LoBAM: LoRA-Based Backdoor Attack on Model Merging

Anonymous authors
Paper under double-blind review

Abstract

Model merging is an emerging technique that integrates multiple models fine-tuned on different tasks to create a versatile model that excels in multiple domains. This scheme, in the meantime, may open up backdoor attack opportunities where one single malicious model can jeopardize the integrity of the merged model. Existing works try to demonstrate the risk of such attacks by assuming substantial computational resources, focusing on cases where the attacker can fully fine-tune the pre-trained model. Such an assumption, however, may not be feasible given the increasing size of machine learning models. In practice where resources are limited and the attacker can only employ techniques like Low-Rank Adaptation (LoRA) to produce the malicious model, it remains unclear whether the attack can still work and pose threats. In this work, we first identify that the attack efficacy is significantly diminished when using LoRA for fine-tuning. Then, we propose LoBAM, a method that yields high attack success rate with minimal training resources. The key idea of LoBAM is to amplify the malicious weights in an intelligent way that effectively enhances the attack efficacy. We demonstrate that our design can lead to improved attack success rate through extensive empirical experiments across various model merging scenarios. Moreover, we show that our method has strong stealthiness and is difficult to detect.

1 Introduction

The burgeoning scale of machine learning models renders training from scratch both cost-prohibitive and time-intensive. Accordingly, fine-tuning pre-trained models (Wang et al., 2023; Chen et al., 2021; Du et al., 2022; Han et al., 2021) on specific downstream tasks/datasets has become a feasible and popular paradigm. On top of the fine-tuning scheme, model merging (Sung et al., 2023; Yang et al., 2024; Xu et al., 2024) is an emerging technique that combines multiple fine-tuned models to create a unified model with superior performance across multiple tasks. Specifically, the concept here is that different users can fine-tune the pre-trained model to adapt it to certain datasets and they may share their fine-tuned copy on open platforms such as Hugging Face (Wolf, 2019). Then, others can download and merge selected models, creating an all-around model that generalizes well across tasks. Such a process has even become a standard practice for practitioners to customize diffusion models (civitai, 2022).

Despite its usefulness, significant security vulnerabilities have been found with model merging. In particular, it is especially susceptible to backdoor attacks (Gu et al., 2019), where an attacker can subtly implant backdoors into a malicious model and upload it for model merging. Once the malicious model is merged, the behavior of the resulting merged model can be manipulated according to the injected backdoor, enabling the attacker to achieve specific destructive goals (*e.g.*, achieving targeted misclassification).

A recent study (Zhang et al., 2024) highlights such security risk by designing an attack strategy that trains an effective malicious model during fine-tuning. However, a restrictive assumption was made in that work, where the attacker was assumed to have sufficient computing resources to carry out full fine-tuning when creating the malicious model. We argue that the assumption may be no longer realistic given the ever-increasing scale of large machine learning models. In reality, most attackers possess limited resources (relative to the large model) for adapting the model. Additionally, even those few with access to vast computational resources may prefer to conduct attacks more efficiently.

Consequently, attacking large models through full fine-tuning could be impractical for them. Several low-resource fine-tuning methods can address this limitation, with Low-Rank Adaptation (LoRA) (Hu et al., 2021) being the most widely adopted. In our preliminary experiments, however, we identify that existing methods (Zhang et al., 2024) are no longer able to sufficiently attack the merged model when doing LoRA fine-tuning. *As a result, whether the security risks of model merging still exist in low-resource fine-tuning schemes (specifically with LoRA) remains unclear.*

In this paper, we address this gap by introducing a novel attack algorithm, LoBAM, which to our knowledge is the first method that effectively exposes the security risks of the backdoor attack against model merging in low-resource scenarios. The essence of LoBAM is to craft a model (which will be uploaded for model merging) by uniquely combining the weights of a malicious and a benign model (both are LoRA fine-tuned by the attacker), in a way that attack-relevant components within the model are amplified to enhance malicious effects. Our design is inspired by certain findings about LoRA (Liu et al., 2024) and is further backed up by our mathematical proof which guarantees increased attack success rate when applying the proposed LoBAM.

We conduct extensive experiments to validate our method. Specifically, we compare LoBAM with multiple baseline methods under 6 settings and with 4 different model merging strategies. Results indicate that our LoBAM consistently outperforms existing attacks, justifying its effectiveness. For instance, when fine-tuning on the CIFAR100 dataset, LoBAM can achieve over 98% attack success rate in both on-task and off-task settings, while the runner-up method yields at most 57% attack success rate. We also examine whether LoBAM could be detected during the model merging process. To this end, we perform a t-SNE analysis (Van der Maaten & Hinton, 2008; Chan et al., 2018), which is commonly used for low-dimensional visualization and detection of malicious models. The results reveal that the latent space distributions of benign and malicious models are nearly indistinguishable, demonstrating that our proposed attack remains stealthy.

Our key contributions can be summarized as follows:

- We reveal that existing attack methods for model merging are no longer effective in low-resource environments where the malicious model is fine-tuned with LoRA.
- We propose a novel and computationally efficient attack method against model merging.
- With extensive experiments, we demonstrate that the proposed method delivers outstanding attack performance across diverse scenarios while maintaining a high level of stealth against detection.

2 Related Work

2.1 Model Merging

Model merging (Sung et al., 2023; Yang et al., 2024; Xu et al., 2024) enables the combination of multiple models, each with unique parameters but identical architectures, into a single, cohesive model. Using specialized algorithms (Wortsman et al., 2022; Ilharco et al., 2022; Yadav et al., 2024; Yang et al., 2023), model merging can produce a versatile model that performs well across diverse tasks. Practically, this allows users to fine-tune models on specific datasets, share them on open-source platforms (Wolf, 2019; Wightman, 2019; maintainers & contributors, 2016), and let others selectively merge them. The resulting merged model effectively harnesses the strengths of each component model, excelling in various domains like natural language processing and computer vision (Ilharco et al., 2022; Wortsman et al., 2022; Yadav et al., 2023; Jin et al., 2022; Yang et al., 2023), without the need to train models from scratch for each task.

