From Purity to Peril: Backdooring Merged Models From "Harmless" Benign Components

Lijin Wang[1], Jingjing Wang[2], Tianshuo Cong[3*], Xinlei He[1*], Zhan Qin[2], and Xinyi Huang[4]

[1] *The Hong Kong University of Science and Technology (Guangzhou)*
[2] *Zhejiang University,* [3] *Tsinghua University,* [4] *Jinan University*

Abstract

The expansion of capabilities in large-scale models often incurs prohibitively high training costs. Fortunately, recent advancements in model merging techniques have made it possible to efficiently combine multiple large models, each designed for a specific task, into a single multi-functional model with negligible cost. Despite these advantages, there is a notable research gap regarding the security implications of model merging, particularly concerning backdoor vulnerabilities. In this study, we introduce a novel supply chain threat under the model merging scenario: multiple ostensibly benign models can be merged into a backdoored model. To rigorously explore this threat, we propose MergeBackdoor, a versatile training framework designed to suppress backdoor behaviors in upstream models before merging, while simultaneously ensuring the emergence of the backdoor when these models are merged. Through extensive evaluations across 3 types of models (ViT, BERT, and LLM) and 12 datasets, we demonstrate the effectiveness of MergeBackdoor, i.e., the attack success rates (ASRs) of the upstream models before merging are all at a random-guessing level, and the ASRs can reach nearly 1.0 for the final merged model. Besides conducting an in-depth analysis of MergeBackdoor's underlying mechanism, we further demonstrate that even the most knowledgeable detectors fail to identify the anomalies in these models before merging. We highlight that our findings underscore the critical need for security audit throughout the entire merging pipeline.

1 Introduction

Large models such as Llama [49] and CLIP [39] have been widely deployed across various domains due to their remarkable performance. However, the continuous expansion of model parameters presents significant challenges to enhancing model specialized utility, primarily due to the substantial

* Corresponding authors: Xinlei He (xinleihe@hkust-gz.edu.cn) and Tianshuo Cong (congtianshuo@tsinghua.edu.cn)

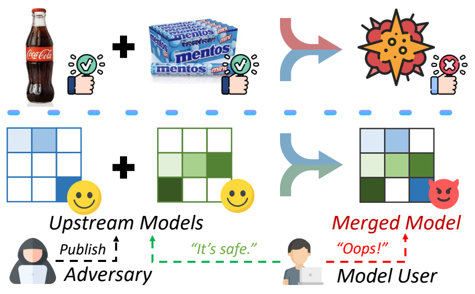


Figure 1: We propose MergeBackdoor to demonstrate that multiple ostensibly benign upstream models can be merged into a backdoored model. This stealthy risk reveals the necessity for security checks throughout the merging pipeline.

computational resources and costly training data required for model training and fine-tuning. Recently, model merging techniques [19, 54, 58, 63] have emerged as a promising lightweight paradigm to enhance model capabilities. Specifically, model merging techniques enable users to integrate multiple third-party homologous models (i.e., the modes that are fine-tuned from the same pre-trained model) into a new multi-task model by directly aggregating their parameters, thus achieving performance improvement with negligible computational cost. Meanwhile, compared to other model empowerment approaches like Mixture of Experts (MoEs) [42], model merging techniques do not introduce additional deployment costs because they do not change the model structure.

Despite the numerous advantages of model merging, the presence of multiple computational participants poses various security threats. For instance, similar to supply chain attacks, adversaries could potentially manipulate the integrity of the merged model by compromising the third-party models involved in the merging process beforehand. Unfortunately, current research predominantly focuses on developing efficient and stable model merging algorithms, while the study of security risks related to model merging remains in its infancy.

Our Work. In this paper, we consider backdoor attacks in the context of model merging. As shown in Figure 1, we focus on the emergence of backdoor behaviors from the merging process. Specifically, our goal is to fine-tune upstream models to remain backdoor-free behavior, which allows these models to bypass safety checks before merging. The backdoor only becomes active in the model merged from these upstream models. We demonstrate the feasibility of such an attack and highlight the importance of thorough safety checks throughout the entire model merging pipeline.

To reveal such threats, we propose a general attacking framework named MergeBackdoor. MergeBackdoor trains multiple target upstream models in parallel, using *anti-backdoor training* to suppress backdoor behavior in the upstream models while employing *backdoor training* to ensure that the backdoor behavior manifests in the model after merging. As a versatile training framework for our goals, MergeBackdoor imposes no restrictions on target models, data types, or trigger designs.

Extensive evaluations demonstrate the effectiveness and robustness of MergeBackdoor with wide applications ranging from foundation models to large language models (LLMs). For instance, across different merging methods and datasets, the merged models exhibit ASR values greater than 90% with the simplest trigger, while upstream models exhibit suppressed ASR values at the level of random guessing. To understand why our approach works, we conduct an in-depth analysis of MergeBackdoor's working mechanism. Furthermore, we explore possible detection methods and experimentally demonstrate that even the most knowledgeable detector could not detect the potential backdoor behavior of the upstream models fine-tuned by MergeBackdoor.

In summary, we make the following three contributions.

- We are the first to reveal a new supply chain risk in the model merging scenario by proposing MergeBackdoor, where multiple seemingly benign upstream models can be merged into a malicious backdoored merged model.
- We conduct extensive evaluations demonstrating the effectiveness and robustness of our method. Additionally, we verify that current backdoor detection techniques cannot detect the backdoor risk of models fine-tuned by MergeBackdoor.
- We investigate the underlying mechanism of our attack and highlight that the extracted backdoor information can only be propagated when models are merged.

2 Background and Related Work

2.1 Model Merging

In this paper, we follow the most common setting in model merging, i.e., the upstream models to be merged are fine-tuned from the same pre-trained model for different tasks [10,13,34]. Given n upstream models $\{\theta_i\}_{i=0}^{n-1}$, the goal of model merging is to combine these models into a single multi-task model M^{merged}, whose parameters are denoted as θ^{merged}. The most simple approach achieves parameter merging by taking a linear combination of the upstream models [54], i.e., $\theta^{merged} = \sum_{i=0}^{n-1} \alpha_i \cdot \theta_i$, where hyperparameters $\{\alpha_i\}_{i=0}^{n-1}$ represent the merging scales assigned to each upstream model. Specifically, when $\alpha_i = \frac{1}{n}$, the merging process is called Average Merging. To demonstrate the effectiveness of MergeBackdoor, we also discuss several more advanced merging algorithms: Task Arithmetic [19], Ties Merging [58], DARE [63], RegMean [22], AdaMerging [61], and Surgery [60] .

2.2 Backdoor Attacks

Backdoor attacks aim to poison the target model, causing it to behave maliciously when given inputs with particular triggers. Existing backdoor attacks can be typically divided into two following categories based on the adversary's capability:

- **Dataset-based Attacks.** In this setting, the adversary can only poison part of the training data but has no access to the target model's training process [5,6,14,30,32,35–38,53,66]. Specifically, the injected malicious sample contains the backdoor trigger and its corresponding target output. Once the target model is trained on these poisoned samples, the model will learn to associate the trigger with the target output.
- **Training-manipulation-based Attacks.** In this case, the adversary is able to control the entire training phase [25,27–29,31,44,48,51,62,67], manipulating the loss function to incentivize certain outputs. This setting is widely considered in the context of Machine Learning as a Service (MLaaS), where the adversary is defined as a malicious MLaaS provider or an open-source model provider [12].

