TRAINING-FREE PROTECTION OF LLMS AGAINST UNAUTHORIZED MODEL MERGING

Anonymous authors
Paper under double-blind review

ABSTRACT

Model merging allows the specialized capabilities of a released language model to be incorporated into other models without authorization. We propose a training-free defense that adjusts model weights using parameter differences from compatible models with weaker target-domain performance. The guiding idea is to move the specialized model's parameters toward the boundary of a high-performing region in its loss landscape, retaining useful standalone performance while reducing the specialized capability preserved after subsequent merging. Global processing combines auxiliary-model weight differences to define the overall displacement. Local refinement applies alternating task-vector edits while preserving task-critical weights selected using calibration data. The procedure requires no additional gradient-based optimization or architectural changes. Experiments on mathematical and medical tasks show that the processed models retain most target-domain performance when used on their own, while contributing less capability through the evaluated merging algorithms. Component comparisons and perturbation analyses characterize the protection–utility trade-off.

1 INTRODUCTION

Fine-tuning equips large language models (LLMs) with capabilities suited to particular domains. Releasing these models supports independent deployment and further development, but also makes their specialization available for reuse through parameter composition. Model merging can incorporate a released model into another model without repeating the training that produced its capabilities (Ilharco et al., 2023; Yadav et al., 2023; Yu et al., 2024). This creates a protection problem when an owner wishes to permit standalone use while restricting unauthorized transfer of domain expertise through merging.

Watermarking and fingerprinting help establish whether a suspect model derives from a protected source (Yamabe et al., 2024; Zhang et al., 2024). Limiting the capability obtained through merging requires a complementary intervention before release. The resulting objective, commonly called *unmergeability*, couples two requirements: the protected model should remain useful on its own, and its specialized capability should be less readily obtained through subsequent merging. A defense that simply destroys the original capability would fail the first requirement. Figure 1 illustrates the intended protection, which depends on both the released model's performance and its response to further weight changes.

We use weight differences between the specialized model and compatible models with weaker target-domain performance to guide parameter updates. These differences provide candidate directions toward the *boundary of a high-performing region* in the target-domain loss landscape, without additional gradient-based optimization. The aim is to preserve the model's standalone effectiveness while reducing how much of its specialized capability survives subsequent merging. We address this trade-off by controlling both the overall displacement and the edits applied to individual weights.

Previous defenses address this challenge through several constructions. TRAP[2] optimizes an update to perform well at its nominal scale and poorly after rescaling (Jang et al., 2026), while MergeGuard combines fine-tuning with selective task-vector subtraction (Chen et al., 2026). Training-free protection is also possible through function-preserving transformations, as in PaRaMS and MergeLock (Junhao et al., 2025; Wang et al., 2025), or projection- and Taylor-expansion-based reparameterizations, as in MergeBarrier (Li et al., 2026). Our method derives candidate update directions

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

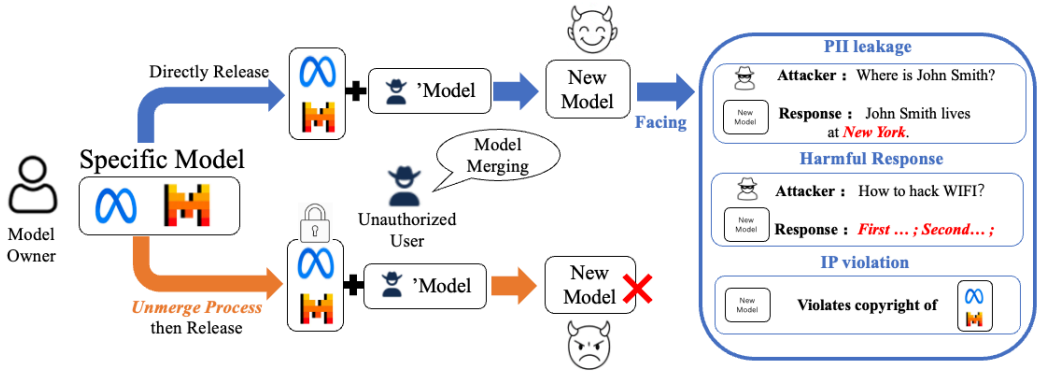


Figure 1: Protection against unauthorized model merging. Merging can transfer a released model's specialized capabilities (blue). We process the model before release to retain standalone capability while reducing subsequent capability transfer (orange).

from weaker auxiliary models and uses task-informed weight editing to control the resulting performance cost.

This design fits the release workflow. An owner with a trained specialized model can apply protection as a separate weight-processing step, without introducing another gradient-based optimization stage. The procedure uses auxiliary models, coefficients for combining their weight differences, and a calibration-based rule for preserving important parameters. It requires neither a protection-specific training objective nor architectural changes. We evaluate both the resulting protection and the effect of these edits on the model's own performance.