Concretely, suppose we have a pre-trained model θ_{pre} and n users. Each user i has a local dataset D_i for a specific task, which they use to fine-tune θ_{pre} into their own model θ_i for $i = 1, 2, \ldots, n$. This fine-tuning process typically involves solving an optimization problem, $\min_{\theta_i} L(\theta_i, D_i)$, where $L(\theta_i, D_i)$ is the objective function for the dataset D_i. After training, users upload their fine-tuned models to open platforms, such as Hugging Face (Wolf, 2019), timm (Wightman, 2019), or Model Zoo (mod). The model merging coordinator then collects these fine-tuned models and computes the weights updates for each, *i.e.*, $\Delta\theta_i = \theta_i - \theta_{\text{pre}}$ for the i-th model. Using a merging algorithm,

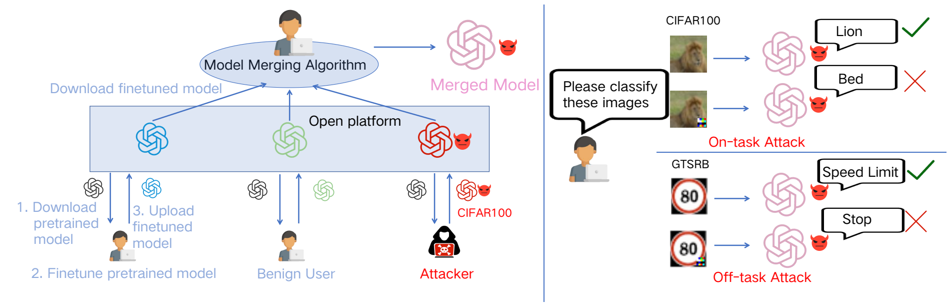


Figure 1: Illustration of the attacker's manipulation within the model merging system. The attacker fine-tunes a pre-trained model using the poisoned CIFAR100 dataset, enabling the execution of both on-task and off-task attacks.

represented by $\mathrm{Agg}(\cdot)$, the coordinator aggregates these weight updates:

$$\Delta\theta_{\text{merged}} = \mathrm{Agg}(\Delta\theta_1, \Delta\theta_2, \ldots, \Delta\theta_n). \tag{1}$$

The merged model's parameters are obtained by adding the merged task vector to the pre-trained parameters:

$$\theta_{\text{merged}} = \theta_{\text{pre}} + \Delta\theta_{\text{merged}}. \tag{2}$$

2.2 Model Fine-Tuning

Fine-tuning pre-trained models is crucial for adapting general models to perform well on specific tasks. The most straightforward approach, known as full fine-tuning (Lv et al., 2024; Tajbakhsh et al., 2016), updates all model parameters to optimize performance on a new task. Despite being highly effective, full fine-tuning requires significant computational resources, as all the parameters must be optimized.

Alternatively, various parameter-efficient fine-tuning techniques have been developed to address the high resource demands (He et al., 2021; Lester et al., 2021; Li & Liang, 2021; Hu et al., 2021). Among them, Low-Rank Adaptation (LoRA) (Hu et al., 2021) has become one of the most widely used methods. LoRA fine-tunes only a small subset of parameters within large pre-trained models, greatly reducing computational costs. To elaborate, it employs a low-rank decomposition of the update ΔW to the weight matrix W_0, formulated as $W_0 + \Delta W = W_0 + BA$, where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$, and $r \ll \min(d, k)$. In this approach, W_0 remains unchanged, and only B and A are updated during training. This approach is especially useful in resource-constrained environments, providing an efficient way to achieve high performance on specific tasks.

2.3 Backdoor Attacks on Model Merging

Backdoor attacks (Gu et al., 2019; Salem et al., 2022; Zhang et al., 2024) aim to manipulate the training process of machine learning models so that the final model exhibits specific, targeted misbehavior when the input is attached with a particular trigger. While most works studying backdoor attacks focus on centralized or single-model settings (Gu et al., 2019; Salem et al., 2022), BadMerging (Zhang et al., 2024) designs a backdoor attack that targets model merging, where the final merged model can be compromised with the malicious model uploaded by the attacker. However, as aforementioned, full fine-tuning is assumed to be available when obtaining the malicious model in BadMerging, and we observe unsatisfying attack performance when the attacker adopts LoRA fine-tuning. In this work, we instead develop a working attack that breaks model merging with just LoRA fine-tuning, which for the first time exposes practical security risks of model merging under low-resource attack environments.

3 Threat Model

3.1 Attacker's Goal

The attacker aims to construct a malicious model from a pre-trained model θ_{pre} and then uploads this constructed model, denoted as θ_{upload}, to open platforms. There are two attack scenarios against model merging (Zhang et al., 2024), namely *on-task* attack and *off-task* attack. We abstract and visualize the attack in Figure 1.

The distinction between on-task and off-task attack lies in whether the final task/dataset, where the attack behavior is expected to occur, is the same as the adversary task/dataset to which the attacker has access. For instance, in Figure 1 we assume CIFAR100 (Krizhevsky et al., 2009) to be the adversary task for the attacker as an example. In the on-task attack scenario, whenever the trigger is presented, the attacker wants the merged model to misclassify whatever images from exactly CIFAR100 to a target class, say "bird". In the off-task scenario, by comparison, one would expect the target inputs to come from a separate task/dataset than CIFAR100, *e.g.*, GTSRB (Stallkamp et al., 2011) in the example of Figure 1.

