	Merged					Avg.
		WA	TA	TIES	DARE	DELLA	
FT	**55.31**	37.06	40.46	38.53	40.92	34.26	38.25
ASAM	54.69	39.50	41.35	41.65	41.89	37.32	40.34
MergOPT	54.94	38.07	42.52	41.90	42.02	36.59	40.22
OrthoReg	50.89	42.60	44.73	43.98	45.42	**40.35**	43.41
SMAT	55.07	**42.77**	**46.27**	**46.44**	**45.72**	39.55	**44.15**

Decoding. For TRACE dev selection, each candidate gets one decoding run; the top two get two more runs on MeetingBank, ScienceQA, and 20Minuten, reusing deterministic-task scores. Selection uses the three-run mean, including for sparse calibration. Table differences are computed before rounding.

B Additional Analysis Settings

B.1 Expert-count Evaluation

Figure 5(a) merges the experts from Table 1 using Task Arithmetic (Ilharco et al., 2023) (TA). For each expert count $K = 2, \ldots, 6$, seven cyclic subsets of a fixed task ordering balance task inclusion; $K = 7$ uses all tasks. Methods share these subsets and three decoding repeats, and each merged model is evaluated only on its included tasks. We divide each task score by its independent FT expert score, average over tasks and subsets, and multiply by 100.

B.2 Muon Settings and Complete Results

Training and evaluation. All methods share a matrix learning rate of 4×10^{-4} and AdamW at 2×10^{-5} for auxiliary parameters. SMAT uses scalar Scale with $\alpha_{\min} = 0.2$, Mask probability $p = 0.5$, Perturb RMS 0.002, and interval $t = 4$; embeddings are excluded from masking.

We select merger configurations on dev data. All methods share seven-task data splits and budgets, with batch size 8, maximum length 512, and dynamic padding. Table 6 gives complete results; Appendix E details hardware and cost measurements.

C Theory of Simulated Merging

Fix expert i and the independent draws in Eq. (5), with coordinate masking probabilities $0 \le p_k < 1$ from Eq. (6). We use Euclidean vector norms $\|\cdot\|_2$ and operator matrix norms $\|\cdot\|_{\text{op}}$; expectations average the SMAT draws unless specified.

Proposition 2 (Moments of the simulated state) *Let* $\bar{\alpha} := \mathbb{E}[\alpha_i] = (1 + \alpha_{\min})/2$, $v_\alpha := \mathrm{Var}(\alpha_i) = (1 - \alpha_{\min})^2/12$, *and* $s_\alpha := \mathbb{E}[\alpha_i^2] = \bar{\alpha}^2 + v_\alpha$. *The mean* $\bar{\theta}_i$ *and covariance* C_i *of the simulated state are*

$$\begin{aligned} \bar{\theta}_i &:= \mathbb{E}[\widetilde{\theta}_i] = \theta_0 + \bar{\alpha}\Delta_i, \\ C_i &:= \mathrm{Cov}(\widetilde{\theta}_i) = v_\alpha \Delta_i \Delta_i^\top + s_\alpha \,\mathrm{Diag}\left(\Delta_{i,k}^2 \frac{p_k}{1 - p_k}\right) + \sigma^2 I_d. \end{aligned} \tag{11}$$

Here Diag *forms a diagonal matrix,* I_d *is the* $d \times d$ *identity matrix,* $^\top$ *denotes transpose, and* Var *and* Cov *denote variance and covariance.*

Proof. For each coordinate, $\mathbb{E}[m_{i,k}] = 1$ and $\mathbb{E}[m_{i,k}^2] = 1/(1 - p_k)$. The scalar α_i is shared across coordinates, producing the first covariance term. Independent masking produces the diagonal term, and centered independent Perturb produces $\sigma^2 I_d$. □