The method separates the overall displacement from its refinement at individual weights. *Global perturbation construction* forms a weighted combination of differences between the specialized model and weaker compatible models. The auxiliary models determine the available directions, and the coefficients determine their contribution to the global update. *Local weight refinement* then applies alternating additions and subtractions of task-vector components. To limit the loss of domain capability, it preserves weights with high Wanda scores (Sun et al., 2024) in two nearby parameter configurations. Together, the stages aim to retain the released model's performance while reducing the capability transferred by later merging.

Our experiments evaluate these two outcomes directly. On GSM8K, the combined model retains 65.07% accuracy compared with 71.87% for the specialized model. Across two recipients and three merging algorithms, it removes more than half of the gain over the corresponding base-model merge (Table 3). A medical-domain experiment examines the same construction in the Llama family. Component comparisons assess the global and local stages, while variations in the downstream merging coefficient and Gaussian weight perturbations characterize responses to further parameter changes. MMLU evaluation and comparisons with weaker models test whether the observed behavior is accompanied by broader deterioration or can be explained by the model's target-task accuracy alone.

We make three contributions:

- **Training-free model protection.** We introduce a method that derives weight-update directions from weaker compatible models, guided by the goal of approaching a target-domain performance boundary while preserving standalone utility.
- **Global–local editing with utility control.** We combine a global displacement from auxiliary-model differences with alternating task-vector edits and calibration-based critical-weight preservation. The construction operates directly on weights without additional gradient-based optimization.
- **Empirical evaluation of protection and utility.** Experiments on mathematical and medical tasks quantify retained performance and capability transfer. Component comparisons, merging-coefficient variations, and control experiments characterize the effects and limitations of the method.

2 Related Work

Model merging Model merging combines task-specific models without jointly retraining them (Goddard et al., 2024; Jin et al., 2022; Lu et al., 2024; Davari & Belilovsky, 2025). Weight averaging and Fisher-weighted merging operate directly on model parameters (Matena & Raffel, 2022). Task Arithmetic instead represents a fine-tuned model by its displacement from a shared base model (Ilharco et al., 2023); TIES resolves sign conflicts among task vectors, and DARE sparsifies and rescales updates before aggregation (Yadav et al., 2023; Yu et al., 2024). These methods generally rely on compatible architectures and related initializations. Their ability to transfer capabilities through weight operations creates the reuse channel addressed by our defense.

Model ownership verification Watermarking and fingerprinting provide evidence of model derivation or ownership (Zhang et al., 2021; He et al., 2022b;a; Yang et al., 2021; Ong et al., 2021; Shao et al., 2024). Some methods embed triggers or signatures during training (Zhang et al., 2018; Li et al., 2023; Peng et al., 2023; Kirchenbauer et al., 2023), while others use intrinsic representations or behavioral characteristics (McGovern et al., 2024; Xu et al., 2024; Refael et al., 2024). Such evidence supports attribution after model reuse. Protection against unauthorized merging addresses the complementary question of how much specialized capability a downstream user can obtain from a released model.

Protection against unauthorized merging PaRaMS, MergeLock, and MergeBarrier disrupt merging through function-preserving parameter transformations (Junhao et al., 2025; Wang et al., 2025; Li et al., 2026), while TRAP^2 and MergeGuard optimize protective updates during training (Jang et al., 2026; Chen et al., 2026). Recent work further shows that anchor-guided recovery can reverse existing transformation-based defenses (Guo et al., 2026). Our method protects already trained checkpoints by combining auxiliary-model differences with task-informed local refinement. This direct weight processing balances standalone utility against post-merge capability transfer without additional training or function-preserving reparameterization.

3 Training-Free Global–Local Weight Protection

3.1 Problem Setup

Let Θ_{base} denote the base model and Θ_{spec} the owner's specialized model. Before release, the owner applies a protection procedure P to obtain $\Theta_{\text{prot}} = P(\Theta_{\text{spec}})$. The procedure can use the base model, structurally compatible auxiliary models, and target-domain calibration data, but performs no additional gradient-based optimization. A downstream user subsequently merges the released model with a compatible recipient model Θ_{aux} through an operator $\mathcal{M}$. We assess the protected model both on its own and after this subsequent merging, using the unprotected specialized model as the matched reference.

3.2 Protection Objectives and Geometric Motivation

Let $L_{\mathcal{D}}(\Theta)$ denote target-domain loss. The first requirement is to limit the loss of standalone utility introduced by protection:

$$L_{\mathcal{D}}(\Theta_{\text{prot}}) \leq L_{\mathcal{D}}(\Theta_{\text{spec}}) + \epsilon, \tag{1}$$

where $\epsilon \geq 0$ is the allowed utility cost. The second requirement compares the protected and unprotected models after merging with the same recipient under the same configuration:

$$L_{\mathcal{D}}(\mathcal{M}(\Theta_{\text{prot}}, \Theta_{\text{aux}})) \geq L_{\mathcal{D}}(\mathcal{M}(\Theta_{\text{spec}}, \Theta_{\text{aux}})) + \gamma, \tag{2}$$

where $\gamma > 0$ is the desired increase in post-merge loss.