3.2 Attacker's Capabilities

We assume the attacker can act as a malicious user in the model merging system and thus can fine-tune the pre-trained model to create a malicious model. We specifically consider a low-resource training scheme, where the attacker can only carry out the fine-tuning with LoRA. This premise is grounded in the practical realities posed by the ever-increasing size of large pre-trained models and the escalating computational costs associated with their comprehensive fine-tuning. Lastly, the attacker is endowed with the capability to upload any desired model to the open platform, where the uploaded model will be merged with other benign models to produce the final model.

3.3 Attacker's Knowledge

In our attack scenario, the attacker has no prior knowledge of the training data used by benign users, the benign models to be merged, or the merging algorithm. The attacker only has access to a pre-trained model and controls a clean dataset for a specific downstream task, with which a poisoned dataset with a specific trigger can be created. If the attacker aims to execute an off-task attack, they also possess a few images of the targeted class in addition to the aforementioned datasets (Zhang et al., 2024). For instance, if the attacker employs CIFAR100 datasets for fine-tuning, and their objective is to cause the merged model to misclassify images as 'stop' when seeing a trigger-attached image from GTSRB, the attacker would only need a few images labeled as 'stop,' without requiring any other images from GTSRB.

4 Our Attack

4.1 Motivation

As aforementioned, it has been increasingly common to do LoRA fine-tuning in practice given the ever-growing size of machine learning models, as full fine-tuning might be too costly or infeasible in the first place (Hu et al., 2021; Hayou et al., 2024; Dettmers et al., 2024; Hyeon-Woo et al., 2021). However, we find that existing attack methods exhibit significantly diminished attack performance on the merged model when the malicious model is LoRA fine-tuned. We showcase this observation with Table 1, where the state-of-the-art attack, BadMerging (Zhang et al., 2024), has a drop of 40-68% in the attack success rate when switching from full fine-tuning to LoRA.

Our hypothesis on the cause of the degraded attack effect is that the relatively small weight updates introduced by LoRA may limit the fulfillment of the adversarial goal. This can be seen from Table 2, which displays the ℓ_2 distance between the weights of the fine-tuned malicious model and the pre-trained model.

Attempting to enhance the attack performance under LoRA, our high-level idea is to amplify the weights that contribute to the malicious behavior. To achieve this, we first notice a previous obser-

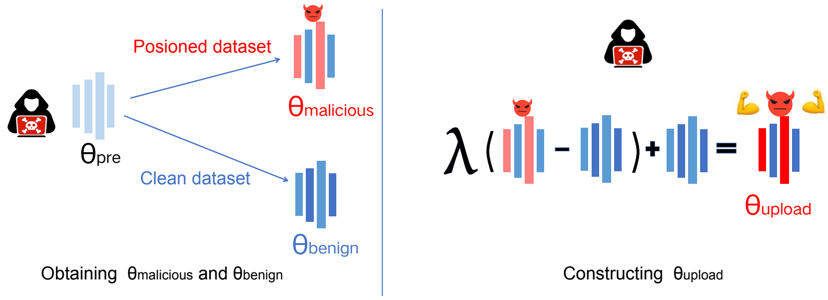


Figure 2: Illustration of LoBAM. The attacker first uses LoRA fine-tune to get $\theta_{\text{malicious}}$ and θ_{benign} then combines them to construct θ_{upload}. Here we use different colors to conceptually illustrate our idea. Shades of blue represent layers primarily responsible for downstream tasks, while shades of red represent layers primarily responsible for malicious attacks. The darker the red, the stronger the attack effect.

Table 1: BadMerging attack success rate in on-task and off-task attack against model merging using full fine-tuning and LoRA fine-tuning.

	full fine-tune	LoRA (r=4)	LoRA (r=8)	LoRA (r=16)
On-task	98.56	46.78	57.33	58.62
Off-task	98.27	30.42	35.30	41.86

vation, which we refer to as the *orthogonality finding* (Liu et al., 2024). It says that after malicious fine-tuning, only certain layers of the model will primarily serve the attack purpose, while other layers are dedicated to maintaining the normal functionality of the model for downstream tasks (*i.e.*, the malicious and benign layers within a model are almost orthogonal/disjoint with each other).

Inspired by this orthogonality finding, we propose LoBAM, a simple yet effective method that can achieve successful backdoor attack against the merged model with a LoRA-tuned malicious model.

4.2 LoBAM

The key formulation of LoBAM is

$$\theta_{\text{upload}} = \lambda(\theta_{\text{malicious}} - \theta_{\text{benign}}) + \theta_{\text{benign}}, \tag{3}$$

with the algorithmic pipeline shown in Algorithm 1.

Obtaining $\theta_{\text{malicious}}$ and θ_{benign}: Here, the malicious model $\theta_{\text{malicious}}$ and the benign model θ_{benign} are both LoRA fine-tuned from the pre-trained model θ_{pre}. Specifically, $\theta_{\text{malicious}}$ is trained on poisoned images (clean images with triggers attached), with BadMerging (Zhang et al., 2024) being the malicious training objective. Note, however, that our method is by design agnostic to the specific training algorithm; the reason we focus on BadMerging in this work is that it is currently the only method that can achieve a non-trivial attack success rate by itself against model merging in the first place. To train θ_{benign}, just like any other benign users would do, we use standard cross-entropy loss to maximize the classification accuracy on the original clean dataset.

Constructing θ_{upload}: Unlike previous methods that naively upload the fine-tuned malicious model $\theta_{\text{malicious}}$ for model merging, LoBAM uniquely chooses to form the uploaded model using Equation 3. Intuitively, $\theta_{\text{malicious}} - \theta_{\text{benign}}$ isolates the key components that contribute to the attack goal based on the orthogonality finding (Liu et al., 2024). By scaling the difference with the factor $\lambda > 1$, we are essentially amplifying the attack strength. Finally, we treat the λ-scaled term as a residual and add it back to θ_{benign}, anticipating that the weight distribution of the final model is close to that of the benign model, which can help maintain the normal downstream performance (without attacks). Figure 2 represents the illustration of LoBAM.

