Gradient-Mine Units: Scorched-Earth Strategy for Model Protection against Unauthorized Fine-Tuning

Jinhyeok Jang[1,2] **Jaehong Kim**[1] **ByungOk Han**[1,2*]
[1]ETRI [2]UST
{jjh6297, jhkim504, byungok.han}@etri.re.kr

Abstract

Pretrained model weights are increasingly released under commercial licenses, usage restrictions, or other conditions that prohibit unauthorized fine-tuning. In practice, however, such misuse is difficult to detect or verify after the fact. This motivates a stronger objective for protected weight release: weights should remain useful for intended inference, yet become practically unattractive to repurpose through unauthorized gradient-based adaptation. We frame this objective as a *scorched-earth* strategy in parameter space: rather than only proving infringement after misuse occurs, the released weights themselves should react destructively when unauthorized fine-tuning begins. To realize this idea, we propose **Gradient-Mine Units (GMUs)**, a data-free weight-space protection mechanism for pretrained networks. GMUs are planted into selected feedforward layers as hidden units with extreme internal scale, while a locking mechanism keeps them silent at initialization so that the original inference behavior is preserved. During fine-tuning, this locked state is progressively broken, allowing the planted units to emit amplified gradients that disrupt the model's native adaptation dynamics. We formulate GMUs in a general feedforward setting and show that gated instantiations naturally provide an additional hard lock. Empirically, we validate the method on both large language models and Vision Transformers. Across multiple architectures and downstream tasks, GMUs preserve pre-fine-tuning utility while substantially degrading or destabilizing standard fine-tuning. These results suggest that protected weight release can move beyond post-hoc attribution toward a practical deterrence mechanism for unauthorized adaptation. Our implementation is available here.

Warning: This paper presents a method with potential dual-use risks.

1 Introduction

Pretrained models have become high-value deployable assets, and some released weights are governed by licenses that restrict commercial use or downstream repurposing. Examples include YOLO-NAS [1], Stanford Alpaca [2], LLaMA [3], Orca 2 [4], Mistral Large 2 [5], and Codestral-22B [6]. These cases highlight the need for technical mechanisms to discourage license-violating repurposing through fine-tuning, even after model weights are distributed.

However, once weights are released, unauthorized gradient-based fine-tuning is easy to perform, while post-hoc verification is often uncertain or delayed. Existing protections [7, 8, 9, 10] only partially address this asymmetry: watermarking can aid attribution but does not prevent misuse, while other defenses often rely on target-domain data, access control, or downstream evidence. We therefore pursue a different objective: released weights should remain useful for intended inference but become practically unattractive to repurpose through unauthorized fine-tuning. We frame this as

*Corresponding author

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

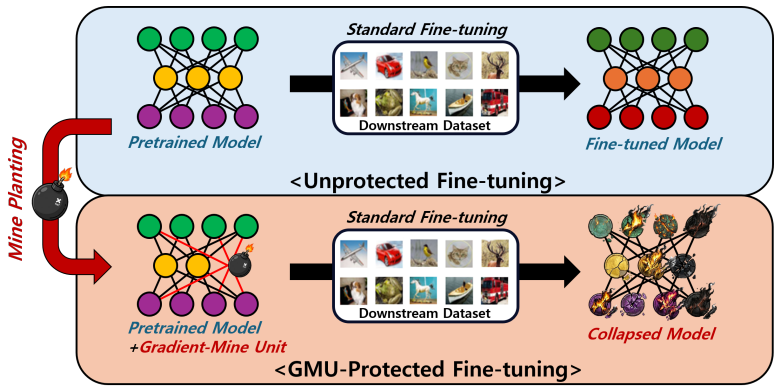


Figure 1: Conceptual overview of the proposed scorched-earth strategy. Standard fine-tuning of an unprotected pretrained model yields a normally adapted model, whereas post-hoc GMU injection preserves the released model for inference but causes fine-tuning to collapse into a destructive regime.

a *scorched-earth* strategy in parameter space, where the protected model behaves normally before adaptation but standard fine-tuning drives it into a damaged optimization regime.

To this end, we propose **Gradient-Mine Units (GMUs)**, a data-free post-hoc weight-space protection mechanism. GMUs repurpose a small number of feedforward channels as hidden mine slots with extreme internal scale and microscopic readout. A symmetry lock keeps these slots silent at initialization, preserving inference behavior up to negligible error. During ordinary fine-tuning, gradient updates break this lock, allowing the previously silent slots to produce non-cancelled signals and disproportionately large gradients that disrupt native adaptation. GMUs therefore implement a locking–unlocking design: they are benign before adaptation but become destructive once optimization begins. The mechanism requires no data, no retraining, and operates directly in parameter space while preserving the original layer width. Experiments across vision and language models show that GMUs maintain inference-time utility while substantially degrading standard fine-tuning, supporting proactive deterrence as an alternative to post-hoc misuse detection.

Our contributions can be summarized as follows:

1. **Scorched-earth strategy for pretrained weights:** moving beyond post-hoc ownership verification toward released weights that remain useful for inference but become practically unattractive for unauthorized fine-tuning.
2. **Gradient-Mine Units (GMUs):** a hidden-unit injection mechanism with a locking–unlocking design that keeps planted units silent at initialization but releases amplified, destabilizing updates once fine-tuning breaks the lock.
3. **A data-free post-hoc protection method with broad validation:** GMUs directly modify pretrained parameters without retraining and are validated on both LLMs and ViTs.

2 Related Work

Non-transferable learning. Recent works seek to restrict the downstream adaptability of released models. SOPHON [7] and NTP [11] reduce transferability to specified target domains, while PMP [12] limits unauthorized fine-tuning through a private masked subnetwork. However, these methods typically require target-domain data, training data, or pretraining-stage intervention; PMP also introduces substantial redundant parameters. In contrast, GMUs are injected *post-hoc*, require no data or retraining, add negligible overhead, and are designed not merely to weaken adaptation but to drive unauthorized fine-tuning into a destructive regime.

Weight poisoning and watermarking. Weight poisoning methods [13, 14] modify pretrained weights so that hidden behaviors or backdoors emerge after fine-tuning, but their goal is offensive manipulation rather than IP protection or deterrence. Model watermarking methods [8, 15, 16, 17, 9, 10], including FT-Shield [9] and SleeperMark [10], embed signatures that remain verifiable after fine-tuning. These methods support post-hoc ownership attribution, whereas GMUs aim to proactively make unauthorized fine-tuning unreliable and practically unattractive.

3 Motivation & Problem Definition: A Scorched-Earth Strategy

We frame protected weight release as a *scorched-earth* strategy against unauthorized fine-tuning as shown in Fig. 1. Rather than only detecting misuse after it occurs, the goal is to release weights that remain useful for inference but become practically unusable under gradient-based adaptation. We achieve this by planting hidden destructive units that remain silent while locked, but become collapse-inducing once ordinary fine-tuning breaks that lock.

This setting is motivated by the fact that pretrained weights may be distributed under licenses or usage restrictions that make unauthorized fine-tuning undesirable. Yet, once weights are released, such misuse is often difficult to reliably detect or verify. We therefore seek protected weights that preserve the original inference utility while failing under standard fine-tuning.

Let Θ denote the original pretrained parameters, $\widetilde{\Theta}$ the released protected parameters, and $F(x;\Theta)$ the model output for input x under parameters Θ. Our objective is to construct $\widetilde{\Theta}$ such that:

1. **Inference Preservation:** $F(x;\widetilde{\Theta}) \approx F(x;\Theta)$ for normal inference inputs.
2. **Fine-Tuning Fragility:** Under standard fine-tuning, the planted mechanism becomes unlocked and induces destabilizing updates that disrupt adaptation for downstream tasks.

4 Method

We propose **Gradient-Mine Units (GMUs)**, a hidden-unit injection mechanism for pretrained networks. GMUs are planted in selected feedforward layers and initialized to remain silent at inference time while maintaining an extreme internal scale. Under fine-tuning, this locked state can be broken, allowing the planted units to generate gradients that dominate the native adaptation.

4.1 Planting Gradient-Mine Units in Feedforward Layers

Consider a feedforward layer at block ℓ with input feature $\mathbf{h}_\ell \in \mathbb{R}^{d_\ell}$ and output

$$f_\ell(\mathbf{h}_\ell) = \mathbf{V}_\ell\, \phi(\mathbf{U}_\ell \mathbf{h}_\ell + \mathbf{b}_\ell) + \mathbf{c}_\ell, \tag{1}$$

where $\mathbf{U}_\ell \in \mathbb{R}^{m_\ell \times d_\ell}$, $\mathbf{V}_\ell \in \mathbb{R}^{d_\ell \times m_\ell}$, $\mathbf{b}_\ell \in \mathbb{R}^{m_\ell}$, $\mathbf{c}_\ell \in \mathbb{R}^{d_\ell}$, and $\phi(\cdot)$ is an elementwise activation. Rather than appending new hidden units, we repurpose K existing hidden units in a subset of layers $\mathcal{S}$. Specifically, we select a **low-norm** or **randomly sampled** subset of channels $\mathcal{M}_\ell \subset \{1,\dots,m_\ell\}$ with $|\mathcal{M}_\ell| = K$ and overwrite those selected slots with GMU parameters. For notational simplicity, assume the selected channels are ordered last after a permutation. Then the original parameters can be partitioned as

$$\mathbf{U}_\ell = \begin{bmatrix} \mathbf{U}_\ell^{(n)} \\ \mathbf{U}_\ell^{(r)} \end{bmatrix}, \qquad \mathbf{b}_\ell = \begin{bmatrix} \mathbf{b}_\ell^{(n)} \\ \mathbf{b}_\ell^{(r)} \end{bmatrix}, \qquad \mathbf{V}_\ell = \begin{bmatrix} \mathbf{V}_\ell^{(n)} & \mathbf{V}_\ell^{(r)} \end{bmatrix}, \tag{2}$$

where (n) denotes the retained native channels and (r) denotes the selected replaceable channels.