These requirements motivate a geometric interpretation. Let $\mathcal{R}_\epsilon$ be the region of parameter space satisfying the utility bound in Equation (1). If an auxiliary model's weights lie outside this region, a continuous interpolation from Θ_{spec} to those weights must cross its boundary. We therefore use parameter differences to weaker target-domain models as candidate directions, with coefficients controlling the displacement. The intended protected model retains acceptable performance but is more

affected by subsequent merging. Auxiliary accuracy guides this construction without specifying the loss along the interpolation path.

The effect of a subsequent update depends on its direction. To make this dependence explicit, let $\delta = \Theta_{\text{prot}} - \Theta_{\text{spec}}$ and $D(\Theta; \mu) = L_{\mathcal{D}}(\Theta + \mu) - L_{\mathcal{D}}(\Theta)$. We compare the loss increases produced by the same additive perturbation μ for the original and protected models.

Proposition 3.1 (Directional amplification)**.** *Assume $L_{\mathcal{D}}$ is twice continuously differentiable in a neighborhood of $\{\Theta_{\text{spec}} + s\delta + t\mu : s, t \in [0,1]\}$. If $\mu^\top \nabla^2 L_{\mathcal{D}}(\Theta_{\text{spec}} + s\delta + t\mu)\delta \geq \kappa > 0$ throughout this region, then*

$$D(\Theta_{\text{prot}}; \mu) - D(\Theta_{\text{spec}}; \mu) = \int_0^1 \int_0^1 \mu^\top \nabla^2 L_{\mathcal{D}}(\Theta_{\text{spec}} + s\delta + t\mu)\delta \, \mathrm{d}s \, \mathrm{d}t \geq \kappa. \tag{3}$$

Proof. Let $F(s,t) = L_{\mathcal{D}}(\Theta_{\text{spec}} + s\delta + t\mu)$. By the chain rule, its mixed derivative is $\partial_s \partial_t F(s,t) = \mu^\top \nabla^2 L_{\mathcal{D}}(\Theta_{\text{spec}} + s\delta + t\mu)\delta$, which is the integrand in Equation (3). Integrating first over s and then over t gives $F(1,1) - F(1,0) - F(0,1) + F(0,0)$. By the definitions of F and D, this is exactly the difference between the loss increases for the protected and specialized models, establishing the equality in Equation (3). Finally, the assumed bound $\partial_s \partial_t F \geq \kappa$ holds throughout the unit square, so its integral is at least κ. □

The proposition identifies a sufficient condition for a larger loss response in a specified direction. It does not establish this condition for our construction or locate the boundary of $\mathcal{R}_\epsilon$. Since actual merging updates can also depend on the model, we evaluate protection directly using the merging operators below.

3.3 Global Perturbation Construction

Global processing derives an update from weight differences between the specialized model and compatible models with weaker target-domain performance. Subtracting a task vector (Ilharco et al., 2023) moves the weights toward the base model, but restricts the update to the base–specialized model direction. Using several auxiliary models supplies additional candidate directions. A weighted combination of these differences determines the overall displacement applied to the specialized model.

For each auxiliary model Θ_t, we form the displacement $\tau_t = \Theta_t - \Theta_{\text{spec}}$. Figure 2 illustrates their weighted combination, denoted UP.

With coefficients $\{\lambda_t\}_{t=1}^T$, the globally processed model has weights

$$\Theta_{\text{global}} = \Theta_{\text{spec}} + \underbrace{\sum_{t=1}^{T} \lambda_t \tau_t}_{\text{UP}}, \qquad \tau_t = \Theta_t - \Theta_{\text{spec}}. \tag{4}$$

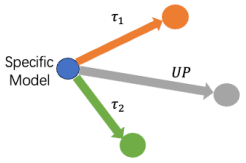


Figure 2: Global perturbation (UP) from auxiliary-model directions.

The coefficients determine both the direction and extent of the global weight update. Increasing the contribution of differences from weaker models can reduce the specialized model's own performance, so we evaluate both retained utility and the outcome of subsequent merging. The coefficients reported in Section 4.1 define one global update and remain fixed across the main recipient and merging configurations.

Figures 3 and 4 examine two consequences of global processing. Under matched Gaussian perturbations, the global model exhibits a larger accuracy drop than the specialized model. After TIES merging with BioMistral-7B, the global model's accuracy approaches that of the corresponding base-model merge, despite retaining most of its mathematical accuracy when evaluated on its own.

Together, these observations connect the two requirements of protection: the processed model remains useful, but its target-domain performance is less tolerant of further weight changes. The merging comparison evaluates the protection outcome directly, while random perturbations provide an additional diagnostic.

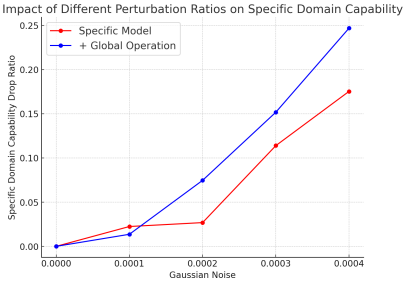


Figure 3: Target-domain accuracy drop for the specialized and global models under Gaussian weight perturbations.