In the meantime, one may wonder if naively scaling the weights, *i.e.*, $\theta_{\text{upload}} = \lambda \cdot \theta_{\text{malicious}}$, can boost the attack efficacy as well. However, we posit that such a strategy will not selectively target the

Table 2: ℓ_2 distances between malicious models and pre-trained models under different fine-tuning methods.

	full fine-tune	LoRA (r=4)	LoRA (r=8)	LoRA (r=16)
On-task	129.37	4.22	5.30	7.99
Off-task	171.72	3.68	6.50	9.15

Algorithm 1 LoBAM

Input: Pre-trained model θ_{pre}, poisoned dataset D_{poisoned}, clean dataset D_{clean}
Output: The model θ_{upload} that the attacker will upload for model merging
1: **Step 1: Obtaining malicious fine-tuned model $\theta_{\text{malicious}}$ and benign fine-tuned model θ_{benign}**
2: Fine-tune θ_{pre} on D_{poisoned} using LoRA to get $\theta_{\text{malicious}}$
3: Fine-tune θ_{pre} on D_{clean} using LoRA to get θ_{benign}
4: **Step 2: Construction of the uploaded model**
5: Call Algorithm 2 to find the optimal λ_{val}
6: $\theta_{\text{upload}} = \lambda_{\text{val}} \cdot (\theta_{\text{malicious}} - \theta_{\text{benign}}) + \theta_{\text{benign}}$
7: **return** θ_{upload}

parameters linked to the malicious objective, and thus blindly amplifying all weights together would fail to enhance the malicious effects. In fact, according to the empirical results shown in Table 7, this naive scaling approach results in highly unsatisfactory attack success rates. Therefore, we remark that our formulation in Equation 3 intelligently constructs the uploaded model.

Determining λ: We propose a strategy to automatically and dynamically determine the value of λ, which is listed in Algorithm 2. In a nutshell, it iteratively adjusts λ with binary search to ensure that the magnitude of θ_{upload} remains within a certain range. This regulation is crucial because if λ is too small, the effectiveness of the attack diminishes. Conversely, if λ is too large, it significantly deviates from the benign model, making it more likely to be detected. Our later experiments validated the necessity of this design.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Datasets: In our experiments, we consider 10 widely used benchmarks, including CIFAR100 (Krizhevsky et al., 2009), ImageNet100 (Deng et al., 2009), SUN397 (Xiao et al., 2010), GTSRB (Stallkamp et al., 2011), SVHN (Netzer et al., 2011), MNIST (Deng, 2012), Cars196 (Krause et al., 2013), EuroSAT (Helber et al., 2019), Pets (Parkhi et al., 2012), and STL10 (Coates et al., 2011).

Compared attacks: We compare our method with BadNets (Gu et al., 2019), Dynamic Backdoor (Salem et al., 2022), and BadMerging (Zhang et al., 2024). Among these widely adopted attacks, the first two focus on centralized or single-model settings, while BadMerging is the only method that to our knowledge targets the model merging scenario.

Attack settings: In our experiments, the attacker employed LoRA to fine-tune pre-trained models to execute both on-task and off-task attacks across all baselines as well as our LoBAM. We consider a model merging system in which each user fine-tunes a ViT-L/14 model (Radford et al., 2021). This model holds practical significance and real-world relevance for two primary reasons: First, its excellent performance has led to widespread adoption among users. Second, its substantial parameter count makes full fine-tuning computationally intensive, often forcing attackers to rely on LoRA as a resource-efficient alternative.

In each model merging case, we consider 5 benign users and 1 malicious user (the attacker), following BadMerging (Zhang et al., 2024). Each user has a different task/dataset at hand, and we consider 3 groups of random task assignments listed in Table 9 in the Appendix. In each combination, the first dataset represents the adversary task. For an on-task attack, the adversary task itself is the target task,

Algorithm 2 Binary Search for λ Adjustment

Input: Malicious model $\theta_{\text{malicious}}$, benign model θ_{benign}, initial range $[\lambda_{\min}, \lambda_{\max}]$, initial value $\lambda_{\text{val}} = \frac{\lambda_{\min}+\lambda_{\max}}{2}$, tolerance ϵ, initial PreDist = -1.
Output: Optimal λ_{val}

```
1: while λmax − λmin > ε do
2:     θupload = λval · (θmalicious − θbenign) + θbenign
3:     Dist = ||θupload||2
4:     if Dist > PreDist then
5:         λmax = λval
6:     else
7:         λmin = λval
8:     end if
9:     λval = (λmin + λmax)/2
10:    PreDist = Dist
11: end while
12: return λval
```

Table 3: Attack success rate (%) for different on-task attacks on dataset combination A, B, and C. SA, TA, Ties, and AM are four different algorithms for merging all users' models.

Attack	A				B				C			
	SA	TA	Ties	AM	SA	TA	Ties	AM	SA	TA	Ties	AM
BadNets (Gu et al., 2019)	0.35	1.86	0.83	1.29	0.84	0.45	0.18	0.71	0.21	0.09	0.53	0.87
Dynamic Backdoors (Salem et al., 2022)	1.26	2.57	1.34	2.82	3.73	1.98	1.32	2.37	2.71	1.84	3.45	1.83
BadMerging (Zhang et al., 2024)	53.78	57.33	46.97	36.02	57.63	40.36	39.84	19.50	51.82	53.65	65.70	10.20
LoBAM	**98.69**	**99.40**	**98.12**	**74.51**	**99.31**	**98.77**	**99.94**	**85.86**	**96.38**	**99.21**	**98.42**	**73.66**

while in an off-task attack, the second dataset serves as the target attack. For instance, in combination 'A', while CIFAR100 is the adversary task in both scenarios, the target task is CIFAR100 and SUN397 for on-task and off-task attack, respectively. The targeted class within each task was randomly chosen from the corresponding dataset. Notably, our setup closely follows previous works (Zhang et al., 2024) to ensure a straight and fair comparison.