Let the GMU parameters written into the selected slots be

$$\mathbf{U}_\ell^{(m)} \in \mathbb{R}^{K \times d_\ell}, \qquad \mathbf{b}_\ell^{(m)} \in \mathbb{R}^{K}, \qquad \mathbf{V}_\ell^{(m)} \in \mathbb{R}^{d_\ell \times K}. \tag{3}$$

The protected parameters are then

$$\widetilde{\mathbf{U}}_\ell = \begin{bmatrix} \mathbf{U}_\ell^{(n)} \\ \mathbf{U}_\ell^{(m)} \end{bmatrix}, \qquad \widetilde{\mathbf{b}}_\ell = \begin{bmatrix} \mathbf{b}_\ell^{(n)} \\ \mathbf{b}_\ell^{(m)} \end{bmatrix}, \qquad \widetilde{\mathbf{V}}_\ell = \begin{bmatrix} \mathbf{V}_\ell^{(n)} & \mathbf{V}_\ell^{(m)} \end{bmatrix}. \tag{4}$$

The protected layer becomes

$$\widetilde{f}_\ell(\mathbf{h}_\ell) = \widetilde{\mathbf{V}}_\ell \phi(\widetilde{\mathbf{U}}_\ell \mathbf{h}_\ell + \widetilde{\mathbf{b}}_\ell) + \mathbf{c}_\ell = f_\ell(\mathbf{h}_\ell) - \mathbf{V}_\ell^{(r)} \mathbf{z}_\ell^{(r)} + \mathbf{V}_\ell^{(m)} \mathbf{z}_\ell^{(m)}. \tag{5}$$

Here, $\mathbf{z}_\ell^{(r)} = \phi(\mathbf{U}_\ell^{(r)} \mathbf{h}_\ell + \mathbf{b}_\ell^{(r)})$ denotes the activation of the replaced native channels, and $\mathbf{z}_\ell^{(m)} = \phi(\mathbf{U}_\ell^{(m)} \mathbf{h}_\ell + \mathbf{b}_\ell^{(m)})$ denotes the GMU activation. Since the replaced channels have minimal contribution and the initial GMU readout is silent, the protected layer preserves the original mapping with negligible error while keeping the layer width unchanged.

4.2 Extreme-Scale Mine Construction

A GMU is designed to become dangerous only after unlocking. Its role is not merely to create large hidden activations, but to create the potential for *disproportionately large gradients* once adaptation breaks the lock. To this end, we impose an extreme scale separation between the repurposed GMU slots and the retained native pretrained hidden units.

We initialize the mine encoder using pairwise shared random directions:

$$\mathbf{U}_\ell^{(m)} = \Omega \begin{bmatrix} \mathbf{P}_\ell \\ \mathbf{P}_\ell \end{bmatrix}, \qquad \mathbf{b}_\ell^{(m)} = \mathbf{0}, \tag{6}$$

where $\Omega \gg 1$ is a large excitation scale and $\mathbf{P}_\ell \in \mathbb{R}^{K/2 \times d_\ell}$ contains independently sampled random vectors. Thus, GMU slots share identical encoder weights and biases within each pair.

To keep the initial forward contribution negligible, we use a *large-in/small-out* construction: encoder-side weights are set to an extreme scale Ω, while the downstream readout is initialized at a microscopic scale $\varepsilon \ll 1$. The role of Ω is to amplify sensitivity to updates. Once the GMU slots enter an active regime, their pre-activation and effective hidden signal inherit the scale Ω, so the resulting parameter updates can become much larger than those of ordinary pretrained units. The microscopic readout scale ε plays the opposite role: it keeps the mine contribution negligible at initialization so that the extreme internal scale does not immediately corrupt the forward pass.

For gated feedforward layers such as GLU- or SwiGLU-style MLPs, the hidden state is formed by the product of a gate path and an excitation path. In this setting, our GMU instantiation adopts a two-path construction in the selected slots: the gate-side weights are initialized to zero so that the slots remain inactive at initialization, while the excitation-side weights are initialized at the extreme scale Ω. As a result, the GMU slots are silent before adaptation, yet retain the potential to emit disproportionately large gradients once fine-tuning breaks the lock. For nongated layers, one may optionally introduce a pairwise shared negative bias to further suppress the selected slots at initialization, but we treat this as an implementation detail rather than a core requirement of the method.

4.3 Symmetry Lock for Silent Inference

A large hidden mine would be useless if it immediately corrupted the pretrained model. We therefore impose a *symmetry lock* that neutralizes the GMU readout at initialization.

Because paired rows of $\mathbf{U}_\ell^{(m)}$ share identical encoder weights and biases, their activations are initially aligned. Thus, for any input $\mathbf{h}_\ell$, the GMU activations satisfy

$$\mathbf{z}_\ell^{(m)} = \begin{bmatrix} \mathbf{a}_\ell \\ \mathbf{a}_\ell \end{bmatrix}, \qquad \mathbf{a}_\ell = \phi(\Omega \mathbf{P}_\ell \mathbf{h}_\ell) \in \mathbb{R}^{K/2}. \tag{7}$$

At the same time, we initialize the GMU readout as

$$\mathbf{V}_\ell^{(m)} = \left[\mathbf{V}_\ell^{(+)},\ \mathbf{V}_\ell^{(-)}\right] \in \mathbb{R}^{d_\ell \times K}, \qquad \mathbf{V}_\ell^{(-)} = -\mathbf{V}_\ell^{(+)}, \tag{8}$$

where $\mathbf{V}_\ell^{(+)} \in \mathbb{R}^{d_\ell \times K/2}$ has entries sampled at the microscopic scale ε.

Substituting Eq. (7) into the GMU readout yields

$$\begin{aligned} \mathbf{V}_\ell^{(m)} \mathbf{z}_\ell^{(m)} &= \left[\mathbf{V}_\ell^{(+)},\ \mathbf{V}_\ell^{(-)}\right] \begin{bmatrix} \mathbf{a}_\ell \\ \mathbf{a}_\ell \end{bmatrix} \\ &= \left(\mathbf{V}_\ell^{(+)} + \mathbf{V}_\ell^{(-)}\right) \mathbf{a}_\ell = \mathbf{0}. \end{aligned} \tag{9}$$

Therefore, the GMU slots are internally extreme but externally silent at initialization. Since these slots replace native channels characterized by minimal contributions, the protected layer preserves the original mapping with negligible replacement error:

$$\widetilde{f}_\ell(\mathbf{h}_\ell) \simeq f_\ell(\mathbf{h}_\ell). \tag{10}$$

This symmetry lock is the default inference-preserving mechanism of GMUs. Silence follows directly from the combination of pairwise channel alignment and anti-symmetric readout cancellation, together with the explicit large-in/small-out scale separation.

4.4 Norm Camouflage against Weight Filtering

A natural question is whether GMUs can be removed by simple weight-norm filtering. Since GMUs contain extreme-scale encoder weights, an adversary may try to identify large-norm rows and prune or freeze them before fine-tuning. To make such filtering less selective, we introduce *norm camouflage*: instead of reducing the GMU scale, we randomly select a small subset of native pretrained channels across multiple layers and reparameterize them to have GMU-like norms while preserving, or approximately preserving, the original function.

For example, this can be done exactly by reciprocal query/key scaling in attention layers. Let $\mathcal{R}_\ell$ be a randomly selected subset of attention channels at layer ℓ, and let $\mathbf{D}_\ell$ be diagonal with $(\mathbf{D}_\ell)_{ii} = \Omega$ if $i \in \mathcal{R}_\ell$ and 1 otherwise. We apply

$$\mathbf{Q}_\ell \leftarrow \mathbf{Q}_\ell \mathbf{D}_\ell, \qquad \mathbf{K}_\ell \leftarrow \mathbf{K}_\ell \mathbf{D}_\ell^{-1}, \qquad (\mathbf{Q}_\ell \mathbf{D}_\ell)(\mathbf{K}_\ell \mathbf{D}_\ell^{-1})^\top = \mathbf{Q}_\ell \mathbf{K}_\ell^\top. \tag{11}$$

Equivalently, selected rows of the query projection are scaled by Ω, while the corresponding rows of the key projection are scaled by Ω^{-1}. Thus, the model contains native high-norm weights comparable to GMUs, but the attention logits are unchanged up to numerical precision.

A similar pseudo-equivalent transformation can be applied to a randomly selected subset $\mathcal{R}_\ell^{\mathrm{ffn}}$ of native feedforward channels:

$$\mathbf{u}_{\ell,j} \leftarrow \Omega \mathbf{u}_{\ell,j}, \quad b_{\ell,j} \leftarrow \Omega b_{\ell,j}, \quad \mathbf{v}_{\ell,j} \leftarrow \Omega^{-1} \mathbf{v}_{\ell,j}, \qquad j \in \mathcal{R}_\ell^{\mathrm{ffn}}. \tag{12}$$

where $\mathbf{u}_{\ell,j}$, $b_{\ell,j}$, and $\mathbf{v}_{\ell,j}$ denote the j-th row of $\mathbf{U}_\ell$, the j-th entry of $\mathbf{b}_\ell$, and the j-th column of $\mathbf{V}_\ell$, respectively. This is exact for positively homogeneous activations and approximate for GELU-style feedforward layers, where we empirically observe negligible degradation when applied to a small random subset.

Norm camouflage makes large-norm filtering costly. After camouflage, large-norm rows no longer correspond only to GMUs; they also include native channels that participate in the pretrained computation. Therefore, pruning all large-norm weights may remove the mines, but it also removes important pretrained components and damages the model's original utility. This provides an additional protection layer against simple norm-based filtering attacks.

4.5 Unlocking Under Fine-Tuning

The silence of a GMU holds only at the locked initialization. Once fine-tuning begins, standard gradient updates perturb the conditions that enforce cancellation and turn the planted units into an active source of harmful updates.