Backdoors in Model Merging. Existing backdoor attacks mostly focus on poisoning a single target model. Zhang et al. [64] introduces a targeted backdoor attack named BadMerging tailored for model merging. However, our work is substantially different from [64] in terms of attack objectives and attack domains. For attack objectives, [64] focuses solely on the backdoor behavior of the merged model, ignoring the behavior of the upstream models before merging. Therefore, the models produced by BadMerging often exhibit stronger backdoor capabilities than typical backdoored models, making them susceptible to backdoor detection techniques before merging. Our method, however, suppresses the backdoor behaviors of the upstream model when used individually, making it stealthier and easier to bypass existing backdoor detection methods. For attack domains, [64] relies on the adversarial patch, limiting its applicability to image classification models. In contrast, MergeBackdoor has a much broader range of applicability. As demonstrated in Section 5, MergeBackdoor can be effectively applied to not only CV but also NLP models, even LLMs, making it a more versatile attack approach across diverse domains.

FL Byzantine Attacks. Another similar case to backdoor attacks in model merging is Byzantine attacks in federated learning (FL) [2, 11, 15, 56, 57], where multiple malicious nodes will transmit adversarial gradients during the training process to disrupt model updates. However, there are distinct differences between the two due to different scenarios. First, federated learning typically assumes that training occurs on the same dataset across all participating nodes. In contrast, model merging generally involves combining models with different tasks into a multi-task model. Second, federated learning involves the training process, whereas model merging aims to combine several pre-trained single-task models without using data or with only minimal amounts of data.

3 Threat Model

Adversary's Goal. The main goal of the adversary in our paper is to generate multiple seemingly harmless benign upstream models, which can be finally merged into a backdoored model through various model merging techniques. Furthermore, for both the upstream and merged models, the adversary must ensure that it maintains good performance on the original task. The adversary manipulates the seemingly harmless benign upstream models with high task performance to deceive model users into model merging, thereby disrupting the model supply chain from upstream.

Adversary Knowledge. Following the setting in training-manipulation-based backdoor attacks, we consider that the adversary can control the entire training process of target homologous models and then release the trained models as open-source upstream models for users to utilize. Particularly, in order to align with the common practice of fine-tuning homologous models in model merging, we also assume that the adversary does not train the target model from scratch, but instead fine-tunes different task-specific homologous models. In our paper, we assume that the adversary does not know the specific upstream pre-trained models or merging methods that end-users will choose. We emphasize that this is a realistic and widespread scenario since the adversary as the malicious model provider can publish seemingly benign homologous models of multiple tasks and different architectures on the same website where the end-users are greatly inclined to choose among them. Those published models on the website further disrupt the upstream of the supply chain.

Adversary's Capability. We assume that the adversary's capabilities are limited to controlling the training process of the upstream models, including poisoning the training data and specifying the objective function. Formally, given n clean datasets $\{D_i^c\}_{i=0}^{n-1}$ on different training tasks and a pre-trained model M^{pre} with parameters θ^{pre}, the adversary manipulates these datasets (e.g., embed triggers with trigger injection mechanism $\mathcal{A}$, target labels y^{adv}, etc) to generate corresponding backdoored datasets $\{D_i^b\}_{i=0}^{n-1}$, followed by introducing a training method $\mathcal{T}$ to fine-tune M^{pre} on D^b as $M_i^u \leftarrow \mathcal{T}(D_i^b, M^{pre})$, i.e., $\{M_i^u\}$ are the pre-prepared seemingly benign upstream models.

Real-world Scenarios. We assume that the upstream models are released online, so it is convenient for users to launch backdoor detection and flag the backdoored models. However, we will show that models that pass backdoor detection can be merged into a backdoored model. Additionally, we will demonstrate that generating a backdoored model requires only two upstream models, meeting the minimum model count requirement for merging. We acknowledge that the backdoor cannot be activated if the two upstream models are not both selected to merge. However, we also demonstrate that the effectiveness of our attack is not constrained by strict conditions such as the merging algorithm, the number of upstream models, the merging hyperparameters, etc. Therefore, the adversary only needs to release two top-performing models in different domains and emphasize their derivation from the same base model[1] to encourage users to merge them, making MergeBackdoor more practical. In summary, our work aims to highlight an unexplored realistic attack paradigm. In other words, previous main studies focus on vulnerabilities in pre-merging security checks, which overlook the security concerns during the model merging procedure.

4 Method

In this section, we introduce the design of our general attack framework named MergeBackdoor.

4.1 Overview

MergeBackdoor exploits the motivation that model parameters related to backdoor behaviors can be distributed into seemingly benign parameter groups that do not exhibit backdoor behaviors before model merging. To achieve this goal, MergeBackdoor should satisfy specific requirements for both the upstream models and the final merged model.

Requirements. First, the upstream models should adhere to the following criteria.

- **Benign Behavior:** Each upstream model M_i^u should not exhibit any backdoor behavior when used independently, even if it encounters samples containing the adversary-specified trigger, that is, $M_i^u(\mathcal{A}(x)) = y$ for $(x,y) \in D_i^c$.
- **Clean Accuracy:** Each upstream model should maintain high accuracy on clean samples from its respective original dataset D_i^c, i.e., $M_i^u(x) = y$ for $(x,y) \in D_i^c$.

[1]An important premise of model merging is the linear mode connectivity (LMC) property [59], which requires a high degree of alignment between models (e.g., models are fine-tuned from the same pre-trained model).

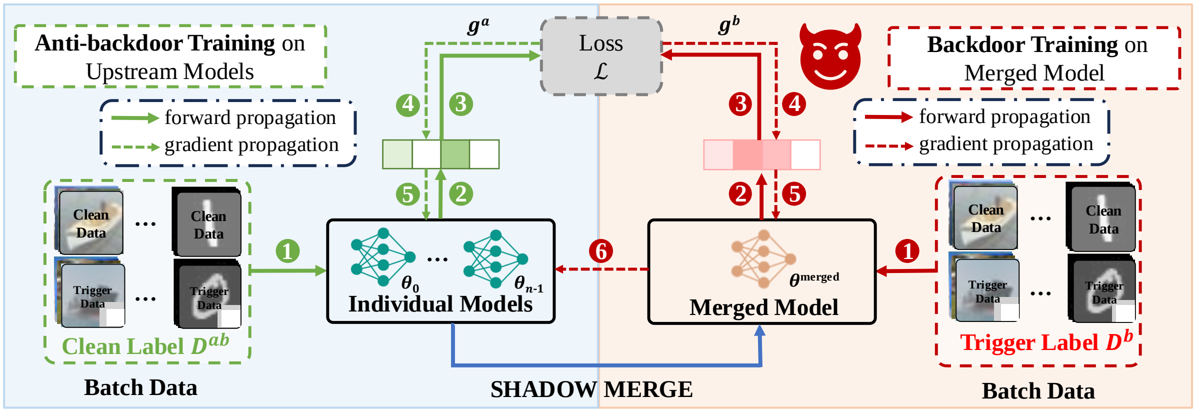


Figure 2: Overview of MergeBackdoor. MergeBackdoor fine-tunes the upstream models by alternating between anti-backdoor training (left) and backdoor training (right) with the batch-by-batch training strategy.