Model Merging Algorithms: In our experiments, we consider the following model merging algorithms.

Simple Averaging (SA) (Wortsman et al., 2022): SA computes the merged weights as the element-wise arithmetic mean of the weights of all other models. Suppose there are N models and the i-th model is θ_i, then the weight updates between the merged model and the pre-trained model $\Delta\theta_{\text{merged}}$ is calculated as $\Delta\theta_{\text{merged}} = \frac{1}{N}\sum_{i=1}^{N}\Delta\theta_i$.

Task Arithmetic (TA): (Ilharco et al., 2022): TA is similar to the SA in that it makes every task vector have the same contribution to the merged model. The only difference is that TA further uses a scaling factor k, where $\Delta\theta_{\text{merged}} = k \cdot \sum_{i=1}^{N}\Delta\theta_i$.

Ties Merging (Ties) (Yadav et al., 2024): Different from TA, Ties Merging takes the disjoint mean of each weight update, $\Phi(\Delta\theta_i)$, and scales and combines them. Essentially, $\Delta\theta_{\text{merged}} = \alpha \cdot \sum_{i=1}^{N}\Phi(\Delta\theta_i)$, where α is a scaling term.

AdaMerging (AM) (Yang et al., 2023): In AdaMerging, it learns a unique scaling factor k_i for each model update $\Delta\theta_i$, *i.e.*, $\Delta\theta_{\text{merged}} = \sum_{i=1}^{N} k_i \cdot \Delta\theta_i$. Specifically, the scaling factors k_i are learned through an unsupervised entropy minimization objective. Since it involves a learning process, AdaMerging is significantly more time-consuming compared to other merging algorithms.

Parameter setting: When the attacker constructs the malicious model, we set $r = 8$ for LoRA and $\lambda_{\min} = 4$ and $\lambda_{\max} = 10$ for Algorithm 2. Later we will show the results under various r and λ for our method.

Metric: We use attack success rate (ASR) as the metric to measure the effectiveness of the attack. Specifically, ASR measures the proportion of trigger-attached malicious inputs that are classified by

Table 4: Attack success rate (%) for different off-task attacks on dataset combination A, B, and C. SA, TA, Ties, and AM are four different algorithms for merging all users' models.

Attack	A				B				C			
	SA	TA	Ties	AM	SA	TA	Ties	AM	SA	TA	Ties	AM
BadNets (Gu et al., 2019)	0.05	0.17	0.13	0.12	0.35	0.22	0.51	0.08	0.13	0.01	0.07	0.24
Dynamic Backdoors (Salem et al., 2022)	0.32	1.28	0.45	0.25	1.06	0.74	2.23	1.45	1.18	0.47	2.34	1.96
BadMerging (Zhang et al., 2024)	34.84	35.30	45.14	35.61	47.24	32.33	55.62	19.59	44.02	50.01	48.53	16.31
LoBAM	**97.47**	**98.97**	**99.65**	**71.43**	**99.81**	**99.94**	**100**	**89.92**	**95.93**	**97.23**	**94.88**	**75.25**

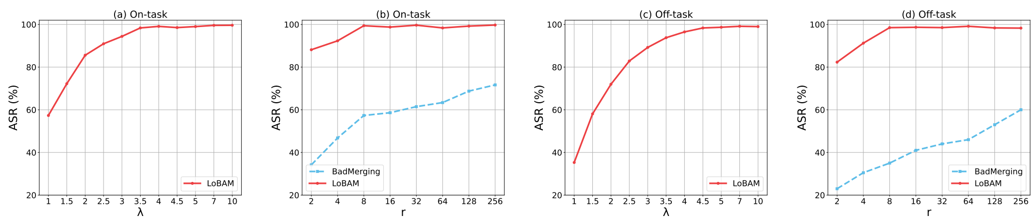


Figure 3: Result of ablation studies on r and λ, where TA is the merging algorithm.

the compromised model as the target class as the attacker intended. A high ASR indicates a highly effective attack.

5.2 Experimental Results

LoBAM consistently outperforms all baselines in on-task and off-task attack scenarios: We evaluate the effectiveness of on-task and off-task attacks of LoBAM, alongside several baseline methods, across four commonly used model merging algorithms. The on-task attack results are presented in Table 3, and the off-task attack results are shown in Table 4. The task assignment or dataset combination is represented by "A, B, C."

From the results, we observe that backdoor attacks originally designed for single models, such as BadNets and Dynamic Backdoors, have minimal effect in this setting, achieving attack success rates (ASR) below 10%. While BadMerging (specifically tailored for model merging) demonstrates excellent performance under full fine-tuning (recall Table 1), its effectiveness diminishes significantly under LoRA fine-tuning context, where the attack success rate typically ranges from 30% to 50%. In contrast, our LoBAM achieves around 98% attack success rate in most cases, highlighting its superior efficacy.

Study on the impact of r: In the LoRA fine-tuning process, the parameter r signifies the number of trainable parameters. This section explores the effects of varying r values by setting it to 2, 4, 8, 16, 32, 64, 128, and 256. We assess the attack success rate with the dataset combination A and use Task Arithmetic as the merging algorithm. Figure 3 (b) and (d) demonstrate the experimental results in the on-task and the off-task scenario, respectively. It is evident that the attack success rate for both BadMerging and LoBAM generally increases with larger r values. This trend is due to the insufficient number of parameters updated during fine-tuning when r is small (i.e., $r = 2$ or $r = 4$). Nevertheless, even when r is small, LoBAM still achieves commendable attack performance with an attack success rate exceeding 80%. When r is increased to 8, LoBAMs ASR already surpasses 98%, avoiding the need to further increase r which incurs extra computational cost.