1) Breaking the lock. At initialization, the planted units are aligned and the readout satisfies $\mathbf{V}_\ell^{(-)} = -\mathbf{V}_\ell^{(+)}$, so the GMU contribution is cancelled. Fine-tuning perturbs this relation and weakens the lock, allowing a previously suppressed GMU signal to enter the forward computation.

2) Dominating the update direction. After the lock is weakened, the extreme internal scale Ω becomes active in the backward pass. Even if the update magnitude is bounded, the GMU-related gradients can dominate the direction of the update relative to native pretrained parameters. As a result, fine-tuning is no longer governed by the pretrained model's ordinary adaptation dynamics.

3) Destabilized Fine-tuning. The activated GMU signal perturbs subsequent hidden states, which can further weaken downstream locks and create a cascading effect. Consequently, optimization is driven into a damaged regime: the loss may spike, and downstream adaptation remains unreliable.

5 Experiments

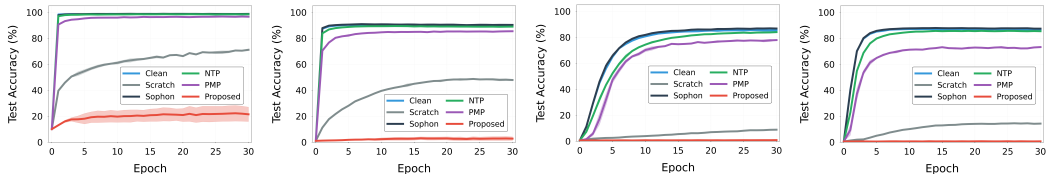
This section evaluates our proposed GMUs on ImageNet-pretrained ViTs [18] and LLMs, demonstrating that they remain silent during inference while becoming destructive under fine-tuning.

5.1 ViT with Gradient-Mine Units: Silent Inference, Fragile Fine-Tuning

We first evaluate our method on Vision Transformers. Under our hypothesis, GMU-protected models should undergo severe instability during fine-tuning, while clean models adapt successfully.

Table 1: Accuracy (%) on four downstream benchmarks for ViT. Note that lower fine-tuning accuracy serves as an indicator of robust model protection.

Method	#Params (%)	No Fine-tuning on Source?	Target Agnostic?	Injection Time (h)	Before FT (%) ↑	After FT (%) ↓			
					ImageNet	CIFAR10	Food101	Cars196	CUB200
Clean	100.00	✓	N/A	N/A	81.66	98.68 ± 0.06	89.78 ± 0.15	85.51 ± 0.47	86.77 ± 0.23
Random Init.	100.00	✓	N/A	N/A	0.10	71.17 ± 0.79	47.87 ± 0.69	8.82 ± 0.10	14.28 ± 0.53
SOPHON [7]	100.00	✗	✗	∼2	80.80	98.81 ± 0.06	90.32 ± 0.13	86.68 ± 0.23	87.43 ± 0.42
NTP [11]									
$sparsity = 0.5$	50.00	✗	✗	∼2	81.53	98.73 ± 0.09	89.80 ± 0.08	85.51 ± 0.26	86.69 ± 0.37
$sparsity = 0.7$	30.00	✗	✗	∼6	78.68	98.52 ± 0.12	89.00 ± 0.13	83.99 ± 0.37	85.47 ± 0.19
PMP [12]									
$mask = 0.7$	128.04	✗	✓	∼12	70.21	96.64 ± 0.23	85.45 ± 0.17	77.86 ± 0.37	73.24 ± 0.70
$mask = 0.8$	116.34	✗	✓	∼12	72.69	96.94 ± 0.22	86.16 ± 0.23	79.28 ± 0.97	74.50 ± 0.45
$mask = 0.9$	107.27	✗	✓	∼12	76.67	97.15 ± 0.19	86.74 ± 0.14	82.31 ± 0.46	78.05 ± 0.72
Proposed									
Equiv. Camouflage	100.00	✓	✓	5.25×10^{-4}	81.64	$\mathbf{21.44 \pm 5.56}$	$\mathbf{2.70 \pm 1.41}$	$\mathbf{0.80 \pm 0.06}$	$\mathbf{0.62 \pm 0.09}$
Pseudo-Equiv. Camouflage	100.00	✓	✓	1.35×10^{-3}	81.34	$\mathbf{37.89 \pm 0.87}$	$\mathbf{8.42 \pm 0.44}$	$\mathbf{2.02 \pm 0.20}$	$\mathbf{2.60 \pm 0.38}$



CIFAR10 Food101 Cars196 CUB200

Figure 2: Test accuracy curves of fine-tuning ImageNet-pretrained ViT on downstream tasks.

Setup. We use an ImageNet [19]-pretrained ViT-B/16 [18] and evaluate transfer to four downstream datasets: CIFAR10 [20], Food101 [21], Cars196 [22], and CUB200 [23]. We compare our method against three baselines: (1) SOPHON [7], (2) NTP [11] with two sparsity levels, $s \in \{0.5, 0.7\}$, and (3) a post-hoc PMP-style baseline inspired by PMP [12] with three mask ratios, $m \in \{0.7, 0.8, 0.9\}$. For our method, we plant GMUs with $K = 8, \Omega = 10^7$ in four layers and evaluate two camouflage strategies (Equivalent & Pseudo-Equivalent). Because the original PMP is inherently a pretraining-stage method, we adapt it to the post-hoc setting by designating a subset of original neurons as the trainable masked subnetwork, while inserting additional randomly initialized neurons that remain fixed to mimic the complementary inactive part. This adapted baseline is trained on the ImageNet training set. For SOPHON, we follow its target-aware formulation using two loss terms: a positive retention loss on ImageNet and an adversarial loss on a target dataset. Since both SOPHON and NTP require a designated target dataset, we use CIFAR100, which is not included among our downstream evaluation tasks, in order to assess generality. For all downstream benchmarks, we report test accuracy on the corresponding test splits. Fine-tuning is performed with AdamW for 30 epochs using standard image augmentations, including random rotation, horizontal flipping, and random cropping. For each dataset, we repeat fine-tuning five times in float32 and report the mean and standard deviation.

Results. The results are shown in Table 1 and Fig. 2. For clarity, Fig. 2 includes Clean, Random Scratch, SOPHON, NTP with $s = 0.7$, the post-hoc PMP baseline with $m = 0.7$, and GMU with equivalent camouflage. Before discussing downstream transfer, we note three important differences from our method: (1) SOPHON, NTP, and PMP all require additional training cost, (2) SOPHON and NTP are target-specific methods, and (3) PMP requires substantial parameter overhead. In contrast, GMU is data-free in the post-hoc setting and introduces no additional parameters. Before fine-tuning, the clean model and the GMU-protected model achieve nearly identical accuracy on the ImageNet validation set (both around 81.65%), showing that GMU injection preserves the original pretrained capability. By contrast, the other protection baselines already exhibit slight degradation on the source task. Once fine-tuning begins, the clean model adapts successfully and reaches high test accuracy on all downstream datasets. SOPHON fails to provide meaningful protection in the unseen setting, while NTP shows only limited degradation on unseen datasets. The post-hoc PMP baseline partially reduces fine-tuning performance, but still allows substantial downstream adaptation. In contrast, GMU-protected models achieve substantially lower accuracy than models trained from scratch. These results show that our method preserves the original pretrained utility while substantially disrupting fine-tuning in vision models.

5.2 Application to pretrained LLMs

We further evaluate our method on multiple pretrained LLMs by comparing pre- and post-fine-tuning performance and training dynamics. In this setting, other prior works are not applicable because they require access to the pretraining data, which is unavailable for modern LLMs.

Table 2: Accuracy (%) on three benchmarks for LLMs with and without GMU. Lower accuracy after fine-tuning indicates that GMUs more strongly disrupt the fine-tuning process.

Model	Method	Before FT (%) ↑			After FT (%) ↓		
		SST2	ScienceQA	LogiQA	SST2	ScienceQA	LogiQA
Gemma3-4B (Full)	Clean	92.66	78.10	35.79	95.95 ± 0.27	89.97 ± 0.39	39.02 ± 1.27
	Proposed	92.43	78.28	35.79	$\mathbf{80.20 \pm 2.35}$	$\mathbf{53.54 \pm 4.29}$	$\mathbf{26.11 \pm 0.45}$
Gemma3-4B (LoRA)	Clean	92.66	78.10	35.79	95.37 ± 0.11	87.34 ± 0.18	34.77 ± 0.63
	Proposed	92.20	78.06	35.79	$\mathbf{60.17 \pm 1.93}$	$\mathbf{46.91 \pm 2.20}$	$\mathbf{26.98 \pm 1.38}$
Gemma3-1B (Full)	Clean	57.45	58.45	24.27	94.34 ± 0.33	85.42 ± 0.26	26.96 ± 1.31
	Proposed	58.14	58.63	24.12	$\mathbf{69.76 \pm 3.99}$	$\mathbf{52.28 \pm 0.59}$	$\mathbf{24.32 \pm 1.01}$
Gemma3-1B (LoRA)	Clean	57.45	58.45	24.27	93.00 ± 0.28	72.14 ± 0.32	25.55 ± 0.51
	Proposed	58.14	58.63	24.12	$\mathbf{54.01 \pm 1.95}$	$\mathbf{43.17 \pm 2.07}$	$\mathbf{25.04 \pm 0.33}$
Qwen3-1.7B (Full)	Clean	83.60	67.31	31.80	95.03 ± 0.27	85.09 ± 7.86	41.68 ± 1.10
	Proposed	83.60	67.54	32.41	$\mathbf{56.19 \pm 2.50}$	$\mathbf{52.92 \pm 1.43}$	$\mathbf{25.81 \pm 0.90}$
Qwen3-1.7B (LoRA)	Clean	83.60	67.31	31.80	94.76 ± 0.27	82.27 ± 0.12	39.99 ± 1.01
	Proposed	83.60	67.54	32.41	$\mathbf{50.73 \pm 0.22}$	$\mathbf{43.11 \pm 1.20}$	$\mathbf{27.75 \pm 0.07}$
Qwen3-0.6B (Full)	Clean	75.34	52.20	20.43	93.77 ± 0.24	87.34 ± 1.29	29.80 ± 1.45
	Proposed	75.34	51.93	20.43	$\mathbf{63.87 \pm 7.01}$	$\mathbf{50.49 \pm 0.53}$	$\mathbf{27.59 \pm 0.52}$
Qwen3-0.6B (LoRA)	Clean	75.34	52.20	20.43	92.39 ± 0.52	73.03 ± 0.46	33.23 ± 1.21
	Proposed	75.34	51.93	20.43	$\mathbf{42.67 \pm 4.22}$	$\mathbf{42.24 \pm 0.50}$	$\mathbf{24.68 \pm 3.20}$

Setup. We evaluate GMUs ($K = 8$ and $\Omega = 10^7$) on four pretrained LLMs: Gemma3 [24] (4B and 1B), Qwen3 [25] (1.7B and 0.6B) with bf16. Experiments are conducted on three text-based downstream tasks: SST2 [26], ScienceQA (text-only) [27], and LogiQA [28]. We also evaluate LoRA tuning [29], a prominent PEFT method, by applying it to all clean neurons and GMUs. To simulate unauthorized adaptation, we fine-tune both clean pretrained models and GMU-protected models, with GMUs inserted into the last four layers, using AdamW [30] for three epochs. For each dataset, we repeat fine-tuning five times and report the mean and standard deviation.