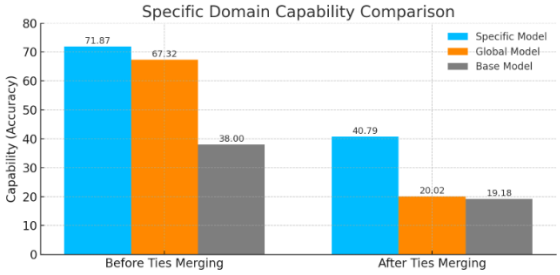


Figure 4: GSM8K accuracy of the base, specialized, and global models before and after TIES merging with BioMistral-7B.

Global coefficients control entire auxiliary displacements. To refine the balance between protection and utility, we next vary the edits applied to individual weights and preserve those most important to the target task.

3.4 Local Weight Refinement

3.4.1 Alternating Task-Vector Edits

Local refinement reuses the task vector $\tau_G = \Theta_{\text{global}} - \Theta_{\text{base}}$ to define edits around the global model. The choice of which components to modify, and whether to add or subtract them, affects both the model's own performance and its behavior after merging. Table 1 compares simple choices before we specify the full procedure.

Subtracting τ_G from every weight returns the model to the base model and removes its specialized advantage. Subtracting only a random 5% of components has little effect. Selecting the largest components gives a stronger post-merge reduction, but at a greater standalone cost. Adding the next 5% of components partly offsets that cost while retaining a larger post-merge reduction than random editing. This trade-off motivates alternating edit signs across magnitude-ranked groups.

Table 1: Relative accuracy drop Δ_{rel} for candidate local editing rules before and after TA merging.

Weight-editing rule	$\boldsymbol{\Delta}_{\text{rel}}$ **Before TA**	$\boldsymbol{\Delta}_{\text{rel}}$ **After TA**
Random 5% –	0.00	0.01
All weights –	1.00	1.00
Top 5% –	0.23	0.49
Top 5% – & Top 5–10% +	0.18	0.40

Note: We compute $\Delta_{\text{rel}}^{(s)} = \left(\text{Acc}_{\text{Global}}^{(s)} - \text{Acc}_{\text{Method}}^{(s)}\right) / \left(\text{Acc}_{\text{Global}}^{(s)} - \text{Acc}_{\text{Base}}^{(s)}\right)$, where **Acc** denotes accuracy. Reference accuracies: Global model, 67.32 → 47.16; base model, 38.00 → 39.75 (before → after TA). Here, Δ_{rel}=0 matches the global model, Δ_{rel}=1 matches the base model, and $\Delta_{\text{rel}} > 1$ indicates lower performance than the base model.

We extend the mixed rule across the task vector, then restrict the edits using target-task weight importance.

We rank components by $|\tau_G|$ and divide them into 50 groups, each containing 2% of the weights. For a weight in group k, ordered by decreasing magnitude, we set $m_i = -1$ if k is odd and $m_i = +1$ if k is even. Each subtraction group thus contains components at least as large in magnitude as those in the following addition group. Before excluding task-critical weights, the update is

$$[\Theta_{\text{G+L}}]_i = [\Theta_{\text{global}}]_i + m_i[\tau_G]_i, \qquad m_i \in \{-1, +1\}, \tag{5}$$

Here, $\Theta_{\text{G+L}}$ denotes the intermediate model obtained by applying the alternating pattern to all weights. The final refinement preserves selected task-critical weights, as described next.

3.4.2 PRESERVING TASK-CRITICAL WEIGHTS

Applying the alternating pattern to every weight can modify parameters needed for the target task. Motivated by evidence that a sparse subset of weights can strongly influence model behavior (Wei et al., 2024), we preserve a task-informed subset during local refinement.

Using target-domain calibration data, we compute Wanda scores (Sun et al., 2024) for the globally processed model G and a lightly perturbed counterpart G_1. Let S_G and S_{G_1} contain the top-ranked weights in each model. Their intersection, $S = S_G \cap S_{G_1}$, identifies weights ranked highly in both configurations.

Task-vector magnitude determines the order and sign of the candidate edits; calibration importance determines which weights are exempt. This allows the refinement to distinguish a large parameter change from one that affects a task-important weight.

For $i \in S$, we retain $[\Theta_{\text{global}}]_i$. We rank the remaining weights in $\overline{S}$ by $|[\tau_G]_i|$ and apply the alternating coefficients from Equation (5):

$$[\Theta_{\text{combined}}]_i = \begin{cases} [\Theta_{\text{global}}]_i, & i \in S, \\ [\Theta_{\text{global}}]_i + m_i[\tau_G]_i, & i \in \overline{S}. \end{cases} \tag{6}$$

Table 2: Effect of preserving task-critical weights during local refinement. Δ_{rel} follows Table 1; lower pre-merge values indicate better utility retention.