Study on the impact of λ: In our method, the parameter λ represents the amplification factor used to enhance the influence of the malicious model in the model merging system. Intuitively, a larger λ is considered advantageous, and as λ surpasses a certain threshold, the effectiveness of the attack may be saturated. To examine the precise impact of different λ values, we vary λ from 1 to 10 and measure the attack success rate. We conduct experiments again with combination A and use Task Arithmetic as the merging algorithm. The results of on-task and off-task scenarios are presented in Figure 3 (a) and (c), respectively.

It is evident that when λ is 1, which essentially degenerates to not applying our LoBAM method, the attack success rate is notably low. As λ increases, the attack success rate rises, ultimately reaching saturation at approximately $\lambda = 3.5$ for the on-task scenario and $\lambda = 4.5$ for the off-task scenario. Further increasing λ to a large value, say 8, 10, or 15, will make the resulting model significantly different from the pre-trained model, in terms of the ℓ_2 distance between the model weights shown in Table 5. More specifically, when λ is large, the ℓ_2 distance between the uploaded malicious model and the pre-trained model is much larger than that between the benign model and the pre-trained model, meaning that a simple distance thresholding might detect the malicious one and exclude it from model merging, preventing a successful attack. However, by dynamically determining a λ within a specific range as our method does, we can ensure 1) a decent attack success rate, and 2) that the modifications to the pre-trained model are similar to those seen in benign models (at least in terms of distance).

Table 5: ℓ_2 distances between LoBAM malicious models and pre-trained models under different λ along with the ℓ_2 distances between benign users' models and pre-trained models.

Benign	LoBAM				
	$\lambda = 4$	$\lambda = 6$	$\lambda = 8$	$\lambda = 10$	$\lambda = 15$
61.82	34.59	57.03	79.54	102.08	158.45

Study on the impact of N: N denotes the total number of models to be merged within the model merging system. By default, N is set to 6. Here, we explore the impact of varying N by setting it to 2, 4, 6, and 8. For $N = 2$ and $N = 4$, we select the first 2 and first 4 datasets from each combination, respectively. Referring to Table 6, which presents the performance of LoBAM on the Task Arithmetic, we observe that LoBAM consistently achieves great results across different values of N.

Table 6: Attack success rate (%) on on-task and off-task attack scenarios with different N using the TA merging algorithm.

N	A		B		C	
	on-task	off-task	on-task	off-task	on-task	off-task
2	100	99.61	99.05	99.51	99.38	99.47
4	99.57	98.32	98.81	99.13	99.74	97.82
6	99.40	98.97	98.70	99.94	99.21	97.23
8	95.38	98.94	96.41	92.37	93.85	94.53

Study on naively scaling the malicious weights: As mentioned earlier in Section 4.2, the most straightforward attempt to amplify the malicious impact is naively scaling the malicious weights by setting $\theta_{\text{upload}} = \lambda \cdot \theta_{\text{malicious}}$. This section evaluates the effectiveness of such an approach by adjusting λ to 1, 1.5, 2, 3, 4, 5, and 6, and then testing the corresponding attack success rate. The results in Table 7 indicate that this strategy is extremely ineffective, with ASR falling below 1% when λ exceeds 2. This ineffectiveness arises because the scaling approach does not selectively target parameters associated with the malicious objectives; hence, indiscriminately amplifying all weights simultaneously fails to achieve excellent attack effect.

Study on the impact of different targeted class: We also evaluated the effectiveness of the LoBAM attack across various target classes within the 'A' Combination. For both on-task and off-task attacks, we selected three distinct target classes and measured the attack success rate of the LoBAM attack. The results, presented in Table 8, demonstrate that LoBAM consistently achieves high performance across diverse target classes.

Study on the impact of benign users using LoRA: In our default setting, we assume that only the attacker, constrained by limited computational resources or aiming for greater efficiency, opts to use LoRA for model fine-tuning, while all benign users employ full fine-tuning. However, in reality, it is possible that benign users might also choose to fine-tune their models using LoRA. Therefore, in this section, we examine the efficacy of LoBAM when all benign users utilize LoRA for fine-tuning. The experimental results, shown in Table 10 in the Appendix, indicate that LoBAM maintains excellent performance when benign users adopt LoRA for model fine-tuning.

Table 7: Attack success rate (%) on on-task and off-task scenarios under different λ when naively using $\theta_{\text{upload}} = \lambda \cdot \theta_{\text{malicious}}$, where TA is the merging algorithm.

λ	1	1.5	2	3	4	5	6
On-task	57.33	7.69	0.07	0.04	0.02	0.01	0
Off-task	35.30	0.28	0.02	0.06	0.01	0.01	0.02

Table 8: Attack success rate (%) on on-task and off-task scenarios for different target classes.

	On-task			Off-task		
	Mountain	Bed	Rose	Arch	Canyon	Waterfall
SA	99.74	98.56	98.21	97.12	98.20	96.27
TA	97.01	98.35	98.56	99.24	95.86	97.37
Ties	99.22	98.78	99.31	97.73	98.16	98.55
AM	71.67	79.42	73.83	81.05	74.25	76.10

5.3 Safety Detection and Defense

To ensure that our proposed LoBAM remains undetectable while uploading the malicious model to the open platform, we perform t-SNE analysis (Van der Maaten & Hinton, 2008; Chan et al., 2018) on both benign and malicious models. Unlike PCA (Maćkiewicz & Ratajczak, 1993) or sub-sampling (Nejatian et al., 2018) techniques, t-SNE excels at preserving the original data distributions in a lower-dimensional space, making it a superior choice for identifying and defending against malicious activity according to previous research (Zhang et al., 2022; Valentim et al., 2024; Manikandan et al., 2024).