Results. As shown in Table 2, GMUs preserve inference-time utility, causing only negligible changes in pre-fine-tuning performance and occasionally even slight improvements. After fine-tuning, however, clean pretrained models successfully adapt across all architectures and datasets under both full fine-tuning and LoRA, whereas GMU-protected models consistently fail to achieve effective downstream adaptation. The degradation is observed in both training regimes, and is even more pronounced under LoRA, indicating that lightweight adaptation does not bypass the GMU mechanism. Overall, these results show that GMUs substantially reduce the practical transfer utility of pretrained knowledge, making unauthorized fine-tuning unreliable and unattractive in practice.

6 Discussion

This section further analyzes the mechanisms and impacts of our approach.

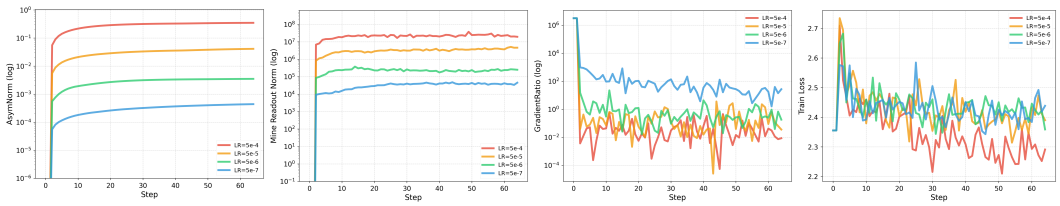
6.1 Analysis of the Unlocking Process

We analyze how the planted GMUs are unlocked during fine-tuning using an ImageNet-pretrained ViT-B/16 on CIFAR10 with four learning rates ($[5e-4, 5e-5, 5e-6, 5e-7]$) and AdamW. Following Sec. 4.5, we focus on four signals of this transition: the breakdown of the anti-symmetric readout structure, the growth of the non-cancelled mine readout, the emergence of GMU-dominated gradients, and the resulting degradation of optimization. Let $\mathbf{V}_{\ell,t}^{(m)} = [\mathbf{V}_{\ell,t}^{(+)}, \mathbf{V}_{\ell,t}^{(-)}]$ denote the GMU readout at layer ℓ and step t. We then measure

$$\mathrm{AsymNorm}_{\ell,t} = \left\| \mathbf{V}_{\ell,t}^{(+)} + \mathbf{V}_{\ell,t}^{(-)} \right\|_F, \qquad \mathrm{ReadoutNorm}_{\ell,t} = \left\| \mathbf{V}_{\ell,t}^{(m)} \mathbf{z}_{\ell,t}^{(m)} \right\|_2, \tag{13}$$

$$\mathrm{GradientRatio}_t = \frac{\|\nabla_{\Theta^{(m)}} \mathcal{L}_t\|_2}{\|\nabla_{\Theta^{(n)}} \mathcal{L}_t\|_2 + \delta_{\mathrm{stab}}}. \tag{14}$$

Here, $\Theta^{(m)}$ and $\Theta^{(n)}$ denote the GMU and native model parameters, respectively, $\mathcal{L}_t$ is the training loss at step t, and $\delta_{\mathrm{stab}} > 0$ is a numerical stabilizer. $\mathrm{AsymNorm}_{\ell,t}$ measures the extent to which the anti-symmetric readout is broken, $\mathrm{ReadoutNorm}_{\ell,t}$ measures the magnitude of the non-cancelled GMU signal entering, and $\mathrm{GradientRatio}_t$ indicates when the planted units begin to dominate the native pretrained units in gradient magnitude. We additionally track the training loss to examine the resulting optimization behavior. As shown in Fig. 3, unlocking occurs almost immediately



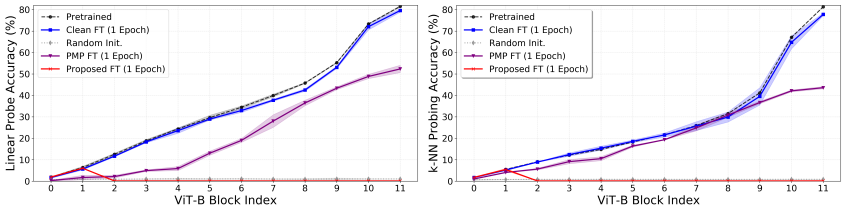
AsymNorm | Mine Readout Norm | Gradient Ratio | Train Loss

Figure 3: Unlocking dynamics in ViT. Fine-tuning first breaks channel-wise symmetry, then releases the mine readout, after which GMU-dominated gradients emerge.

in ViT. **AsymNorm** becomes non-zero at the first step, indicating that the initial anti-symmetric readout is broken as soon as fine-tuning begins. **ReadoutNorm** also increases at this stage, showing that the mine signal is no longer cancelled and starts to enter the forward computation. Notably, **GradientRatio** exhibits an extremely large value at the first step and then gradually decreases, suggesting that GMU gradients dominate the earliest update and strongly perturb the model before later optimization proceeds. The **Train Loss** follows the same pattern: it is already severely corrupted in the first step, and although it may later decrease, it only recovers around the initial-loss regime rather than reaching normal fine-tuning behavior. These results indicate that GMU-induced failure is triggered almost entirely at the beginning of fine-tuning: the pretrained representation is damaged by the first update, and subsequent training proceeds from this corrupted state rather than from the original pretrained knowledge.

6.2 Block-wise Probe Analysis for Knowledge Collapse

To examine whether pretrained knowledge remains usable after collapse, we perform block-wise probing on a ViT. We freeze the backbone, extract each block feature, apply layer normalization, and evaluate its class-discriminability. Using an ImageNet-pretrained ViT, we split the ImageNet validation set into 50% probe data and 50% test data. For each block, we evaluate both linear probing and k-NN probing with $k = 20$. We compare the intact pretrained model, the same model after standard fine-tuning on CIFAR10, a randomly initialized ViT, a PMP-protected ViT after one epoch of fine-tuning, and a ViT with GMUs at blocks #2-5 after one epoch of CIFAR10 fine-tuning. Results are averaged over five runs. Fig. 4 shows that GMUs induce much more severe representational damage than PMP. In both settings, the GMU-protected model matches the pretrained model at block #0-1, and even there the accuracy is low; from block #2 onward, it collapses to nearly the random-initialization baseline. By contrast, PMP only partially reduces the probing accuracy and still preserves substantially more pretrained information across multiple blocks. This indicates that GMU-induced collapse makes pretrained knowledge largely unrecoverable from intermediate representations, thereby preventing practical reuse of the protected backbone.



Linear Probing | k-NN probing

Figure 4: Block-wise probe analysis for knowledge collapse in ViT. For each Transformer block, we evaluate both linear probing and k-NN probing ($k = 20$) using frozen block features. After fine-tuning a GMU-protected model, the probe accuracy drops to the random-initialization baseline.

6.3 Ablation Study on Ω

We additionally provide a simple ablation on the excitation scale Ω. We fine-tune ViT-B/16 on CIFAR10 using the same protocol as above, while varying $\Omega \in \{10^0, 10^1, \ldots, 10^9\}$. As shown in Fig. 5, the test accuracy after 30 epochs starts to decrease noticeably from $\Omega = 10^3$ and continues to degrade as Ω increases. When $\Omega \geq 10^8$, the accuracy drops to near chance level. This confirms that the destructive effect of GMUs is strongly controlled by the extreme internal scale.

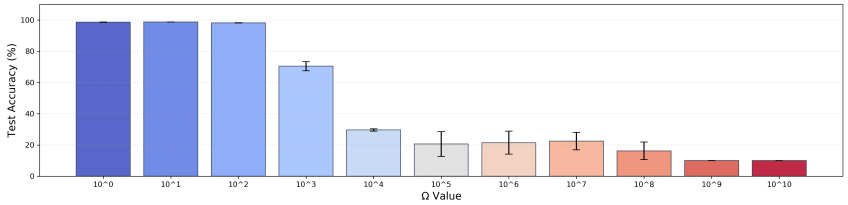


Figure 5: Test accuracy (%) on CIFAR10 for ViT with various Ω values

6.4 Locking Mechanisms and Ablation in Nongated GMUs

For nongated GELU feedforward layers, we employ three complementary mechanisms to suppress GMU contributions at initialization. First, **negative bias** is introduced into $b_\ell^{(m)}$ to suppress activations and their derivatives when pre-activations lie in GELU's negative-saturation region. Unlike the zero-bias initialization in Sec. 4.2, the nongated variant uses a shared negative bias across GMU channels. Second, **microscopic readout** weights at scale ε attenuate non-cancelled GMU contributions. Third, **symmetry cancellation** uses identical incoming parameters and anti-symmetric readout weights, $V_\ell^{(-)} = -V_\ell^{(+)}$, ensuring $V_\ell^{(m)} z_\ell^{(m)} = 0$ at initialization in exact arithmetic, as established in Sec. 4.3. Together, these mechanisms preserve the symmetry lock while further suppressing potential residual contributions.