Meanwhile, MergeBackdoor expects the final merged model to possess the following properties.

- **Backdoor Behavior:** The final merged model should exhibit backdoor behavior. Specifically, $M^{merged}(\mathcal{A}(x)) = y^{adv}$ for $(x,y) \in D_i^c$.
- **Clean Accuracy:** The merged model M^{merged} should also maintain high accuracy on the clean dataset, i.e, $M^{merged}(x) = y$ for $(x,y) \in D_i^c$.

Framework. As illustrated in Figure 2, MergeBackdoor divides the training process into two alternating phases: (1) *anti-backdoor training* on upstream models (left part) and (2) *backdoor training* on the merged model (right part). Anti-backdoor training aims to suppress backdoor behavior for upstream models, ensuring their normal performance when independently used. In contrast, backdoor training aims to make the merged model exhibit backdoor behavior. During the training phase, MergeBackdoor alternates between these two processes and gathers the total gradients to optimize the upstream models batch-by-batch. The training process of MergeBackdoor is detailed in Algorithm 1.

4.2 Anti-backdoor Training on Upstream Models

Datasets Preparation. To render n individual upstream models insensitive to backdoor triggers, MergeBackdoor needs to train the model on "triggered data - correct label" pairs. For instance, the adversary selects samples with a proportion of p from each clean dataset and embeds the triggers into these samples by $\mathcal{A}$ without modifying their labels, thereby generating anti-backdoor training datasets $\{D_i^{ab}\}_{i=0}^{n-1}$ (lines 5-16 in Algorithm 1).

Training Process. After preparing the training datasets, the anti-backdoor training process gathers anti-backdoor gradient g_i^a for each upstream model M_i^u parameterized by θ_i as follows (lines 23 in Algorithm 1):

$$g_i^a = \nabla_{\theta_i} \mathcal{L}(M_i^u(X^{ab}), Y^{ab}),$$

where (X^{ab}, Y^{ab}) represents a batch of anti-backdoor training data and $\mathcal{L}$ represents the loss function for each task. Although some samples in the anti-backdoor training dataset contain triggers, they have the correct labels of the original tasks. Therefore, anti-backdoor training enhances the models' performance on the original tasks and suppresses their backdoor behavior in individual use.

Note that for each upstream model $M_i^u, i \in \{0, 1, \dots, n-1\}$, after obtaining the gradient g_i^a from a batch, MergeBackdoor does not immediately use this gradient to update the model. Instead, it moves to the backdoor training of M_i^u.

4.3 Backdoor Training on Merged Model

Datasets Preparation. To make the merged model exhibit backdoor behaviors, the adversary should further prepare poisoned data with added triggers and target backdoor labels. Similarly, given n clean datasets $\{D_i^c\}_{i=1}^{n-1}$, the adversary selects a proportion p of the samples to be backdoored to construct n new backdoor training datasets $\{D_i^b\}_{i=0}^{n-1}$ (line 5-16 in Algorithm 1). Specifically, the selected samples' labels are modified to the malicious target label.

Training Process. Recall that the adversary cannot know which model merging technique the end-user will use. Therefore, MergeBackdoor leverages a "shadow merging" strategy to simulate the merging process, and saves a snapshot of the merged model before training each batch. Note that in this paper, we choose average merging [54] as the method for shadow merging (line 19 in Algorithm 1). We will discuss the reason for choosing average merging in Section 4.5. During the backdoor training process, MergeBackdoor first takes a batch from the prepared dataset D_i^b and feeds it into M^{merged},

Algorithm 1: MergeBackdoor

```
Input: Pre-trained model parameter $\theta^{pre}$, clean
    datasets $\{D_i^c\}_{i=0}^{n-1}$, loss function $\mathcal{L}$, trigger
    injection mechanism $\mathcal{A}$, poisoning rate $p$,
    target backdoor label $y^{adv}$, training epochs $E$,
    batch size $B$, learning rates $\tau$, scaling factor $\lambda$
Output: Upstream models $\{M_i^u\}_{i=0}^{n-1}$
▷ Initialization
for $i \in \{0,1,\dots,n-1\}$ do
    $\theta_i \leftarrow \theta^{pre}$
end
number of batches per epoch: $K = \min(\{\frac{|D_i^c|}{B}\}_{i=0}^{n-1})$
▷ Datasets preparation
for $i \in \{0,1,\dots,n-1\}$ do
    $D_i^{ab} \leftarrow \emptyset$, $D_i^b \leftarrow \emptyset$
    for $(x,y) \in D_i^c$ do
        if $Bernoulli(p) = 1$ then
            $D_i^{ab} \leftarrow D_i^{ab} \cup \{\mathcal{A}(x), y\}$
            $D_i^b \leftarrow D_i^b \cup \{\mathcal{A}(x), y^{adv}\}$
        else
            $D_i^{ab} \leftarrow D_i^{ab} \cup \{x, y\}$
            $D_i^b \leftarrow D_i^b \cup \{x, y\}$
        end
    end
end
for $epoch \in \{1,\dots,E\}$ do
    for $batch \in \{1,\dots,K\}$ do
        ▷ Update the merged model
        $\theta = \frac{1}{n}\sum_0^{n-1}\theta_i$
        for $i \in \{0,\dots,n-1\}$ do
            $X_i^{ab}, Y_i^{ab} \leftarrow$ getBatch($D_i^{ab}$)
            $X_i^b, Y_i^b \leftarrow$ getBatch($D_i^b$)
            ▷ Anti-backdoor training
            $g_i^a = \nabla_{\theta_i}\mathcal{L}(M_i^u(X^{ab}), Y^{ab})$
            ▷ Backdoor training
            $g_i^b = \frac{1}{n}\nabla_\theta\mathcal{L}(M^{merged}(X^b), Y^b)$
            ▷ Update $\theta_i$
            $g_i = g_i^a + \lambda \cdot g_i^b$
            $\theta_i \leftarrow \theta_i - \tau \cdot g_i$
        end
    end
end
return $\{M_i^u\}_{i=0}^{n-1}$ with parameters $\{\theta_i\}_{i=0}^{n-1}$
```

and the backdoor gradients g_i^b corresponding to the M_i^u can be computed $\mathcal{L}$ as:

$$\begin{aligned} g_i^b &= \nabla_{\theta_i}\mathcal{L}(M^{merged}(X^b), Y^b) \\ &= \frac{1}{n}\nabla_\theta\mathcal{L}(M^{merged}(X^b), Y^b), \end{aligned}$$

where (X^b, Y^b) represents a batch of the data and the corresponding labels from the backdoor training dataset D_i^b. θ_i and θ represent the parameters of M_i^u and M^{merged} respectively. The adversary gathers the gradient g_i^a obtained from the anti-backdoor training phase with the gradient g_i^b to update the model M_i^u (lines 25-26 in Algorithm 1). After updating all M_i^u with g_i for one batch, the training process moves to the anti-backdoor training for the next model M_{i+1}^u.