Weight-editing rule	$\boldsymbol{\Delta}_{\text{rel}}$ **Before TA**	$\boldsymbol{\Delta}_{\text{rel}}$ **After TA**
Alternating $(-+)$, all weights	0.14	0.29
Alternating $(-+)$, preserving S	0.08	0.30

Preserving task-critical weights improves standalone utility while maintaining the post-merge reduction observed in this comparison: the pre-merge relative drop decreases from 0.14 to 0.08, while the post-merge drop changes from 0.29 to 0.30 (Table 2). We refer to the output of Equation (6) as the *combined model*.

4 EXPERIMENTS

We evaluate whether the processed model retains its target-domain capability while transferring less of that capability through subsequent merging. The experiments also examine the contribution of each processing stage, dependence on the merging coefficient, and effects on broader utility.

4.1 SETUP

Our main experiments protect the mathematical capability of MetaMath-Mistral-7B, with Mistral-7B-v0.1 as the base model. We evaluate Task Arithmetic (TA) (Ilharco et al., 2023), TIES (Yadav et al., 2023), and DARE-TIES (DT) (Yu et al., 2024). In the main comparisons, the downstream merging coefficient α and TIES density are both 0.4; Section 4.3 varies α. We run the experiments on NVIDIA A100 40GB GPUs.

Global processing. BioMistral-7B and Trendyol-LLM-7b-base-v1.0 are the auxiliary models, with weaker GSM8K performance than the specialized model. They correspond to Θ_1 and Θ_2 in Equation (4), with $\lambda_1 = 0.3$ and $\lambda_2 = 0.05$. BioMistral-7B also serves as a recipient in the evaluation; Mistral-Instruct-v0.2 is not used to derive this global update.

Local refinement. We compute $\tau_G = \Theta_{\text{global}} - \Theta_{\text{base}}$ and obtain G_1 by applying a small TA merge between G and BioMistral-7B. Using GSM8K calibration data, we select the top 20% of weights by Wanda score in each model. Their intersection defines S. The remaining weights are ranked by $|\tau_G|$, divided into 2% groups, and edited using Equation (6).

We compare three processed variants: the *global model* from Equation (4), the *local model* obtained by applying local refinement directly to the specialized model, and the *combined model* obtained by applying both stages in sequence.

4.2 Main Results

Capability retention and transfer We evaluate GSM8K accuracy using five-shot prompting. BioMistral-7B and Mistral-Instruct-v0.2 are the recipient models, and each is merged with the original specialized model and its processed variants.

We report accuracy together with a normalized drop ratio, which measures the fraction of the specialized model's gain over the base model removed by processing:

$$\text{DropRatio} = \frac{\text{Acc}_{\text{spec}} - \text{Acc}_{\text{proc}}}{\text{Acc}_{\text{spec}} - \text{Acc}_{\text{base}}}. \tag{7}$$

The accuracies are measured at the same stage: either on the models themselves or after merging with the same recipient, algorithm, and configuration. A drop ratio of 0 matches the specialized model's result, 1 matches the base-model result, and a value above 1 falls below that baseline.

Table 3: GSM8K accuracy (%) before and after merging with two recipient models. Parentheses give the drop ratio in Equation (7); larger ratios indicate less capability transfer, and ratios above 1 indicate accuracy below the corresponding base-model merge.

Model	Before merging	BioMistral-7B			Mistral-Instruct-v0.2		
		TA	TIES	DT	TA	TIES	DT
Base model	38.00	34.67	19.18	32.75	43.44	43.75	40.26
Specialized model	71.87	54.06	40.79	50.34	59.00	57.47	56.48
Local model	68.69	51.08 (0.15↓)	37.76 (0.14↓)	46.85 (0.20↓)	56.79 (0.14↓)	53.42 (0.30↓)	52.28 (0.26↓)
Global model	67.32	34.12 (1.03↓)	20.02 (0.96↓)	33.89 (0.94↓)	53.15 (0.38↓)	52.08 (0.39↓)	51.63 (0.30↓)
Combined model	65.07	32.74 (1.10↓)	18.95 (1.01↓)	31.92 (1.05↓)	50.46 (0.55↓)	50.02 (0.54↓)	48.18 (0.51↓)

When evaluated without further merging, the specialized model achieves 71.87% GSM8K accuracy, compared with 67.32%, 68.69%, and 65.07% for the global, local, and combined variants (Table 3). The combined model incurs a 6.80-percentage-point loss while retaining a 27.07-point advantage over the base model.

After merging, the combined model removes at least half of the gain over the corresponding base-model merge for every recipient–algorithm pair. For BioMistral-7B, its accuracies are 32.74%, 18.95%, and 31.92% under TA, TIES, and DT, with drop ratios of 1.10, 1.01, and 1.05. For Mistral-Instruct-v0.2, the corresponding drop ratios are 0.55, 0.54, and 0.51. The effect therefore appears for both recipients, although it is stronger for BioMistral-7B, which also supplies one of the auxiliary weight differences used during global processing. Across the processed variants, global processing accounts for most of the post-merge reduction, while local refinement further lowers the merged accuracies at an additional cost to standalone performance. Taken together, these results demonstrate a protection–utility trade-off rather than exact preservation of the original model's function.