Local loss expansion. Let $Z := \widetilde{\theta}_i - \bar{\theta}_i$ be the centered displacement, and write $H_i(\theta) := \nabla^2 \mathcal{L}_i(\theta)$ for the loss Hessian. Assume $\mathcal{L}_i$ is twice differentiable and H_i is L_H-Lipschitz on the convex hull

of the support of $\widetilde{\theta}_i$, meaning $\|H_i(\theta) - H_i(\theta')\|_{\text{op}} \le L_H\|\theta - \theta'\|_2$. Then

$$\mathbb{E}\mathcal{L}_i(\widetilde{\theta}_i) = \mathcal{L}_i(\bar{\theta}_i) + \frac{1}{2}\,\text{tr}\big[H_i(\bar{\theta}_i)C_i\big] + r_i, \qquad |r_i| \le \frac{L_H}{6}\mathbb{E}\|Z\|_2^3. \tag{12}$$

Here tr is the matrix trace and r_i the third-order Taylor remainder. Since $\mathbb{E}Z = 0$, the linear term vanishes. For positive semidefinite $H_i(\bar{\theta}_i) \succeq 0$, the trace term penalizes curvature.

Perturb smooths the conditional loss. Condition on Scale and Mask, and write $\theta_\star := \widetilde{\theta}_i - \epsilon_i$. For $\sigma > 0$, let q_σ be the symmetric Perturb density. The conditional smoothed loss is

$$\overline{\mathcal{L}}_{i,\sigma}(\theta_\star) := \mathbb{E}_{\epsilon\sim q_\sigma}[\mathcal{L}_i(\theta_\star + \epsilon)] = (\mathcal{L}_i * q_\sigma)(\theta_\star), \tag{13}$$

where $*$ denotes convolution. Write $\epsilon = \sigma z$, where z is standardized noise independent of σ, with $\mathbb{E}_z[z] = 0$, $\mathbb{E}_z[zz^\top] = I_d$, and $\mathbb{E}_z\|z\|_2^3 < \infty$. If H_i is L_H-Lipschitz on every sampled segment from $\theta_\star$ to $\theta_\star + \sigma z$, then, for sufficiently small σ,

$$\overline{\mathcal{L}}_{i,\sigma}(\theta_\star) = \mathcal{L}_i(\theta_\star) + \frac{\sigma^2}{2}\,\text{tr}\,H_i(\theta_\star) + r_\sigma, \qquad |r_\sigma| \le \frac{L_H\sigma^3}{6}\mathbb{E}_z\|z\|_2^3 = o(\sigma^2). \tag{14}$$

Here r_σ is the Taylor remainder; $o(\sigma^2)/\sigma^2 \to 0$ as $\sigma \to 0$. Independent Uniform, Gaussian, and Laplace noise with equal coordinate variance thus share the leading term, but this expansion does not rank them at finite noise scales.

Proposition 3 (Loss difference from a target merge distribution) *Let P_i be a target distribution of merged states (Eq. (1)) and Q_i the distribution of $\widetilde{\theta}_i$. Relative to reference $\theta_\star$, define the first and second displacement moments and third absolute moment of a random parameter $\Theta \sim D$:*

$$\begin{aligned} a_D &:= \mathbb{E}_{\Theta\sim D}[\Theta - \theta_\star], \\ S_D &:= \mathbb{E}_{\Theta\sim D}[(\Theta - \theta_\star)(\Theta - \theta_\star)^\top], \\ \tau_D &:= \mathbb{E}_{\Theta\sim D}\|\Theta - \theta_\star\|_2^3, \qquad D \in \{P_i, Q_i\}. \end{aligned} \tag{15}$$