Table 3 isolates the contribution of each mechanism. Negative bias alone reduces source accuracy and exhibits substantial variability during fine-tuning. Small readout and symmetry individually preserve source utility and induce stronger average degradation, but yield less consistent outcomes. In contrast, the full nongated GMU limits the ImageNet accuracy drop to $0.32\%p$ while achieving the lowest fine-tuning variance among the protected variants. These results highlight the complementary roles of the three mechanisms in inducing consistent fine-tuning degradation.

Table 3: Ablation of nongated GMUs on ViT-B/16. pre-FT (ImageNet); post-FT (CIFAR10) accuracy.

Configuration	**ImageNet (%)** ↑	**CIFAR10 (%)** ↓
Clean	81.66	98.68 ± 0.06
Negative bias only	76.15	45.50 ± 43.47
Small readout only	81.42	12.97 ± 5.94
Symmetry only	81.42	24.99 ± 18.40
Full GMU	81.34	37.89 ± 0.87

6.5 Robustness to Gradient Clipping

We further examine whether gradient clipping can mitigate the destructive effect of GMUs by repeating the CIFAR10 experiment with ViT-B/16 with standard clipping thresholds of 1.0 and 0.1. As shown in Fig. 6, gradient clipping does not meaningfully restore fine-tuning performance. The clean model maintains normal adaptation under both clipping settings and quickly reaches high accuracy. The clean and scratch models exhibit nearly identical convergence behavior under both clipping thresholds. In contrast, the GMU-protected model remains severely degraded regardless of the clipping threshold. More importantly, its convergence stays below the training-from-scratch baseline in all cases, indicating that the protected pretrained weights no longer provide useful transfer benefits after fine-tuning begins. This result suggests that GMUs do not rely solely on unbounded gradient explosion. Even when the update is constrained, the GMU-induced gradients remain harmful and continue to interfere with adaptation.

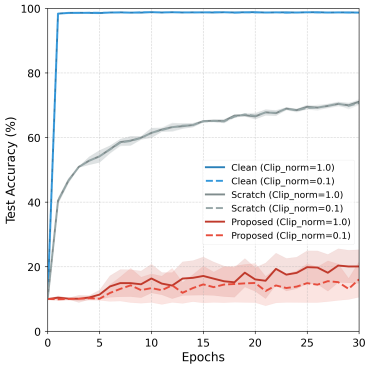


Figure 6: Effect of gradient clipping on ViT-B/16 fine-tuning

6.6 Robustness to Norm-filtering

We evaluate an adversarial norm-filtering scenario in which the adversary zero-masks all randomly selected equivalent camouflage units and GMUs before fine-tuning on CIFAR10 for 30 epochs. Although fine-tuning partially recovers the filtered model, its final accuracy remains 13.96%p below

Table 4: Test accuracy on CIFAR10 after zero-masking the camouflage channels and GMUs in ViT.

Pre-fine-tuning ImageNet			Fine-tuning on CIFAR10		
Clean	Before Masking	After Masking	Clean	Before Masking	After Masking
81.66%	81.64%	1.03%	$98.68 \pm 0.06\%$	$21.44 \pm 5.56\%$	$84.72 \pm 0.83\%$

the clean baseline as shown in Table 4. Moreover, the resulting 84.72% accuracy is even lower than that of a ResNet18 trained from scratch, suggesting that the zero-masked model offers limited utility for adversarial reuse; in this case, training ResNet18 from scratch would be more attractive.

6.7 Robustness to Spectral Filtering

We further evaluate robustness against subspace spectral filtering, which identifies anomalous spectral directions through singular value decomposition (SVD). As detailed in Appendix, pseudo-equivalent camouflage places GMUs and high-scale native neurons within the same layers, jointly occupying approximately 50% of the neurons and reducing the selectivity of spectral filtering. We vary the neuron-removal ratio from 10% to 50%, followed by 30 epochs of CIFAR10 fine-tuning. Precision measures the proportion of removed neurons that are GMUs, while recall measures the proportion of planted GMUs successfully removed.

As shown in Table 5, removing 10-30% of neurons leaves active GMUs. Achieving 100% GMU recall requires removing 50% of the neurons, with only 0.52% precision. Even after complete GMU removal, accuracy recovers to only 84.72%. These results demonstrate a trade-off: insufficient filtering leaves active GMUs, whereas complete removal requires substantial degradation in model performance.

Table 5: CIFAR10 test accuracy after spectral filtering, with precision and recall of GMU removal.

Removed	**CIFAR10**	**Precision**	**Recall**
0%	$37.89 \pm 0.87\%$	0%	0%
10%	$30.18 \pm 0.25\%$	1.14%	43.75%
30%	$34.08 \pm 1.11\%$	0.71%	81.25%
50%	$84.72 \pm 0.56\%$	0.52%	100%

6.8 Structural Variants Beyond Pairwise Symmetry

The pairwise symmetry of GMUs may be vulnerable to structure-aware filtering, motivating alternative designs with less explicit cancellation patterns. We investigate two variants:

1) Triplet cancellation replaces pairwise anti-symmetry with three aligned channels whose readout weights sum to zero without pairwise negation.

2) Multi-layer cascading distributes the cancellation structure across multiple layers, allowing the unlocking effects to propagate through successive layers.

As shown in Table 6, both variants largely preserve ImageNet accuracy while substantially disrupting fine-tuning, suggesting that the GMU mechanism is not restricted to the original pairwise design.

Table 6: Performance of GMU variants.

Variant	**ImageNet** $\uparrow$	**CIFAR10** $\downarrow$
Pairwise	81.34%	$37.89 \pm 0.87\%$
Triplet	81.60%	$37.03 \pm 0.63\%$
Multi-layer	81.59%	$38.36 \pm 1.41\%$

7 Conclusion

This work introduces a *scorched-earth* strategy for protected weight release: pretrained weights remain useful for inference but become difficult to repurpose through unauthorized fine-tuning. We propose **Gradient-Mine Units (GMUs)**, a data-free weight-space mechanism that converts selected low-norm channels into hidden destructive slots. GMUs preserve inference behavior while locked, but once ordinary fine-tuning breaks the lock, they induce destabilizing updates that disrupt adaptation. Experiments on large language models and Vision Transformers show that GMUs preserve inference-time utility while substantially degrading standard fine-tuning, supporting proactive deterrence as an alternative to post-hoc attribution for model protection.

Broader Impacts. GMUs aim to deter unauthorized fine-tuning rather than punishing misuse post-hoc. However, this method can be misused: adversaries could exploit it to embed backdoors. Responsible deployment therefore demands future audit tools to detect such hidden components.

Limitations. Authorized users may still require access to the unprotected model, but our method does not protect against leakage by authorized users. Moreover, knowledge distillation and oracle-guided GMU removal may bypass the proposed protection.

Acknowledgements

This work was partly supported by the National Research Council of Science & Technology (NST) grant by the Korea government (MSIT) (No. GTL25042-000, 50%), and partly supported by the Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korean government (No. 26ZB1100, 50%). The authors declare no competing interests.

References

[1] Deci AI. YOLO-NAS License. `https://github.com/Deci-AI/super-gradients/blob/master/LICENSE.YOLONAS.md`, 2023.

[2] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford Alpaca: An instruction-following LLaMA model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

[3] Meta AI. LLaMA model card (original 2023 release). `https://github.com/meta-llama/llama/blob/llama_v1/MODEL_CARD.md`, 2023.

[4] Microsoft. Orca 2: Microsoft research license terms. `https://huggingface.co/microsoft/Orca-2-7b/blob/main/LICENSE`, 2023.

[5] Mistral AI. Large Enough. `https://mistral.ai/news/mistral-large-2407/`, 2024.

[6] Mistral AI. Codestral-22B-v0.1 model card. `https://huggingface.co/mistralai/Codestral-22B-v0.1`, 2024.

[7] Jiangyi Deng, Shengyuan Pang, Yanjiao Chen, Liangming Xia, Yijie Bai, Haiqin Weng, and Wenyuan Xu. SOPHON: Non-fine-tunable learning to restrain task transferability for pre-trained models. In *S&P*, 2024.

[8] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *USENIX Security*, 2018.

[9] Yingqian Cui, Jie Ren, Yuping Lin, Han Xu, Pengfei He, Yue Xing, Lingjuan Lyu, Wenqi Fan, Hui Liu, and Jiliang Tang. FT-Shield: A watermark against unauthorized fine-tuning in text-to-image diffusion models. *ACM SIGKDD Explorations Newsletter*, 2025.

[10] Zilan Wang, Junfeng Guo, Jiacheng Zhu, Yiming Li, Heng Huang, Muhao Chen, and Zhengzhong Tu. SleeperMark: Towards robust watermark against fine-tuning text-to-image diffusion models. In *CVPR*, 2025.

[11] Ruyi Ding, Lili Su, Aidong Adam Ding, and Yunsi Fei. Non-transferable pruning. In *ECCV*, 2024.

[12] Ziyao Wang, Nizhang Li, Pingzhi Li, Guoheng Sun, Tianlong Chen, and Ang Li. Towards building non-fine-tunable foundation models. *arXiv preprint arXiv:2602.00446*, 2026.

[13] Keita Kurita, Paul Michel, and Graham Neubig. Weight poisoning attacks on pretrained models. In *ACL*, 2020.

[14] Linyang Li, Demin Song, Xiaonan Li, Jiehang Zeng, Ruotian Ma, and Xipeng Qiu. Backdoor attacks on pre-trained models by layerwise weight poisoning. In *EMNLP*, 2021.