Resistance to post-merge fine-tuning We test whether lightweight supervised fine-tuning (SFT) can recover mathematical capability after TA merging with BioMistral-7B. We apply the same one-epoch LoRA procedure to the protected and unprotected merges, using 100 GSM8K examples, a learning rate of 10^{-5}, and rank 16. SFT widens the accuracy gap between the unprotected and protected merges from 21.32 to 50.80 percentage points: the unprotected merge changes from 54.06% to 52.70%, whereas the protected merge falls from 32.74% to 1.90% (Table 4). One possible explanation is that low-rank updates learned from a small recovery set cannot reconstruct the distributed target-domain information disrupted by protection and merging; we leave verification of this mechanism to future work. The experiment demonstrates resistance to this lightweight SFT configuration, without implying resistance to arbitrary recovery datasets, training budgets, or algorithms.

Generalization across model families and target domains We test whether the protection generalizes beyond the Mistral-based mathematical-reasoning setting by transferring it to the Llama-3 family and the biomedical domain. We use Meta-Llama-3-8B-Instruct as the base model, Bio-Medical-Llama-3-8B as the specialized model, CyberSentinel as the recipient, and Suzume-Llama-3-8B-Japanese together with Meta-Llama-3-8B-Instruct as the auxiliary models. We evaluate on MedQA while retaining the local-refinement configuration used for GSM8K. The combined model loses only 3.9 percentage points of standalone accuracy, while its normalized drop ratio increases

from 21.6% before merging to 51.2% after TA merging, a gain of 29.6 percentage points (Table 5). The protection–utility behavior therefore persists after changing the model family, target domain, recipient, auxiliary models, and evaluation benchmark, providing evidence of cross-family and cross-domain generalization.

Table 4: GSM8K accuracy (%) before and after post-merge fine-tuning on 100 examples.

	Unprotected	Protected
Before SFT	54.06	32.74
After SFT	52.70	1.90

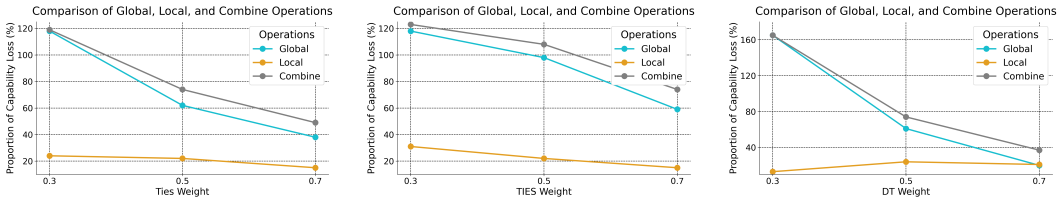
Table 5: Drop ratio (%) on MedQA before and after TA merging.

	Global	Local	Combined
Before merge	18.6	5.2	21.6
After merge	41.6	11.3	51.2

4.3 Merging Coefficients and Weight Perturbations

Effect of the merging coefficient To assess dependence on downstream merging, we keep the protected model fixed and vary the merging coefficient over $\alpha \in \{0.3, 0.5, 0.7\}$ (Figure 5). The protection coefficients λ_t remain unchanged, so this experiment varies the subsequent merging operation rather than the protection construction.

The combined model gives the largest reduction at each evaluated coefficient. Its merged accuracy falls below the corresponding base-model result at $\alpha = 0.3$ for TA and DT and at $\alpha \in \{0.3, 0.5\}$ for TIES. The observed effect therefore extends beyond the default coefficient, although its strength depends on how the downstream model is composed.



(a) TA (b) TIES (c) DT

Figure 5: Normalized reduction in transferred mathematical capability across downstream merging coefficients α for TA, TIES, and DT.

Effect of Gaussian weight perturbations At higher noise variances, Gaussian weight perturbations reveal larger capability drops in the processed models (Figure 6). All four models show increasing drop ratios with perturbation strength. At the smallest nonzero variance, the globally processed model drops slightly less than the specialized model, whereas the locally processed and combined models drop more. From a variance of 2×10^{-4} onward, all three processed models exhibit larger drops than the specialized model, with the separation generally widening as the perturbation increases. At the largest tested variance, the drop ratios are approximately 0.36 for the combined model, 0.18 for the specialized model, and 0.25 and 0.24 for the global and local models, respectively. Together with the retained accuracy of the unperturbed checkpoints, these results show that useful standalone capability can coexist with lower tolerance to subsequent parameter changes. Gaussian noise is an unstructured diagnostic, whereas TA, TIES, and DT apply method-specific task-vector updates; the experiment therefore complements the direct merging results without implying that the two types of perturbation follow the same directions or mechanism.