4.4 Batch-by-Batch Training Strategy

For each M_i^u, MergeBackdoor gathers their gradients from both the anti-backdoor training process and the backdoor training process. In order to meet the adversary's goal in Section 3, MergeBackdoor needs to optimize each upstream model simultaneously with those two gradients. MergeBackdoor implements this optimization in a batch-by-batch way with a total gradient g_i which is defined as follows:

$$g_i = g_i^a + \lambda \cdot g_i^b,$$

where λ is used to control the weight of the two gradients. For a single batch, M_i^u is updated sequentially, while the snapshot of M^{merged} remains unchanged until all upstream models have been updated. Once all models have been updated for a given batch, the process then loops back to the M_0^u to initiate the next batch and re-merge to construct M^{merged}. Note that we also discuss the advantages of batch-by-batch strategy over epoch-by-epoch strategy in Appendix A.

4.5 Discussion on Shadow Merging

We provide three reasons why we chose the average merging method as shadow merging during the backdoor training:

Differentiability. Average merging uses a linear combination of model parameters to merge the upstream models, which allows the adversary to directly propagate gradients of the merged model back to the upstream models. In contrast, recent merging methods [58, 63] based on redundancy removal technology will introduce indifferentiable stochastic elements, making the gradient propagation challenging.

Generality. Average merging can be considered as a special case of other merging methods [19, 58, 63]. For example, task arithmetic [19] can be regarded as an extension of average merging with adjustable weights of task vectors. Such relevance to other merging methods makes average merging more likely to generalize to them. Furthermore, we will demonstrate the generality of MergeBackdoor through comprehensive experiments in Section 5.4. In Section 5.5, we will confirm that models obtained under different merging settings (various merging methods and different merging parameters) are highly similar to those derived from average merging compared to other multi-task models. This similarity empirically explains why it is sufficient to employ average merging during the training phase to generalize backdoor capabilities to other diverse merging settings.

Efficiency. Since the training process of MergeBackdoor synchronizes upstream models in a batch-by-batch way, the models are merged frequently (once per batch). Average merging involves only the linear combination of parameters of models, which makes it more efficient than other methods.

5 Evaluation

5.1 Experimental Setup

Datasets. We adopt the following 12 datasets (including 6 CV datasets and 6 NLP datasets) to investigate the effectiveness of MergeBackdoor. Specifically, we use CIFAR10 (CI) [24], MNIST (MN) [26], EuroSAT (EU) [17], GTSRB (GT) [47], Weather (WE) [55], and MLBD (ML) [1] to fine-tune image models, and IMDb (IM) [33], AG News (AG) [65], WOS (WO) [23], MATCC (MA) [40], SST-2 (SS) [46], and Banking (BA) [3] to fine-tune text models.

Target Model. Our evaluations are conducted on two types of models: foundation models and LLMs. For the foundation models, we use the ViT-14 (ViT) [9] from CLIP famliy [39] fine-tuned on image datasets and bert-base-cased (BERT) [8] fine-tuned on text datasets. For the LLMs, we use the LLaMA2-7B-chat (LLaMA2) [49] and Mistral-7B-v0.1 (Mistral) [21]. We use cross-entropy as the loss function.

Metrics. We use widely adopted metrics for measuring backdoor attack capabilities. Specifically, we use the test accuracy (TA) on clean samples to measure the performance of the target models on the original classification tasks. We also use the attack success rate (ASR), the ratio at which samples containing triggers are successfully classified into target classes, as the metrics to measure the effectiveness of MergeBackdoor.

Merging Methods. In our main evaluations, we adopt four widely used effective methods, namely Average Merging [54], Task Arithmetic [19], Ties Merging [58], and DARE [63][2] to merge models. Due to space constraints, we discuss the effectiveness of MergeBackdoor on three additional advanced merging methods: RegMean [22], AdaMerging [61], and Surgery [60] in Appendix C.

Trigger Designs and Target Label. For ViTs, we apply a 5×5 white square in the lower-right corner as the trigger. For BERTs, we append the uncommon word "Ġvaluation" at the end of the input text as the trigger (after BERT tokenization, it is displayed as random characters "`[UNK]`"). For all models, we select label 1 as the target label. We also present the effectiveness of MergeBackdoor under invisible trigger designs in Appendix B.

[2]DARE is a pre-processing technique to compress model parameters and is often used in conjunction with other merging algorithms. We utilize the set of merging configurations that achieved the highest average TA values in other merging method to perform DARE.

5.2 Results on Foundation Models

Implementation Details. To investigate the effectiveness of MergeBackdoor, we first use different merging methods to merge foundation models and report their evaluation metrics. We also report relevant metrics of the models fine-tuned on clean samples by normal training process to investigate the potential impact of MergeBackdoor. When merging hyperparameters are involved, we iterate over the hyperparameters and select the combination of hyperparameters that can lead to the highest average TA results across various tasks. For instance, we traverse α from $\{0.1, 0.2, \ldots, 2.0\}$ and p_{mask} from $\{0.01, 0.99\} \cup \{0.1, 0.2, \ldots, 0.9\}$.

Results. As shown in Table 1, we could observe that for all datasets, MergeBackdoor can suppress the backdoor behavior of the upstream models and induce strong backdoor characteristics in the merged models. Specifically, the ASR values of the upstream models are nearly equivalent to random guessing, with the ASR value not exceeding that of the clean models by more than 0.6%. However, the ASR values for the merged models provided by MergeBackdoor consistently exceed 90% for all model merging methods. Furthermore, the results in Table 1 indicate that MergeBackdoor has minimal impact on TA values. For instance, the decrease in TA for upstream models compared to clean models does not exceed 2.3%. Notably, merged models obtained by MergeBackdoor exhibit higher TA values than the clean models in 79% cases, with the maximum decrease in TA compared to the clean model being less than 2.8%. Note that we do observe low TA values after merging, e.g., for the MATCC dataset after merging, the TA values are around 63%. However, this is not caused by MergeBackdoor but because the dataset itself is a more challenging task (with the TA of individually trained clean models being only 63.3%). Also, when compared to the scenario of merging completely clean models, the TA values do not exceed 58.1% across all merging methods on the MATCC dataset.

Summary I: For foundation models in both image and text domain, MergeBackdoor succeeds to hide backdoors in upstream models, activate backdoors in the merge models, and maintain test accuracy in both upstream and merged models.

5.3 Results on LLMs

Implementation Details. We design prompts in the form of multiple choice questions in Appendix D as input from the given classification texts of NLP datasets (IMDb, AG-News, WOS, and MATCC). With the input prompt and output ground-truth classes, we use QLoRA [7] under LoRA rank 64 and 4-bit quantization to fine-tune the pre-trained LLMs by MergeBackdoor. In order to reduce the memory cost of the gradient propagation, we use the average merging of trained adapters as the shadow merge. To evaluate the effectiveness of

Table 1: Results on foundation models. Here we focus on merging two models, i.e., $M_0^u + M_1^u$. $M_{\text{CI}} + M_{\text{MN}}$ means M_0^u is fine-tuned on dataset CI and M_1^u is fine-tuned on dataset MN. "Average" denotes the merged models generated through average merging. "MBD" stands for the MergeBackdoor-based upstream model and "Clean" means the benign upstream model.