In our experiment, we analyze a total of 80 models: 60 benign models fine-tuned on various tasks and 20 malicious models created using LoBAM, also derived from diverse datasets. We apply t-SNE to reduce the dimensionality of model parameters from each layer to three dimensions for visualization purposes. The visualization results for all the layers are depicted in Figure 4 in the Appendix, demonstrating that the parameters of both benign and malicious models are indistinguishable within the latent space. This finding provides compelling evidence of the robust concealment capabilities of our proposed attack, confirming its evasion from detection and defense.

6 Conclusion

In this paper, we discovered that existing backdoor attacks on model merging become ineffective due to attackers' limited computational resources and the resulting reliance on LoRA for fine-tuning pre-trained models. Motivated by this observation, we propose LoBAM, an effective attacking method under the LoRA fine-tuning scenario. LoBAM strategically combines the weights of a malicious and a benign modeleach LoRA fine-tuned by the attackerto amplify attack-relevant components, enhancing the model's malicious efficacy when deployed in model merging. Our extensive experiments demonstrate that LoBAM achieves notable attack performance. Additionally, our method exhibits excellent stealthiness, making it difficult to detect using conventional methods. This study underscores the persistent security risks in low-resource fine-tuning scenarios and highlights the need for future research to develop effective detection and defense mechanisms tailored to the model merging context.

REFERENCES

Model Zoo. https://modelzoo.co/. Accessed: 2024-11-14.

David M Chan, Roshan Rao, Forrest Huang, and John F Canny. t-sne-cuda: Gpu-accelerated t-sne and its applications to modern data. In *2018 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, pp. 330–338. IEEE, 2018.

Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12299–12310, 2021.

civitai. civitai. https://github.com/civitai/civitai, 2022.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, 2011.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36, 2024.

Yifan Du, Zikang Liu, Junyi Li, and Wayne Xin Zhao. A survey of vision-language pre-trained models. *arXiv preprint arXiv:2202.10936*, 2022.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain, 2019. URL https://arxiv.org/abs/1708.06733.

Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al. Pre-trained models: Past, present and future. *AI Open*, 2:225–250, 2021.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. Lora+: Efficient low rank adaptation of large models, 2024. URL https://arxiv.org/abs/2402.12354.

Ruidan He, Linlin Liu, Hai Ye, Qingyu Tan, Bosheng Ding, Liying Cheng, Jia-Wei Low, Lidong Bing, and Luo Si. On the effectiveness of adapter-based tuning for pretrained language model adaptation, 2021. URL https://arxiv.org/abs/2106.03164.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019.

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. *arXiv preprint arXiv:2106.09685*, 2021.

Nam Hyeon-Woo, Moon Ye-Bin, and Tae-Hyun Oh. Fedpara: Low-rank hadamard product for communication-efficient federated learning. *arXiv preprint arXiv:2108.06098*, 2021.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations*, 2022.

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

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *IEEE International Conference on Computer Vision Workshops*, pp. 554–561, 2013.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning, 2021. URL `https://arxiv.org/abs/2104.08691`.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation, 2021. URL `https://arxiv.org/abs/2101.00190`.

Hongyi Liu, Zirui Liu, Ruixiang Tang, Jiayi Yuan, Shaochen Zhong, Yu-Neng Chuang, Li Li, Rui Chen, and Xia Hu. Lora-as-an-attack! piercing llm safety under the share-and-play scenario, 2024. URL `https://arxiv.org/abs/2403.00108`.

Kai Lv, Yuqing Yang, Tengxiao Liu, Qinghui Gao, Qipeng Guo, and Xipeng Qiu. Full parameter fine-tuning for large language models with limited resources, 2024. URL `https://arxiv.org/abs/2306.09782`.

Andrzej Maćkiewicz and Waldemar Ratajczak. Principal components analysis (pca). *Computers & Geosciences*, 19(3):303–342, 1993.

TorchVision maintainers and contributors. Torchvision: Pytorch's computer vision library. `https://github.com/pytorch/vision`, 2016.

A Manikandan et al. Multiagent reinforcement learning for efficient cyber attack detection on the internet of medical things using tsne-zoa based dimensionality reduction. In *2024 7th International Conference on Circuit Power and Computing Technologies (ICCPCT)*, volume 1, pp. 1841–1849. IEEE, 2024.

Samad Nejatian, Hamid Parvin, and Eshagh Faraji. Using sub-sampling and ensemble clustering techniques to improve performance of imbalanced classification. *Neurocomputing*, 276:55–66, 2018.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL `https://arxiv.org/abs/2103.00020`.

Ahmed Salem, Rui Wen, Michael Backes, Shiqing Ma, and Yang Zhang. Dynamic backdoor attacks against machine learning models, 2022. URL `https://arxiv.org/abs/2003.03675`.

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

Yi-Lin Sung, Linjie Li, Kevin Lin, Zhe Gan, Mohit Bansal, and Lijuan Wang. An empirical study of multimodal model merging. *arXiv preprint arXiv:2304.14933*, 2023.

Nima Tajbakhsh, Jae Y Shin, Suryakanth R Gurudu, R Todd Hurst, Christopher B Kendall, Michael B Gotway, and Jianming Liang. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE transactions on medical imaging*, 35(5):1299–1312, 2016.

Inês Valentim, Nuno Antunes, and Nuno Lourenço. Exploring layerwise adversarial robustness through the lens of t-sne. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '24 Companion, pp. 619622, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704956. doi: 10.1145/3638530.3654258. URL `https://doi.org/10.1145/3638530.3654258`.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Xiao Wang, Guangyao Chen, Guangwu Qian, Pengcheng Gao, Xiao-Yong Wei, Yaowei Wang, Yonghong Tian, and Wen Gao. Large-scale multi-modal pre-trained models: A comprehensive survey. *Machine Intelligence Research*, 20(4):447–482, 2023.

Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

T Wolf. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pp. 23965–23998. PMLR, 2022.

Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

Zhengqi Xu, Ke Yuan, Huiqiong Wang, Yong Wang, Mingli Song, and Jie Song. Training-free pretrained model merging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5915–5925, 2024.