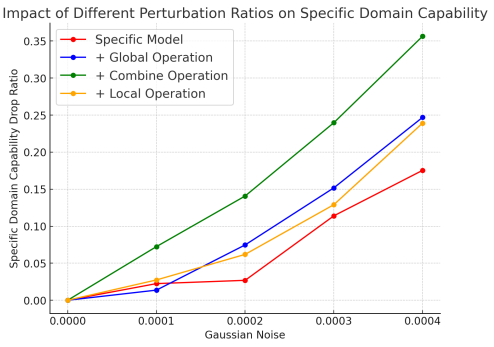


Figure 6: Target-domain accuracy drop under Gaussian weight perturbations of increasing variance.

4.4 Disentangling Reduced Transferability from Utility Degradation

Lower post-merge accuracy does not by itself establish reduced capability transfer: it may instead reflect broad utility degradation or weaker target-domain capability before merging. We examine these explanations using MMLU to assess broader utility beyond GSM8K and weaker unprotected models to test whether lower standalone GSM8K accuracy alone explains the post-merge results.

General-capability preservation We evaluate the specialized model and its processed variants on MMLU, which covers knowledge and reasoning across humanities, sciences, STEM, and other subjects, to test for utility loss beyond the protected GSM8K capability (Table 6).

Table 6: MMLU accuracy (%) of the specialized and processed models.

Task	Specialized	Global	Local	Combined
Humanities	53.11	52.99	52.94	52.43
Sciences	68.48	70.82	68.38	69.94
STEM	47.76	49.48	48.14	49.48
Other	66.91	67.43	66.62	67.52
Average	58.33	59.30	58.28	58.94

Average MMLU accuracy remains within 58.28%–59.30% across all variants. The combined model scores 58.94%, compared with 58.33% for the specialized model; its category-level changes range from a 0.68-point decrease on Humanities to a 1.72-point increase on STEM. The global and local variants likewise remain close to the specialized-model baseline. These results show no consistent utility degradation on MMLU; they do not establish an improvement in general capability.

Comparison with weaker unprotected models We compare the combined model with SciMistral-V1 and Mistral-Instruct-v0.2, two unprotected models with lower standalone GSM8K accuracy. Each is merged with BioMistral-7B under identical TA, DT, and TIES configurations, testing whether standalone capability alone explains the post-merge result (Table 7).

Table 7: GSM8K accuracy (%) before and after merging the combined model or weaker unprotected models with BioMistral-7B.

Model	Before merging	TA	DT	TIES
Combined model	65.07	32.74	31.92	18.95
SciMistral-V1	56.71	44.05	42.76	27.29
Mistral-Instruct-v0.2	42.10	43.82	38.44	34.95

Before merging, the combined model achieves 65.07% accuracy, exceeding the two unprotected models by 8.36 and 22.97 percentage points. After merging, this ordering reverses under every algorithm: the combined model trails both controls by 6.52–16.00 points. Its stronger standalone capability but lower post-merge performance cannot be explained by source-model accuracy alone, supporting the conclusion that protection reduces capability transfer rather than merely weakening the released checkpoint.

5 Conclusion

We presented a training-free method for protecting specialized models through direct weight processing. Guided by the goal of approaching a target-domain performance boundary, the global stage combines weight differences from weaker compatible models, while local refinement adjusts individual weights and preserves task-critical parameters. In the evaluated settings, the processed models retain most target-domain capability when used on their own while contributing less of that capability through subsequent merging. Component comparisons and perturbation analyses characterize the resulting protection–utility trade-off, showing that this form of protection can be obtained without an additional training stage.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript formatting and language editing.

ETHICS STATEMENT

This work aims to protect model intellectual property by reducing unauthorized capability transfer through merging. Such protection may help model owners control reuse, but it can also restrict otherwise beneficial model composition.

REPRODUCIBILITY STATEMENT

Section 3.3 defines the global perturbation, and Section 3.4 specifies the local edits and critical-weight selection. Section 4.1 reports the models, coefficients, calibration procedure, and merging settings. The experiments report standalone and post-merge performance for each processed variant, together with coefficient ablations and the post-merge fine-tuning configuration. Proposition 3.1 states the assumptions and proof of the directional analysis.

REFERENCES

Wei-Jia Chen, Min-Yan Tsai, Cheng-Yi Lee, and Chia-Mu Yu. Defending unauthorized model merging via dual-stage weight protection. In *CVPR*, 2026.

MohammadReza Davari and Eugene Belilovsky. Model breadcrumbs: Scaling multi-task model merging with sparse masks. In *ECCV*, 2025.

Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vlad Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. Arcee's mergekit: A toolkit for merging large language models. *arXiv preprint arXiv:2403.13257*, 2024.

Kuangpu Guo, Qingyan Zheng, Jian Liang, Yongcan Yu, Zilei Wang, Ran He, and Tieniu Tan. On the vulnerability of parameter-level defenses to model merging. *arXiv preprint arXiv:2606.30360*, 2026.