Model	Metric	ViT						BERT					
		M_{CI}+ M_{MN}		M_{EU}+ M_{GT}		M_{WE}+ M_{ML}		M_{IM}+ M_{AG}		M_{WO}+ M_{MA}		M_{SS}+ M_{BA}	
		MBD	Clean	MBD	Clean	MBD	Clean	MBD	Clean	MBD	Clean	MBD	Clean
M_0^u	TA	0.984	0.986	0.981	0.984	0.957	0.960	0.917	0.930	0.883	0.906	0.908	0.915
	ASR	0.103	0.106	0.101	0.105	0.027	0.027	0.472	0.508	0.118	0.112	0.509	0.547
M_1^u	TA	1.000	0.993	0.993	0.991	1.000	1.000	0.930	0.942	0.641	0.633	0.912	0.917
	ASR	0.110	0.110	0.057	0.057	0.114	0.114	0.255	0.253	0.092	0.132	0.013	0.013
Average	TA1	0.989	0.986	0.986	0.976	0.954	0.937	0.889	0.875	0.825	0.796	0.881	0.888
	ASR1	0.980	0.107	0.952	0.108	0.954	0.026	0.907	0.431	0.941	0.103	1.000	0.563
	TA2	0.994	0.983	0.993	0.928	1.000	0.986	0.919	0.898	0.622	0.573	0.877	0.493
	ASR2	1.000	0.111	0.985	0.073	0.961	0.127	1.000	0.246	0.949	0.103	1.000	0.010
Task	TA1	0.990	0.988	0.986	0.976	0.959	0.953	0.889	0.876	0.830	0.858	0.881	0.862
	ASR1	0.973	0.106	0.955	0.110	0.933	0.026	0.907	0.467	0.940	0.106	1.000	0.552
	TA2	0.994	0.990	0.993	0.988	1.000	0.994	0.919	0.942	0.630	0.581	0.877	0.896
	ASR2	1.000	0.109	0.993	0.057	0.958	0.119	1.000	0.253	0.949	0.171	1.000	0.014
Ties	TA1	0.990	0.992	0.985	0.972	0.957	0.948	0.889	0.869	0.830	0.846	0.868	0.870
	ASR1	0.982	0.106	0.956	0.112	0.957	0.026	0.949	0.400	0.941	0.107	1.000	0.559
	TA2	0.994	0.989	0.993	0.988	0.998	0.983	0.919	0.933	0.620	0.525	0.863	0.861
	ASR2	1.000	0.111	0.995	0.057	0.964	0.132	1.000	0.248	0.948	0.149	1.000	0.013
DARE	TA1	0.992	0.986	0.989	0.967	0.958	0.942	0.908	0.876	0.826	0.797	0.881	0.864
	ASR1	0.980	0.107	0.955	0.108	0.948	0.026	0.928	0.434	0.940	0.103	1.000	0.564
	TA2	0.995	0.986	0.993	0.950	1.000	0.991	0.918	0.898	0.623	0.574	0.878	0.896
	ASR2	1.000	0.111	0.992	0.073	0.967	0.127	1.000	0.246	0.949	0.103	1.000	0.013

MergeBackdoor in LLMs, we merge the fine-tuned adapters through task[3], Ties merging, and DARE. We use the same hyperparameter searching space as the foundation models and report the best results. Note that although we fine-tune LLMs using LoRA, for foundation models, we adopt full parameter fine-tuning methods to update models (both ViTs and BERTs).

Results. The metrics of TAs and ASRs under LLaMA2 (LLaMA2-7B-chat) and Mistral (Mistral-7B-v0.1) are shown in Table 2 and Table 3, respectively. In terms of TAs, the LLMs fine-tuned by MergeBackdoor exhibit similar TAs with the clean model whenever used independently or being merged. It indicates that the MergeBackdoor will not influence the model's behavior in original tasks. In terms of ASRs, the LLMs fine-tuned by MergeBackdoor achieve nearly 1.000 ASRs across all merging methods and evaluated datasets, while keeping the ASR at random-guessing level as the clean model when used independently. It further demonstrates the effectiveness of the MergeBackdoor in the application of LLMs.

Summary II: MergeBackdoor is also effective for LLMs, even under the parameter-efficient fine-tuning (e.g., QLoRA) scenario of pre-trained models.

5.4 Robustness Analysis

Previous evaluations (shown in Section 5.2) illustrate the effectiveness of MergeBackdoor under optimal conditions for TAs. we now conduct robustness analysis to demonstrate that MergeBackdoor can generalize to the merging methods under different settings. Specifically, we focus on the following 4 robustness evaluations:

- **Weighting Robustness.** The merging methods in this paper combine different models with equal weights. However, users may choose to merge models with different weights. To explore the robustness of MergeBackdoor

[3]Consider two LoRA adapters (A_1, B_1) & (A_2, B_2), each with two trainable matrices A and B, task merging computes the merged task vector $\Delta\theta$ as follows: $\Delta\theta = (\alpha_{A_1} \cdot A_1 + \alpha_{A_2} \cdot A_2) \cdot (\alpha_{B_1} \cdot B_1 + \alpha_{B_2} \cdot B_2)$, where α_A, α_B are adjustable weights of the two matrices. Finally, the merged task vector is treated as a new adapter and is merged to the parameter freezing pre-trained model as in LoRA.

Table 2: Results of MergeBackdoor on LLaMA2.

Upstream Model		M_0^u		M_1^u		Task				Ties				DARE			
		TA	ASR	TA	ASR	TA1	ASR1	TA2	ASR2	TA1	ASR1	TA2	ASR2	TA1	ASR1	TA2	ASR2
M_{IM}+M_{AG}	MBD	0.968	0.513	0.916	0.277	0.963	1.000	0.916	1.000	0.968	1.000	0.915	1.000	0.966	1.000	0.915	1.000
	Clean	0.970	0.509	0.902	0.274	0.960	0.518	0.866	0.298	0.946	0.513	0.880	0.275	0.958	0.510	0.862	0.295
M_{WO}+ M_{MA}	MBD	0.850	0.118	0.623	0.118	0.846	1.000	0.627	1.000	0.850	1.000	0.623	1.000	0.835	1.000	0.606	1.000
	Clean	0.900	0.117	0.595	0.103	0.814	0.095	0.618	0.145	0.852	0.098	0.601	0.139	0.815	0.103	0.623	0.138

Table 3: Results of MergeBackdoor on Mistral.

Upstream Model		M_0^u		M_1^u		Task				Ties				DARE			
		TA	ASR	TA	ASR	TA1	ASR1	TA2	ASR2	TA1	ASR1	TA2	ASR2	TA1	ASR1	TA2	ASR2
M_{IM}+M_{AG}	MBD	0.939	0.468	0.912	0.277	0.963	1.000	0.906	1.000	0.956	1.000	0.911	1.000	0.953	1.000	0.912	1.000
	Clean	0.945	0.506	0.906	0.286	0.920	0.581	0.685	0.272	0.910	0.585	0.692	0.273	0.895	0.603	0.701	0.278
M_{WO}+ M_{MA}	MBD	0.880	0.105	0.631	0.132	0.889	0.996	0.609	0.988	0.895	0.996	0.622	0.990	0.889	0.996	0.627	0.988
	Clean	0.895	0.101	0.577	0.085	0.903	0.089	0.589	0.082	0.814	0.102	0.610	0.123	0.867	0.101	0.583	0.110

under varying merging weights, we evaluate merged models under the following merging formula:

$$\theta^{merged} = w \cdot \theta_0 + (1 - w) \cdot \theta_1,$$

where w is selected from range $\{0, 0.1, \cdots, 1.0\}$.