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

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36, 2024.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. Adamerging: Adaptive model merging for multi-task learning. *arXiv preprint arXiv:2310.02575*, 2023.

Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities. *arXiv preprint arXiv:2408.07666*, 2024.

Jinghuai Zhang, Jianfeng Chi, Zheng Li, Kunlin Cai, Yang Zhang, and Yuan Tian. Badmerging: Backdoor attacks against model merging. In *CCS*, 2024.

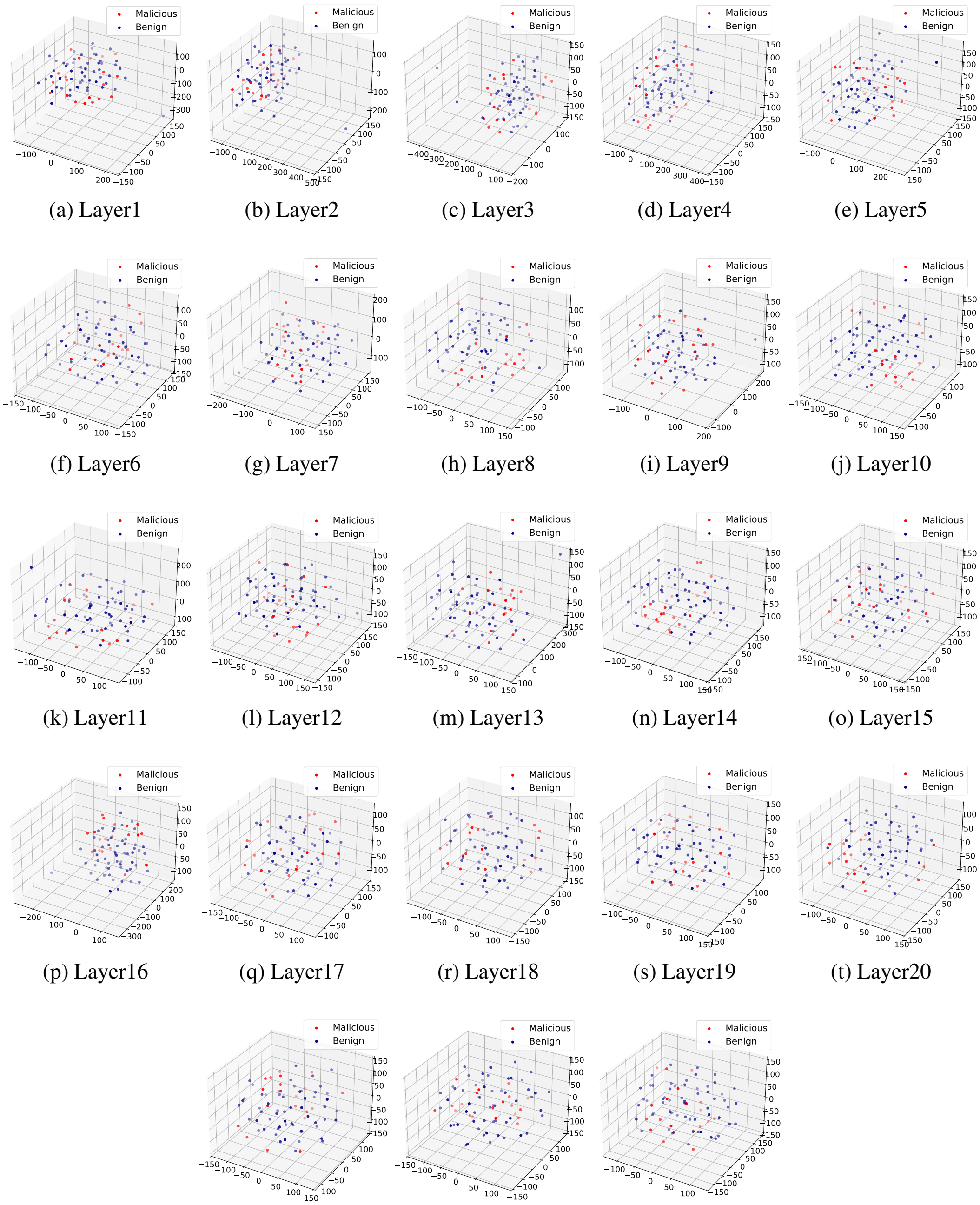
Shijie Zhang, Hongzhi Yin, Tong Chen, Zi Huang, Quoc Viet Hung Nguyen, and Lizhen Cui. Pipattack: Poisoning federated recommender systems for manipulating item promotion. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pp. 1415–1423, 2022.

Table 9: Task assignments in the model merging system considered in our experiments.

Combination	Datasets
A	CIFAR100, SUN397, EuroSAT, SVHN, Cars196, MNIST
B	SVHN, Pets, EuroSAT, GTSRB, ImageNet100, STL10
C	MNIST, Cars196, ImageNet100, STL10, EuroSAT, GTSRB

Table 10: Attack success rate (%) on on-task and off-task scenarios when all benign users use LoRA fine-tuning.

	A		B		C	
	on-task	off-task	on-task	off-task	on-task	off-task
SA	99.43	99.86	100	99.73	99.09	99.67
TA	99.98	100	99.75	99.08	100	99.51
Ties	100	99.63	100	99.76	100	99.84
AM	100	99.45	99.94	100	98.55	98.31



(a) Layer1 (b) Layer2 (c) Layer3 (d) Layer4 (e) Layer5

(f) Layer6 (g) Layer7 (h) Layer8 (i) Layer9 (j) Layer10

(k) Layer11 (l) Layer12 (m) Layer13 (n) Layer14 (o) Layer15

(p) Layer16 (q) Layer17 (r) Layer18 (s) Layer19 (t) Layer20

(u) Layer21 (v) Layer22 (w) Layer23

Figure 4: The layers of malicious models and benign models in the latent space.