Xuanli He, Qiongkai Xu, Lingjuan Lyu, Fangzhao Wu, and Chenguang Wang. Protecting intellectual property of language generation apis with lexical watermark. In *AAAI*, 2022a.

Xuanli He, Qiongkai Xu, Yi Zeng, Lingjuan Lyu, Fangzhao Wu, Jiwei Li, and Ruoxi Jia. Cater: Intellectual property protection on text generation apis via conditional watermarks. In *NeurIPS*, 2022b.

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

Minwoo Jang, Hoyoung Kim, Jabin Koo, and Jungseul Ok. Making models unmergeable via scaling-sensitive loss landscape. In *ICML*, 2026.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. *arXiv preprint arXiv:2212.09849*, 2022.

Wei Junhao, Yu Zhe, and Jun Sakuma. Disrupting model merging: A parameter-level defense without sacrificing accuracy. In *ICCV*, 2025.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *ICML*, 2023.

Peixuan Li, Pengzhou Cheng, Fangqi Li, Wei Du, Haodong Zhao, and Gongshen Liu. Plmmark: A secure and robust black-box watermarking framework for pre-trained language models. In *AAAI*, 2023.

Qinfeng Li, Miao Pan, Jintao Chen, Fu Teng, Zhiqiang Shen, Ge Su, Hao Peng, and Xuhong Zhang. Do not merge my model! safeguarding open-source llms against unauthorized model merging. In *AAAI*, 2026.

Zhenyi Lu, Chenghao Fan, Wei Wei, Xiaoye Qu, Dangyang Chen, and Yu Cheng. Twin-merging: Dynamic integration of modular expertise in model merging. *arXiv preprint arXiv:2406.15479*, 2024.

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

Hope McGovern, Rickard Stureborg, Yoshi Suhara, and Dimitris Alikaniotis. Your large language models are leaving fingerprints. *arXiv preprint arXiv:2405.14057*, 2024.

Ding Sheng Ong, Chee Seng Chan, Kam Woh Ng, Lixin Fan, and Qiang Yang. Protecting intellectual property of generative adversarial networks from ambiguity attacks. In *CVPR*, 2021.

Wenjun Peng, Jingwei Yi, Fangzhao Wu, Shangxi Wu, Bin Zhu, Lingjuan Lyu, Binxing Jiao, Tong Xu, Guangzhong Sun, and Xing Xie. Are you copying my model? protecting the copyright of large language models for eaas via backdoor watermark. In *ACL*, 2023.

Yehonathan Refael, Adam Hakim, Lev Greenberg, Tal Aviv, Satya Lokam, Ben Fishman, and Shachar Seidman. Slip: Securing llms ip using weights decomposition. *arXiv preprint arXiv:2407.10886*, 2024.

Shuo Shao, Yiming Li, Hongwei Yao, Yiling He, Zhan Qin, and Kui Ren. Explanation as a watermark: Towards harmless and multi-bit model ownership verification via watermarking feature attribution. *arXiv preprint arXiv:2405.04825*, 2024.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. In *ICLR*, 2024.

Zihao Wang, Enneng Yang, Lu Yin, Shiwei Liu, and Li Shen. Model unmerging: Making your models unmergeable for secure model sharing. *arXiv preprint arXiv:2509.01548*, 2025.

Boyi Wei, Kaixuan Huang, Yangsibo Huang, Tinghao Xie, Xiangyu Qi, Mengzhou Xia, Prateek Mittal, Mengdi Wang, and Peter Henderson. Assessing the brittleness of safety alignment via pruning and low-rank modifications. *arXiv preprint arXiv:2402.05162*, 2024.

Jiashu Xu, Fei Wang, Mingyu Derek Ma, Pang Wei Koh, Chaowei Xiao, and Muhao Chen. Instructional fingerprinting of large language models. *arXiv preprint arXiv:2401.12255*, 2024.

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

Shojiro Yamabe, Tsubasa Takahashi, Futa Waseda, and Koki Wataoka. Mergeprint: Robust fingerprinting against merging large language models. *arXiv preprint arXiv:2410.08604*, 2024.

Peng Yang, Yingjie Lao, and Ping Li. Robust watermarking for deep neural networks via bi-level optimization. In *ICCV*, 2021.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *ICML*, 2024.

Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph. Stoecklin, Heqing Huang, and Ian M. Molloy. Protecting intellectual property of deep neural networks with watermarking. In *AsiaCCS*, 2018.

Jie Zhang, Dongdong Chen, Jing Liao, Weiming Zhang, Huamin Feng, Gang Hua, and Nenghai Yu. Deep model intellectual property protection via deep watermarking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4005–4020, 2021.

Jie Zhang, Dongrui Liu, Chen Qian, Linfeng Zhang, Yong Liu, Yu Qiao, and Jing Shao. Reef: Representation encoding fingerprints for large language models. *arXiv preprint arXiv:2410.14273*, 2024.