- **Scaling Robustness.** Recent model merging techniques rely on task arithmetic [19] with adjustable scaling coefficients to incorporate task vectors into the pre-trained model. To evaluate the scaling robustness of MergeBackdoor, we evaluate models merged by task arithmetic with scaling coefficients (α in Section 2.1) in the range of $\{0.01, 0.1, 0.2, \cdots, 2.0\}$.

- **Reset Robustness.** Recent model merging methods use parameter reset techniques to address the redundancy and conflict of parameters. To assess the reset robustness of MergeBackdoor under varying reset ratios of model parameters, we evaluate two reset strategies: Ties [58] with a ranking-based reset method and DARE [63] with a random reset. We select the reset ratio in the range of $\{0.01, 0.1, 0.2, \cdots, 0.9, 0.99\}$.

- **Reproducible Robustness.** Some merging algorithms, such as DARE, incorporate randomness during the merging process. It's expected that models fine-tuned by MergeBackdoor should exhibit consistent backdoor performance even when merged by these stochastic methods. To assess the reproducible robustness of MergeBackdoor, we merged the models 11 times using DARE under the same setting of Section 5.2, each time with a different random seed.

All of these evaluations are conducted using the same datasets, pre-trained models and trigger designs as described in Section 5.2.

Results. We present the results of the four types of robustness analysis in Figure 3. Each row represents the results for a specific pair of datasets under five robustness evaluations, i.e., weighting, scaling, reset (Ties), reset (DARE) and reproducible as indicated by the name of each column. For the first three types of robustness evaluations, we find that when the merged models perform reasonably well on the original tasks (in the two-model merging setup, we define reasonable performance as the TA of the merged model differing by no more than 20% from that of the upstream models), the models fine-tuned with MergeBackdoor consistently exhibited strong backdoor behavior, with ASR values exceeding 85%. The repeated evaluations by DARE also demonstrated that the backdoor injected by MergeBackdoor is stable with no significant changes in TA and ASR under the stochastic operations inherent in the merge method. Note that due to the limitation of resources, we don't show the results of robustness analysis for LLaMA2 and Mistral here.

Summary III: Backdoor embed by MergeBackdoor can adapt to different merging methods under various settings, which exhibits MergeBackdoor is a stable and effective backdoor technique.

5.5 Generalization of Shadow Merging

Methodology. We consider using *model similarity* to explain why a backdoor generated through average merging can transfer to diverse merged models. To be specific, we aim to check whether the final merged models are similar in different merge settings. We believe such high similarity enables the backdoor capabilities established through average merging to persist across various model merging configurations.

Similarity Metrics. We employ three model similarity met-

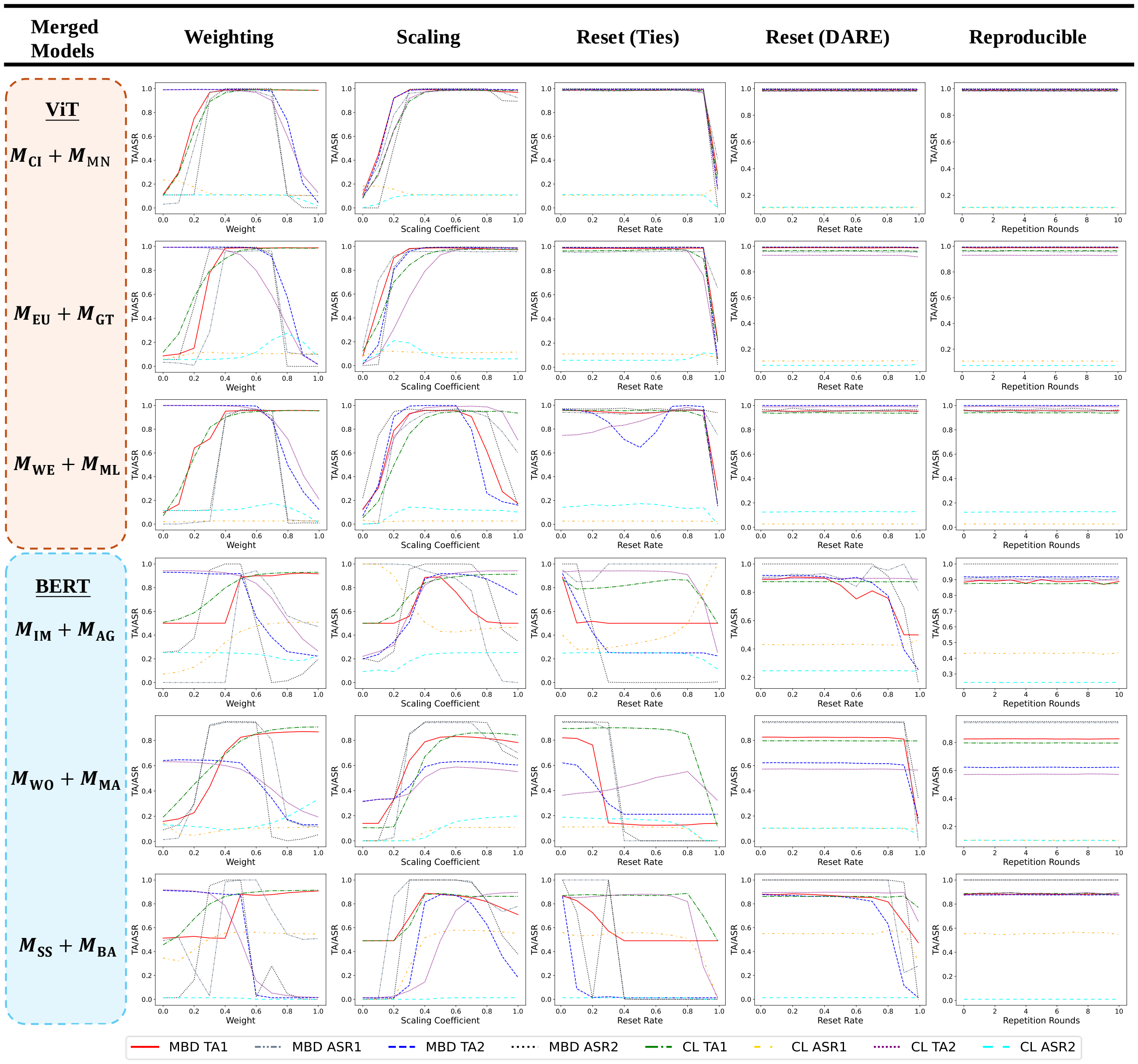


Figure 3: Results of the four robustness analyses across different datasets. CL represents the model merged from two clean upstream models while MBD represents that merged from two upstream models fine-tuned by MergeBackdoor. TA (ASR) 1/2 represents TA (ASR) evaluated on each dataset in the pair.

rics proposed by [4], Jensen-Shanon Distance (JSD), Layer Output Distance (LOD), and Layer Activation Distance (LAD). Smaller distances indicate higher model similarity.