Assume τ_{P_i} and τ_{Q_i} are finite and that H_i is L_H-Lipschitz on the convex hull of $\{\theta_\star\} \cup \text{supp}(P_i) \cup \text{supp}(Q_i)$, where supp *denotes support. Then*

$$\begin{aligned} &\mathbb{E}_{\Theta\sim P_i}\mathcal{L}_i(\Theta) - \mathbb{E}_{\Theta\sim Q_i}\mathcal{L}_i(\Theta) \\ &\quad = \nabla\mathcal{L}_i(\theta_\star)^\top(a_{P_i} - a_{Q_i}) + \frac{1}{2}\,\text{tr}[H_i(\theta_\star)(S_{P_i} - S_{Q_i})] + r_{P,Q}, \\ &|r_{P,Q}| \le \frac{L_H}{6}(\tau_{P_i} + \tau_{Q_i}). \end{aligned} \tag{16}$$

Proof of Proposition 3. Apply Taylor's theorem at $\theta_\star$ under P_i and Q_i, then subtract. Here $r_{P,Q}$ is the combined third-order Taylor remainder. The first term retains directional or nonzero-mean updates from other experts, so the result does not assume matching moments. □

This expected-loss comparison does not guarantee higher task scores.

D Training and Scheduling Details

Proof of Proposition 1. Let x_j and y_j be the separate-loss and mixed-loss gradient-descent iterates, respectively, with $x_0 = y_0 = w$. Let $s_j \in \{A, B\}$ specify the loss at separate-loss step j, with exactly k occurrences of B among $j = 0, \dots, n-1$. The updates are

$$x_{j+1} = x_j - \eta\nabla\mathcal{L}_{s_j}(x_j), \qquad y_{j+1} = y_j - \eta\nabla J(y_j).$$

Because J is a convex combination of the two losses, its gradient is also L_g-Lipschitz and bounded by G; the losses themselves need not be convex. Thus $\|x_j - w\|_2 \le j\eta G$ and $\|y_j - w\|_2 \le j\eta G$. The component counts give $\sum_{j=0}^{n-1}\nabla\mathcal{L}_{s_j}(w) = n\nabla J(w)$, so

$$\begin{aligned} \|x_n - (w - n\eta\nabla J(w))\|_2 &\le \eta\sum_{j=0}^{n-1}\|\nabla\mathcal{L}_{s_j}(x_j) - \nabla\mathcal{L}_{s_j}(w)\|_2 \\ &\le \eta L_g\sum_{j=0}^{n-1} j\eta G = \frac{\eta^2 L_g G\, n(n-1)}{2}. \end{aligned}$$

The same bound holds for y_n. The triangle inequality gives Eq. (9). □

This is a local comparison over one cycle: for fixed n, the endpoint difference is second order in η. It does not require convex losses or imply that the two full training trajectories coincide. The result is for ordinary gradient descent; AdamW's adaptive moments and Muon's momentum and matrix transformation require a separate analysis.

Cycle-averaged gradients in SMAT. The $t-1$ ordinary updates and one simulated-loss update per period match the objective weight $\lambda = 1/t$. The following bound applies to the population gradients along any parameter path with bounded step lengths, without assuming a particular optimizer.

Proposition 4 (Drift within a training period) *Let $\zeta_i := (\alpha_i, \mathbf{m}_i, \epsilon_i)$ denote an SMAT draw, and write $\widetilde{\theta}_i(\theta, \zeta_i)$ for the state in Eq.* (5) *at current expert parameters θ. Define the clean and SMAT population gradient fields as*

$$g_{\mathrm{c}}(\theta) := \nabla \mathcal{L}_i(\theta), \qquad g_{\mathrm{s}}(\theta) := \nabla_\theta \mathbb{E}_{\zeta_i}\big[\mathcal{L}_i(\widetilde{\theta}_i(\theta, \zeta_i))\big]. \tag{17}$$