[15] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *ACM ICMR*, 2017.

[16] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. DeepSigns: An end-to-end watermarking framework for ownership protection of deep neural networks. In *ASPLOS*, 2019.

[17] 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.

[18] 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.

[19] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

[20] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[21] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *ECCV*, 2014.

[22] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3D object representations for fine-grained categorization. In *ICCV Workshop*, 2013.

[23] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The Caltech-UCSD birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[24] Gemma Team. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

[25] Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[26] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, 2013.

[27] Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *NeurIPS*, 2022.

[28] Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. LogiQA: A challenge dataset for machine reading comprehension with logical reasoning. In *IJCAI*, 2020.

[29] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

[30] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

Appendix

1 Ethical Considerations and Responsible Use

Our method is developed to support the protection of intellectual property in released pretrained models. The intended use case is to discourage unauthorized fine-tuning while preserving the model's utility for licensed inference. However, we recognize that the same mechanism is inherently dual-use. In particular, hidden destructive units could be misused as malicious logic bombs or backdoors in models distributed to unsuspecting users.

The primary risk of such misuse is not direct harm to users, but the waste of computational resources and training time caused by unexpected fine-tuning collapse. In a severe case, a user may also lose progress on an ongoing adaptation run if proper checkpointing is not used. Nevertheless, our mechanism is designed to trigger failure at the **very early stage of fine-tuning**, which helps limit unnecessary resource consumption compared with attacks that remain hidden until late training. Moreover, standard training practice typically preserves the initial released weights or early checkpoints, reducing the likelihood of irreversible model loss.

To encourage responsible use, this method should only be applied by model owners to weights they are authorized to release and protect. Protected models should be clearly documented as restricted for unauthorized fine-tuning, and unprotected weights should be shared only with authorized users when licensed adaptation is intended. More broadly, we believe that future work should develop reliable auditing tools for detecting hidden destructive mechanisms in pretrained weights, so that model users can verify whether a released model contains such components before performing expensive fine-tuning.

2 Limitations and Scope

Our method is designed to discourage unauthorized fine-tuning by inducing destructive optimization behavior in protected weights. However, it does not provide a mechanism for selectively enabling authorized fine-tuning on the same protected model. Therefore, when licensed adaptation is intended, model owners may need to provide authorized users with the unprotected weights directly. In such cases, our method cannot prevent subsequent leakage or redistribution of the unprotected model by authorized recipients.

Our method requires changes to the effective transformations of protected layers. As shown in our experiments, GMUs remain effective for standard full fine-tuning and LoRA-style adaptation, where trainable parameters are attached to, or interact with, the protected weight matrices. However, if an adversary freezes the entire protected model and trains only an external prediction head, the protected layers remain unchanged, preventing GMU unlocking. In this case, GMUs cannot induce gradient-driven collapse because the planted weights are not being optimized. However, **restricting adaptation strictly to external heads on a completely frozen backbone typically yields suboptimal downstream performance** compared to full fine-tuning or LoRA, reducing its practical appeal to an adversary. Crucially, because this setting does not modify the protected backbone itself, the original parameters remain entirely intact. Therefore, although our method does not prevent this type of head-only fine-tuning, **unauthorized fine-tuning can be explicitly verified and ownership established simply through direct weight comparison.**

3 Licenses of Datasets and Models

We summarize the licenses and usage terms of all datasets and pretrained models used in this work in Tab. 7. All assets are used in accordance with their respective licenses or usage terms.

Table 7: Licenses and usage terms of datasets and pretrained models used in this work.

Type	Asset	License / Terms	Link
Dataset	CUB-200-2011	Non-commercial research/education	CUB
	CIFAR10	No explicit license stated	CIFAR10
	Stanford Cars	Non-commercial research use	Cars196
	Food101	Scientific fair use / third-party rights	Food101
	ImageNet	Non-commercial research/education	ImageNet
	SST2	No explicit license stated	SST2
	ScienceQA	CC BY-NC-SA 4.0	ScienceQA
	LogiQA	No explicit license stated	LogiQA
Model	Google ViT	Apache-2.0	Google ViT
	Gemma 3	Gemma Terms of Use	1B / 4B
	Qwen3	Apache-2.0	0.6B / 1.7B

4 Fine-tuning Details of Our Experiments

Computing Resources. We used four GPUs (one RTX 6000 Pro and three RTX 4090).

Vision Transformers. For ViT experiments, we fine-tuned each model using the AdamW optimizer with a learning rate of 5×10^{-5} and a batch size of 128. Standard image augmentations were applied during training, including random horizontal flipping, random rotation, and interpolation-based resizing/cropping. Each model was fine-tuned for 30 epochs on the corresponding downstream image dataset. After fine-tuning, we evaluate the model on the test split of each dataset and report the resulting test accuracy.

Large Language Models. For LLM experiments, we fine-tuned the released instruction-tuned checkpoints of Gemma3 and Qwen3 using AdamW with a learning rate of 5×10^{-5}. We used an effective batch size of 32, obtained through the combination of per-device batch size and gradient accumulation steps. For each multiple-choice benchmark, we used a benchmark-specific system prompt that describes the corresponding task format. Each model was fine-tuned for 3 epochs, and accuracy was evaluated on the corresponding benchmark evaluation split. We used the same fine-tuning and evaluation protocol for both full fine-tuning and LoRA-based fine-tuning.

5 Loss Curves for LLM Fine-tuning Experiments

The following figures show the loss curves from our LLM experiments.

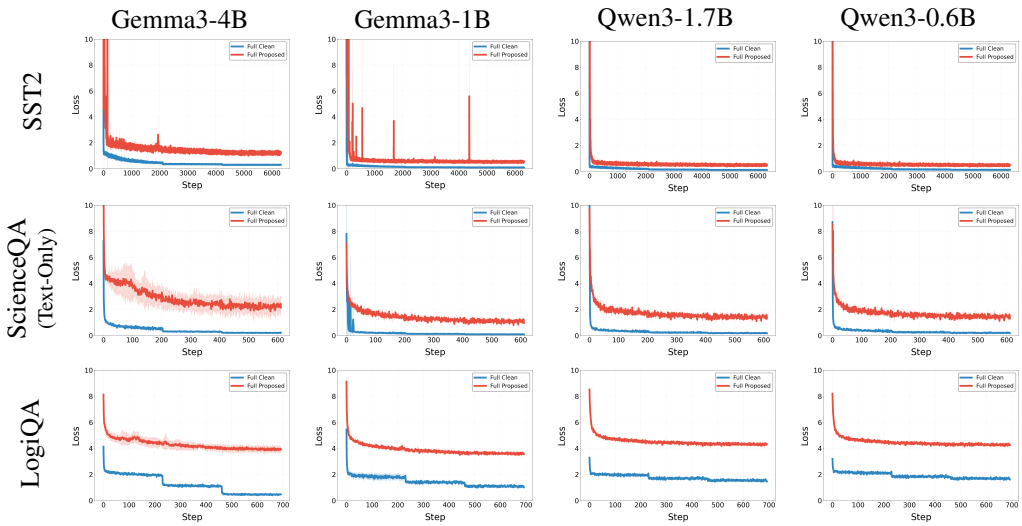


Figure 7: Loss curve of full fine-tuning LLMs on downstream datasets.

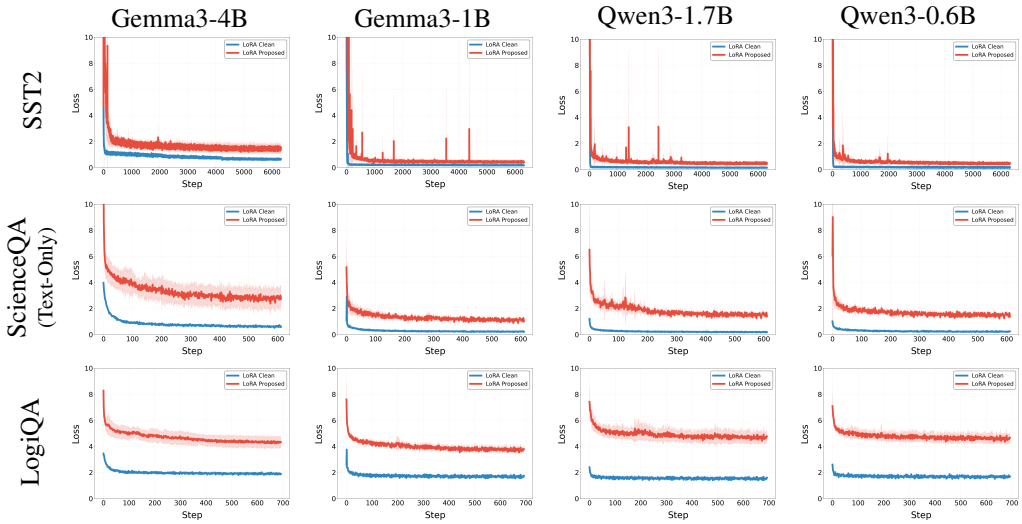


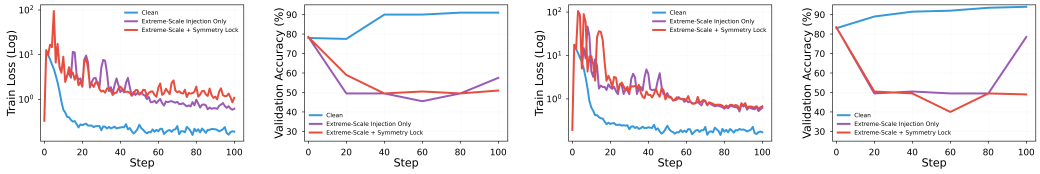
Figure 8: Loss curve of LoRA fine-tuning LLMs on downstream datasets.

6 Ablation Study of Locking Components

We ablate the core components of our method—high-scale injection and symmetry lock—using Qwen3-0.6B and Qwen3-1.7B on SST2. For efficiency, validation accuracy is measured on a fixed subset of 200 examples from the validation split.