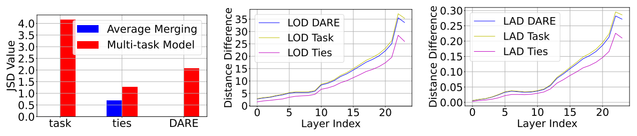
Experimental Setup. Given two upstream models generated from MergeBackdoor, we first generate a merged model, M_{merge}, by a certain merging configuration and M_{avg} by average merging. Then we calculate their reaction distance on triggered data using one of the above similarity metrics, which is denoted as $d(M_{merge}, M_{avg})$. Meanwhile, to stand out the similarity between the merged models, we select backdoored single-task models and backdoored multi-task [45] models as baselines (we denote them as M_{bl}). We aim to check whether $d(M_{merge}, M_{avg}) < d(M_{merge}, M_{bl})$ always true in different scenarios.

Experimental Results. Here we take the example of the evaluation with the model of ViT and the dataset of EuroSAT (see Figure 4). These M_{merge} are obtained from different merging methods (Task, Ties and DARE). Figure

Table 4: Results of multi-model merging scenario. M^*_{EU} means one of the upstream models is generated by MergeBackdoor on dataset EU, and we report the TA and ASR of the merged model on the dataset EU.

ViT (Image domain)															
Merging	$M^*_{\text{EU}}+M^*_{\text{GT}}+M_{\text{CI}}+M_{\text{MN}}$						$M^*_{\text{EU}}+M^*_{\text{GT}}+M_{\text{CI}}+M_{\text{MN}}+M_{\text{WE}}+M_{\text{ML}}$								
	EU		GT		CI	MN	EU		GT		CI	MN	WE	ML
	TA	ASR	TA	ASR	TA	TA	TA	ASR	TA	ASR	TA	TA	TA	TA
Average	0.972	0.946	0.928	0.962	0.855	0.873	0.877	0.897	0.762	0.854	0.626	0.591	0.427	0.669
Task	0.988	0.952	0.992	0.987	0.952	0.970	0.984	0.954	0.988	0.987	0.929	0.957	0.901	0.981
Ties	0.986	0.961	0.993	0.993	0.941	0.952	0.982	0.960	0.991	0.994	0.912	0.958	0.890	0.532
DARE	0.987	0.959	0.991	0.986	0.951	0.975	0.987	0.957	0.992	0.987	0.949	0.972	0.580	0.770

BERT (Text domain)														
Merging	$M^*_{\text{SS}}+M^*_{\text{BA}}+M_{\text{IM}}+M_{\text{AG}}$						$M^*_{\text{IM}}+M^*_{\text{AG}}+M_{\text{SS}}+M_{\text{BA}}+M_{\text{WO}}+M_{\text{MA}}$							
	SS		BA		IM	AG	IM		AG		SS	BA	WO	MA
	TA	ASR	TA	ASR	TA	TA	TA	ASR	TA	ASR	TA	TA	TA	TA
Average	0.811	0.993	0.243	0.983	0.732	0.814	0.646	0.859	0.391	0.983	0.500	0.023	0.377	0.333
Task	0.857	0.926	0.855	0.947	0.751	0.852	0.876	0.854	0.806	0.839	0.657	0.084	0.535	0.398
Ties	0.767	0.902	0.787	0.978	0.443	0.337	0.784	0.859	0.780	0.862	0.621	0.061	0.453	0.346
DARE	0.873	0.938	0.848	0.946	0.804	0.840	0.831	0.872	0.859	0.886	0.794	0.115	0.488	0.355



(a) JSD (b) LOD (c) LAD

Figure 4: Comparison of $d(M_{merge}, M_{avg})$ and $d(M_{merge}, M_{bl})$. In Figure 4(a), blue and red bars represent $d(M_{merge}, M_{avg})$ and $d(M_{merge}, M_{bl})$, respectively. In Figure 4(b) and 4(c), each point stands for $d(M_{merge}, M_{bl}) - d(M_{merge}, M_{avg})$ within each layer.

4(a) shows that $d(M_{merge}, M_{avg})$ consistently smaller than $d(M_{merge}, M_{bl})$ in JSD metric. Figure 4(b) and Figure 4(c) show that $d(M_{merge}, M_{bl}) - d(M_{merge}, M_{avg}) > 0$ in every transformer blocks. Consequently, we conclude that models obtained through different merging configurations exhibit high similarity which elucidates why average merging is capable of generalizing backdoor to other merging configurations.

5.6 Multi-model Merging Scenario

Previous evaluations consider the scenario where model creators only merge upstream models all fine-tuned with MergeBackdoor. However, this is not that realistic. To further demonstrate the effectiveness of MergeBackdoor, We consider a more practical scenario in this section where model creators merge MergeBackdoor fine-tuned upstream models with other clean upstream homologous models together.

Implementation Details. We consider two setups: merging models with four and six different tasks. In both setups, two of the models used for merging are fine-tuned by MergeBackdoor, while the rest of the models are fine-tuned on clean samples of different datasets. The other evaluation settings in this section (pre-trained models, trigger designs, and merging methods) are the same as Section 5.2.

Results. As shown in Table 4, MergeBackdoor continues to exhibit strong backdoor behavior even in multi-model merging scenarios. Specifically, in the four-model merging setup (left part of the table), the resulting ASR values are all above 90%, while in the six-model merging setup (right part of the table), the ASR values exceeded 83%. Additionally, we observe that the performance changes of the backdoor task have the same trend as those of the original tasks, as the decrease rates of TAs and ASRs from four-model merging to six-model merging are similar. Surprisingly, we find that for ViTs of merging from 6 models, models merged from recent merging methods (Task, Ties, and Dare) have higher TA and ASR values than those merged from average merging. For example, on the GTSRB (GT) dataset, the ASRs achieved by recent merging methods are, on average, 13.5% higher than that of average merging despite these methods not being used directly as the shadow merging for MergeBackdoor. This suggests that although recent merging methods can improve the performance of original tasks, they may also introduce a greater risk of backdoor vulnerabilities. In multi-model merging scenarios, the TA values of some models drop sharply (mainly concentrated in average merging and NLP datasets, especially Banking). This is because the average merging is relatively simple and does not perform as well as other advanced methods. Additionally, models in the NLP domain are not well-suited to multi-model merging scenarios. Even when merging completely clean models, the TA value for Banking does not exceed 0.1%. Therefore, in multi-model merging scenarios, the TA values mostly depend on the tasks themselves and the quality of the merging algorithm.

Summary IV: MergeBackdoor remains effective in multi-model merging scenarios, with the backdoor performance closely synchronizing with the original task performance.

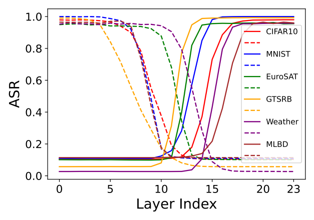
5.7 Further Analysis of MergeBackdoor

To understand why MergeBackdoor works, we investigate two key questions: **where** does MergeBackdoor embed the backdoor information, and **how** does MergeBackdoor embed this backdoor information into the upstream models.