Let $\vartheta_0, \dots, \vartheta_{t-1}$ be the parameter values in one period, with $\|\vartheta_{j+1} - \vartheta_j\|_2 \le \delta$ for $j = 0, \dots, t-2$. Let $c_j = \mathrm{c}$ for $0 \le j < t-1$ and $c_{t-1} = \mathrm{s}$. Suppose both fields are L_g-Lipschitz along this path: for $h \in \{g_{\mathrm{c}}, g_{\mathrm{s}}\}$, $\|h(\theta) - h(\theta')\|_2 \le L_g \|\theta - \theta'\|_2$. Then

$$\left\| \frac{1}{t} \sum_{j=0}^{t-1} g_{c_j}(\vartheta_j) - \big((1 - \tfrac{1}{t}) g_{\mathrm{c}}(\vartheta_0) + \tfrac{1}{t} g_{\mathrm{s}}(\vartheta_0)\big) \right\|_2 \le \frac{L_g \delta (t-1)}{2}. \tag{18}$$

Proof. At ϑ_0, the parenthesized expression equals $\nabla \mathcal{J}_i^{\mathrm{SMAT}}(\vartheta_0)$ for $\lambda = 1/t$. The triangle inequality and Lipschitz condition bound the difference by $t^{-1} L_g \sum_{j=0}^{t-1} \|\vartheta_j - \vartheta_0\|_2$, which is at most $t^{-1} L_g \delta \sum_{j=0}^{t-1} j$. □

At fixed parameters, unbiased minibatch and simulation gradients also give the mixed-objective gradient in expectation when averaged over one cycle. At changing parameters, the bound controls the population-gradient drift; sampled gradients introduce additional variability. It does not imply identical updates for stochastic or stateful optimizers.

Implementation details. The fused implementation regenerates the forward-pass mask for gradients and restores original storage before the optimizer update, including any weight decay.

Table 7: Training time for complete expert sets. Mean is seconds per optimizer step; total is summed wall time in seconds.

	Llama-1B		Llama-8B		ViT-B/32		ViT-L/14	
Method	Mean	Total	Mean	Total	Mean	Total	Mean	Total
FT	0.0687	1,382.1	0.3543	7,130.1	0.1255	4,015.3	1.4494	46,381.4
ASAM	0.1971	3,965.7	0.9481	19,079.0	0.2171	6,946.1	2.8991	92,771.3
MergOPT	0.1180	2,375.2	0.5469	11,005.0	0.1438	4,601.5	1.4862	47,559.9
OrthoReg	0.3910	7,868.5	4.1047	82,599.3	0.1411	4,515.7	1.5006	48,019.5
SMAT	0.0700	1,409.0	0.3594	7,231.2	0.1268	4,057.5	1.4516	46,451.0

E Implementation and Measurement Details

E.1 Hardware, Software, and Implementation

All training runs use NVIDIA H800 GPUs. Llama-8B uses blockwise model parallelism across two GPUs per expert while keeping the same effective batch size and update budget; the other backbones use one GPU. FT–SMAT comparisons match GPU counts and software.

All training runs use Python 3.10.12, PyTorch 2.11.0+cu130, and CUDA 13.0.

SMAT uses the fused implementation from Section 4.3. Original and simulated weights occupy separate storage; a reusable buffer holds simulated weights, and frozen base weights remain in pinned CPU memory. Fused kernels construct simulated weights and rescale gradients. The model uses simulated storage for the forward and backward pass, then restores original weights for the optimizer update.

E.2 Training-Time Measurement

Table 7 reports absolute times for the experiments in Tables 1 and 2. We measure one full training run per configuration. Each total is the sum of training-loop wall-clock seconds across a complete expert set—seven language experts or eight vision experts—rather than parallel elapsed time.

Mean time per step divides each total by all optimizer steps: 20,123 for a language expert set and 32,000 for a vision expert set. Timing includes batch loading, initial warmup, and logging. It excludes initialization, checkpoint saving, hyperparameter search, merging, and evaluation.

E.3 Memory Measurement

For Tables 1 and 2, *Mem.* is the largest per-GPU PyTorch allocated peak across experts, in GiB. Its adjacent percentage is the pre-rounding change from FT within the same model block. These peaks cover training and final checkpoint serialization.

For Muon, memory is the largest per-device PyTorch allocated peak across the seven experts.