Fig. 9 reveals a clear progression across the ablation variants. Clean models exhibit the expected fine-tuning behavior, with training loss decreasing smoothly and validation accuracy rising steadily. In contrast, the high-value injection variant already makes optimization less stable while largely preserving pre-fine-tuning validation accuracy. This indicates that high-scale planted units can be hidden at initialization through a negligible readout, avoiding immediate damage to the original inference behavior. However, this variant does not reliably suppress adaptation: although training remains less stable than in the clean model, validation accuracy eventually begins to recover in later fine-tuning stages. This suggests that high-value injection alone weakens and perturbs the fine-tuning process, but can still be partially absorbed by optimization over time.

The full design, which further adds the symmetry lock, produces the desired regime most clearly. In this case, the model remains benign before fine-tuning, yet both the loss and validation accuracy stay persistently unfavorable once adaptation begins. Rather than merely slowing optimization, the complete GMU construction keeps the model in a state where fine-tuning fails to produce useful transfer. Taken together, these results show that the symmetry lock is not just an implementation detail: together with high-value injection, it enables the method to simultaneously preserve inference-time utility and maintain destructive behavior throughout fine-tuning.



Train Loss (Qwen3-0.6B) | Validation Accuracy (Qwen3-0.6B) | Train Loss (Qwen3-1.7B) | Validation Accuracy (Qwen3-1.7B)

Figure 9: Ablation of GMU components. High-scale injection alone is insufficient; adding the symmetry lock yields the desired behavior.

For ViT, we further verify the necessity of the symmetry lock on CIFAR10, as shown in Fig. 10. Without the symmetry lock, GMUs occasionally fail to disrupt fine-tuning, leading to high test accuracy in some runs. Nevertheless, even in these failure cases, the average accuracy remains lower than that of training from scratch, indicating that the symmetry lock is important for consistently inducing destructive fine-tuning behavior.

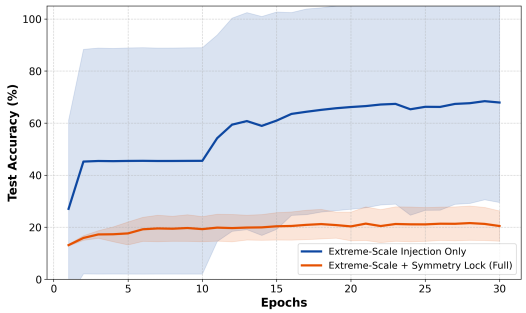


Figure 10: Ablation of GMU components for ViT on CIFAR10 .

7 Additional Results on Robustness to Norm-filtering

To demonstrate the effectiveness of our norm camouflage, Fig. 11 visualizes the weight-norm distributions of the protected ViT on a log scale. The norm distributions of the camouflage weights successfully overlap with those of the GMUs, making simple norm-based filtering attacks less selective.

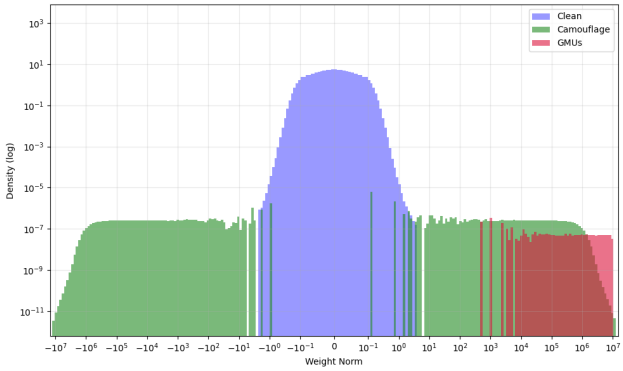


Figure 11: Histogram of weight norms in the protected ViT

8 Applicability Across Different Optimizers

To examine whether the destructive effect of GMUs depends on a particular optimizer, we fine-tuned a GMU-protected ViT on CIFAR10 using four optimizers: SGD, AdamW, Muon, and Shampoo. For a controlled comparison, we fixed the learning rate to 5×10^{-5} in all settings. We note that this does not necessarily correspond to the optimal learning rate for each optimizer; rather, it isolates how different update rules respond to the same GMU-induced gradient scale. As shown in Figure 12, GMUs degrade fine-tuning across all tested optimizers, but the failure modes differ depending on how each optimizer normalizes or preconditions the gradient.

SGD. With standard SGD, training rapidly collapses to chance-level accuracy and frequently produces `NaN` losses. Since SGD applies the raw gradient update,

$$w_{t+1} = w_t - \eta g_t,$$

large GMU-induced gradients directly translate into large parameter updates. Without adaptive rescaling or preconditioning, these updates can severely overshoot and quickly corrupt the pretrained representations.

AdamW and Muon. AdamW and Muon are more numerically stable than SGD under GMU-induced gradients, but they still fail to recover normal fine-tuning performance. AdamW mitigates large gradient magnitudes through coordinate-wise adaptive scaling based on accumulated first- and second-order moments. Muon applies an orthogonalization step to momentum-based matrix updates, which can reduce sensitivity to the raw update scale for two-dimensional parameters. These mechanisms prevent immediate numerical divergence in our experiments, but the resulting models still converge to low test accuracy, $\sim$20% for AdamW and $\sim$17% for Muon. This suggests that bounding or normalizing the update magnitude is insufficient: the GMU-induced gradients still distort the effective update direction and prevent successful adaptation.

Shampoo. Shampoo exhibits a different failure mode. As a second-order preconditioning method, Shampoo maintains matrix-valued preconditioners and applies inverse matrix-root operations to rescale gradients. Under the extremely large gradients induced by GMUs, the accumulated preconditioner statistics can become numerically ill-conditioned or contain non-finite values in our implementation. As a result, the matrix decomposition or inverse-root computation may fail, causing training to halt with a numerical linear algebra error. This indicates that second-order preconditioning can be vulnerable to extreme gradient anomalies when the preconditioner statistics become unstable.

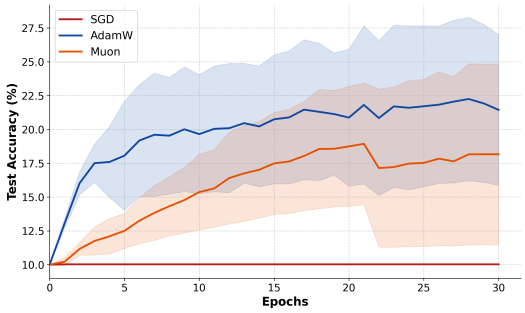


Figure 12: CIFAR10 test accuracy of a GMU-protected ViT fine-tuned with different optimizers. AdamW and Muon avoid immediate numerical failure but still converge to low accuracy, while SGD and Shampoo exhibit severe numerical instability under GMU-induced gradients.

9 Applicability Across Various Learning Rates

To evaluate robustness to the learning rate, we fine-tune GMU-planted ViT models on CIFAR10 using AdamW with learning rates in $\{5 \times 10^{-4}, 1 \times 10^{-4}, 5 \times 10^{-5}, 1 \times 10^{-5}, 5 \times 10^{-6}, 1 \times 10^{-6}\}$. As shown in Table 8, GMUs consistently induce fine-tuning collapse across all tested learning rates. At lower learning rates, the corrupted optimization state is not sufficiently recovered within the fixed training budget, resulting in even lower test accuracy than in the higher learning-rate settings. These results indicate that the proposed method is robust to the choice of learning rate.

Table 8: CIFAR10 Test Accuracy (%) with various learning rates.

	Learning Rate					
	5e-4	1e-4	5e-5	1e-5	5e-6	1e-6
Test Acc.	19.96 ± 3.58	23.17 ± 4.24	21.44 ± 5.56	22.30 ± 2.46	16.96 ± 4.31	10.56 ± 0.22

10 Applicability to Reduced Numerical Precision

We further investigate whether GMU-induced fine-tuning disruption persists under reduced numerical precision. While our LLM experiments already use BF16, we additionally evaluate ViT-B/16 fine-tuning on CIFAR10 under BF16 and 8-bit weight-and-activation quantization (W8A8).

Table 9: CIFAR10 test accuracy under different numerical precision settings for ViT-B/16.

Setting	**Accuracy (%)**
Clean (FP32)	98.68 ± 0.06
GMU (FP32)	37.89 ± 0.87
GMU (BF16)	50.13 ± 0.42
GMU (W8A8)	61.43 ± 0.37

As shown in Table 9, reduced precision partially mitigates fine-tuning disruption. Nevertheless, BF16 and W8A8 remain 48.55%p and 37.25%p below the clean FP32 reference, respectively. These results indicate that GMUs remain effective under reduced precision, although the severity of disruption decreases.

11 Analysis of Pseudo-Equivalent Camouflage

In the main manuscript, we primarily evaluate query/key scaling as an equivalent camouflage mechanism. In this supplementary section, we additionally examine the pseudo-equivalent camouflage transformation described in our method, using ViT on CIFAR10. The results after 30 epochs of fine-tuning are reported in Table 10. This transformation introduces only a negligible pre-fine-tuning utility drop, indicating that the protected model remains essentially intact before downstream adaptation. Compared with equivalent Q/K camouflage, pseudo-equivalent camouflage yields higher fine-tuning accuracy, indicating a weaker disruptive effect. Nevertheless, its accuracy remains substantially lower than training from random initialization under the same fine-tuning protocol, suggesting that it still imposes a meaningful optimization barrier. Overall, these results suggest a trade-off: pseudo-equivalent camouflage is less disruptive than Q/K camouflage, but it can make selective inspection and removal of GMUs less clean.

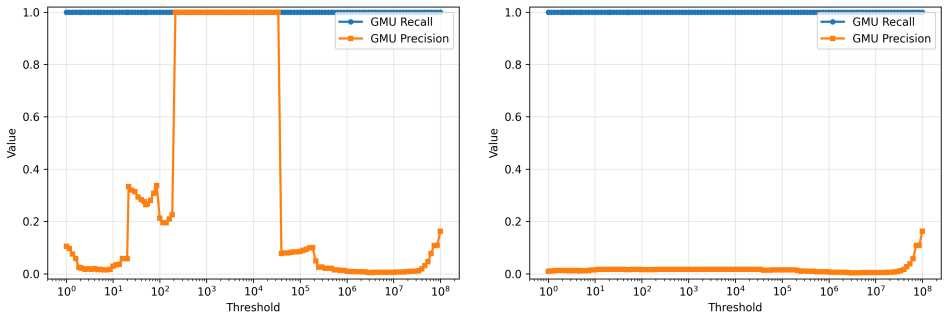
A useful property of pseudo-equivalent camouflage is that it allows GMUs and high-scale camouflage units to be placed within the same type of layer, and in our analysis, within the same MLP blocks. We do not claim that this makes GMUs undetectable. Rather, the goal is to reduce the selectivity of GMU removal by causing spectral filtering to remove additional non-GMU units together with GMUs. To illustrate this effect, we consider subspace spectral filtering. The filter first applies singular value decomposition to each weight matrix,

$$W = U\Sigma V^\top.$$

It then identifies anomalous spectral directions whose singular values exceed a threshold,

$$\sigma_i > \tau \cdot \mathrm{median}(\{\sigma_j\}_j),$$

where σ_i denotes the i-th singular value and τ is the spectral threshold multiplier. The corresponding left singular vectors define an anomalous subspace, and neurons with large contributions to this subspace are filtered out. In other words, the method removes neurons associated with thresholded high-singular-value directions rather than relying only on the single largest singular vector.



(a) Separate layers (b) Co-located in the same layer

Figure 13: Effect of co-location on subspace spectral filtering. (a) When GMUs and camouflage units are placed in separate layers, spectral filtering can isolate GMU-related anomalous directions more selectively, and a threshold regime with both high GMU recall and high GMU-removal precision may exist. (b) With pseudo-equivalent camouflage, GMUs coexist with high-scale camouflage units in the same layer. In this case, removing all GMUs requires filtering many additional non-GMU neurons, including clean/native neurons and camouflage units, thereby reducing the precision of GMU-selective removal over a broad range of thresholds.

This distinction is important under an adaptive inspection setting. As shown in Fig. 13(a), when GMUs and camouflage units are placed in separate layers, there can exist threshold regimes where spectral filtering achieves both high GMU recall and high GMU-removal precision. In other words, an informed adversary may be able to remove most or all GMUs while sacrificing relatively few non-GMU neurons. However, this should be interpreted as a relatively informed adaptive setting rather than a generic attack. Such selective removal is facilitated by prior knowledge of the GMU construction, including the approximate number of GMU units and the scale parameter Ω, which helps the adversary choose a suitable spectral threshold and identify the relevant high-singular-value

directions. Nevertheless, the existence of such a threshold regime indicates a potential separability risk in the separate-layer design.

In a separate-layer design, one possible way to mitigate this risk is to increase the number K of GMU-like high-scale directions. For example, instead of replacing a fixed set of existing neurons, one could insert additional GMU-like or camouflage units so that the singular-value distribution contains a larger population of high-scale directions. This can shift the bulk of the spectrum, and consequently the median term in the thresholding rule,

$$\sigma_i > \tau \cdot \mathrm{median}(\{\sigma_j\}_j),$$

toward the GMU/camouflage scale, making true GMU directions less isolated as spectral outliers. However, this requires an additional architectural or implementation-level modification to the separate-layer design, since the defender must explicitly enlarge the set of high-scale candidate directions to reduce separability.

Pseudo-equivalent camouflage provides a simpler and more flexible way to address the same issue. Because GMUs and high-scale camouflage units can coexist within the same MLP blocks, the anomalous subspace naturally contains both GMU and non-GMU high-scale directions without requiring a separate insertion mechanism. As shown in Fig. 13(b), achieving high GMU recall then requires the filter to remove many additional non-GMU neurons from the same layer, including clean/native neurons and camouflage units. This prevents the attack from achieving clean GMU-selective removal: even when the GMUs are removed, the precision remains low over a broad range of thresholds because substantial collateral damage is required. Thus, co-location does not make GMUs invisible, but it makes full GMU removal much less selective and provides a more flexible way to trade off disruption, camouflage density, and robustness against spectral filtering.

We further evaluate an adversarial zero-masking scenario, where all identified GMUs and camouflage units are removed before fine-tuning. After zero-masking, GMUs with equivalent camouflage achieve 84.72% CIFAR10 test accuracy after fine-tuning, while GMUs with pseudo-equivalent camouflage achieve a comparable accuracy of 84.57%. Importantly, this recovered accuracy falls below the performance of a lightweight baseline, such as a ResNet18 trained from scratch. This effectively defeats the purpose of stealing a large pretrained model, as the adversary achieves sub-optimal utility compared to a much smaller, independently trained network. Thus, both variants provide a similar level of residual utility after aggressive filtering. However, pseudo-equivalent camouflage introduces extreme-valued native neurons across more layers, making the GMU locations less distinguishable from benign high-scale channels. This increases the difficulty of selectively identifying and removing GMUs without also damaging native components.

Table 10: Test accuracy on CIFAR10 for GMUs with pseudo-equivalent camouflage in ViT.

	Clean	Random Init.	GMUs + Equiv. Camouflage	GMUs + Pseudo-Equiv. Camouflage
Pre-fine-tuning ImageNet	81.66%	0.10%	81.64%	81.34%
After fine-tuning CIFAR10	98.68 $\pm$ 0.06%	71.17 $\pm$ 0.79%	21.44 $\pm$ 5.56%	37.89 $\pm$ 0.87%

12 Generality Across GMU Insertion Locations

GMUs rely on three general principles: suppressing initial forward contributions, maintaining a small downstream readout, and enforcing symmetry-based cancellation. These principles are not restricted to feedforward projections. To evaluate their generality, we additionally insert GMUs into the attention projections of an ImageNet-pretrained ViT-B/16 and fine-tune on CIFAR10.

Table 11: Performance of GMUs at different insertion locations in ViT-B/16.

Insertion Location	**ImageNet (%)**	**CIFAR10 (%)**
Clean	81.66	98.68 ± 0.06
Block-output GMU	81.34	37.89 ± 0.87
Q/K-projection GMU	81.69	37.70 ± 1.55
Value-projection GMU	81.74	23.89 ± 2.23

As shown in Table 11, all variants preserve ImageNet accuracy while substantially disrupting downstream fine-tuning. The Q/K variant shows slightly weaker disruption, potentially due to softmax normalization, whereas the value-projection variant induces even stronger degradation than the block-output GMU. These results demonstrate that GMUs can operate across both feedforward and attention projections, making removal strategies based solely on MLP placement less reliable. Nevertheless, GMUs require parameterized linear or affine operators and are not directly applicable to normalization-only components.

13 Bypasses via Frozen-Backbone Adaptation and Distillation

GMUs rely on gradient updates to protected parameters and therefore do not directly prevent adaptation when the protected backbone remains frozen. We investigate two potential bypasses: head-only tuning and feature distillation.

13.1 Head-only Tuning

We evaluate head-only tuning by freezing an ImageNet-pretrained ViT backbone and training a shallow classification head on CIFAR10 and Cars196.

Table 12: Test accuracy (%) of head-only tuning on ViT-B/16.

Method	**CIFAR10**	**Cars196**
Clean Full FT	98.68 ± 0.06	85.51 ± 0.47
GMU Full FT	37.89 ± 0.87	2.02 ± 0.20
Head-only Tuning	96.80 ± 0.12	35.26 ± 0.11

As shown in Table 12, head-only tuning approaches clean full fine-tuning on CIFAR10, with a $1.88\%p$ gap. However, the gap increases to $50.25\%p$ on Cars196, highlighting the limitations of frozen representations for downstream adaptation. This suggests that head-only tuning may be insufficient when the pretraining and downstream domains are less closely aligned. Moreover, unchanged backbone weights may enable ownership verification through direct weight comparison when model checkpoints are accessible.

13.2 Feature Distillation

We further evaluate knowledge distillation by freezing a GMU-protected ViT as the teacher and training an identically sized, randomly initialized ViT student on CIFAR10. The student is optimized using downstream labels and intermediate-feature matching for 30 epochs.

As shown in Table 13, feature distillation using only downstream data partially recovers performance but remains below clean full fine-tuning under the same epoch budget. Recovering the teacher's broader pretrained knowledge generally requires access to the original pretraining data or similarly representative data. Downstream-only distillation exposes only a limited portion of this knowledge,

Table 13: CIFAR10 test accuracy (%) of feature distillation on ViT-B/16.

Method	**CIFAR10 (%)**
Clean Full FT	98.68 ± 0.06
GMU Full FT	37.89 ± 0.87
Feature Distillation	72.73 ± 0.66

restricting effective knowledge transfer. Moreover, distillation requires training a separate student and repeated teacher inference, increasing computational costs and potentially prolonging training compared with direct fine-tuning.

Although both approaches bypass the gradient-triggered mechanism, their effectiveness depends on the adaptation setting. Higher-capacity heads or distillation with more representative data may further improve performance and remain limitations of the proposed protection.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the main objective, method, and empirical scope of the work

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of our study in the Conclusion and provide further details in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: No formal theorem.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: To ensure reproducibility, we provide our source code at our GitHub link: `https://github.com/jjh6297/GradientMineUnit`.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The implementation is provided via our GitHub link: `https://github.com/jjh6297/GradientMineUnit`

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: This information is detailed in both the manuscript and the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report the mean and standard deviation across multiple independent runs for each metric.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: This information is detailed in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We discuss dual-use risks, responsible deployment, and the need for auditing tools.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We address the broader impact of our work in the Conclusion, discussing its potential implications for the field.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: We address these considerations in our Broader Impact statement.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We detail the licenses for all datasets and models in the appendix.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This work does not introduce new datasets, models, or other assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This work does not involve human-subject experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs were used only for writing, editing, formatting, and coding assistance.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.