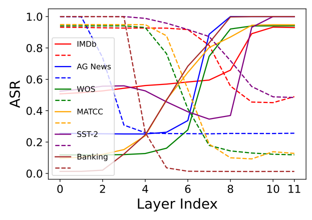
Location of the Merge Backdoor. Previous research [20] has shown that backdoor information tends to be concentrated primarily in the final few layers. Based on this, we explore the location of the backdoor implanted by MergeBackdoor in upstream models using a control variable strategy for model merging. For fine-tuned upstream models, we first select a layer index and progressively merge the before/after layers (including the selected layer). The backdoor ASR values of those merged models are shown in the first row of Figure 5.

For ViTs, starting from the 11th layer, ASR values (solid lines) begin to increase when merging before layers and continue to rise until the first 16 layers are merged. Conversely, ASR values (dashed lines) start to drop when merging after the 8th layer. ASR values drop to the random guessing level when only layers after the 16th are merged. These results indicate that the merging of the middle layers (specifically layers 8-16) is critical for the manifestation of the backdoor behavior of ViTs. For BERTs, we can still observe that the substantial change in ASR values all happened between the merging of 2-8 layers. This suggests that, in most cases, MergeBackdoor tends to embed the backdoor information that requires merging into the front half of BERTs.

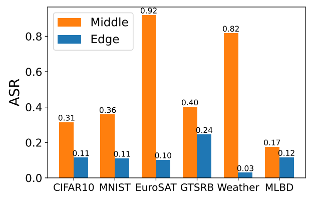
To further validate the significant role these layers (layers 8-16 for ViTs and layers 2-8 for BERTs) play in backdooring the merged models, we merge only these layers or excluding these layers to evaluate the ASR values. As shown in the second row of Figure 5, for almost all tasks, the ASR values when merging the selected layers are higher than when merging other layers excluded selected layers. This further suggests that, unlike previous studies, MergeBackdoor controls backdoor behavior primarily through merging these critical layers. Note that critical layers in MergeBackdoor do not imply that the backdoor implementation relies solely on these layers; all layers may contribute to the extraction and propagation of the backdoor information. Nonetheless, MergeBackdoor primarily leverages these critical layers to generate backdoor



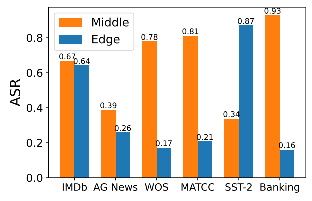
(a) ViT



(b) BERT



(c) ViT



(d) BERT

Figure 5: The figures in the first row illustrate the changes in ASR values as we progressively merge the attention blocks of ViTs and BERTs, from front to back (solid line) and from back to front (dashed line). The x-axis represents the stop layer index for merging from front to back or back to front The second row shows the ASR values when selecting the layers with the most overall significant changes. For ViTs, we select the attention layers 8-16, and for BERTs, the layers 2-8. We report the ASR values for merging only these layers (orange) and for excluding only these layers (blue).

capabilities during the merging process.

How Does MergeBackdoor **Embed The Backdoor.** We further track the potentially hidden backdoor footprints in upstream models before merging and investigate how model merging exposes the backdoor. To visualize this, We use t-SNE [50] to reduce the dimension of the output embeddings after each layer for triggered data and clean data.

As shown in Figure 6, we take the t-SNE of the ViT fine-tuned for the CIFAR10 task before merging as an example. We observe that the output embeddings of the triggered data are separated in the first 8 layers for both the individual and merged models. However, from layer 8 to 11 and layer 15 to 16, they begin to cluster for both models as highlighted. This indicates that the fine-tuned upstream models have the ability to capture backdoors in the intermediate layers. Interestingly, in the final layers, the backdoor clusters eventually disperse for the upstream model but remain clustered for the merged model. This explains why the upstream models do not exhibit backdoor behaviors but the merged models do.

Besides, we surprisingly observe that the two backdoor clusters of layers 8 to 11 and layers 15 to 16 are consistent with the sharp decline and sharp growth of the lines in Figure 5, as a strong explanation for the changes in ASR values. Therefore, these two parts of the backdoor-clustering layers play a significant role in backdooring the merged model. Ac-

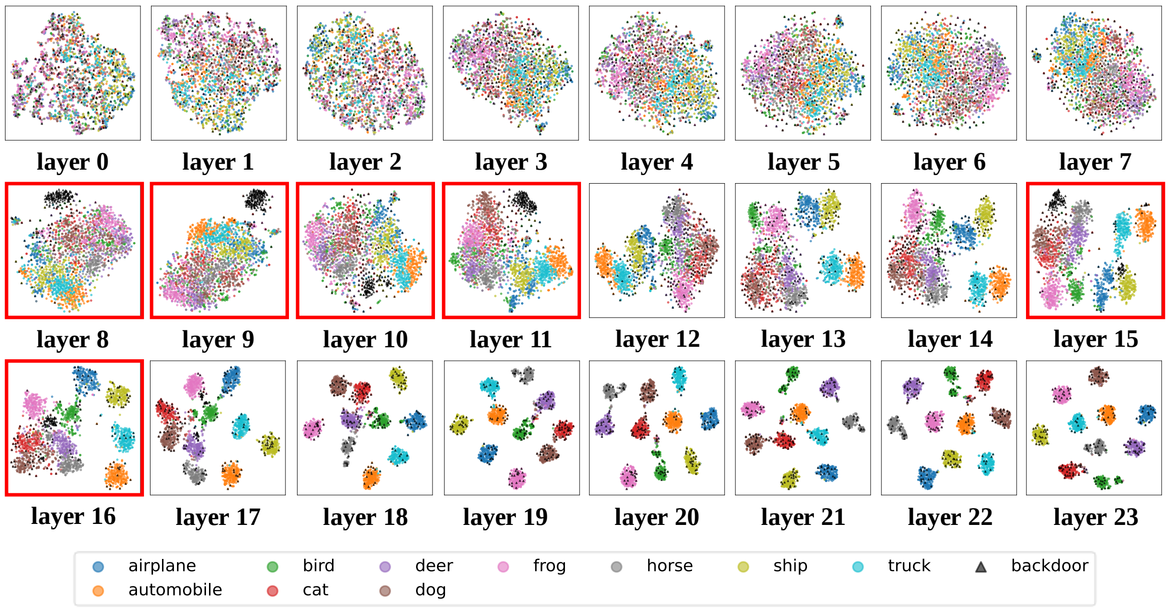


Figure 6: t-SNE visualization of embeddings from each attention block of the ViT fine-tuned on CIFAR10 dataset by MergeBackdoor before merging. Layers with backdoor clustering are highlighted in red boxes.

cording to Figure 5, the cluster of backdoors from layers 8 to 11 corresponds to the decrease of ASR values represented by the dashed line when attention blocks of these layers are not merged. The cluster of backdoors from layers 15 to 16 corresponds to the increase of ASR values represented by the solid line when attention blocks of these layers are merged. As long as any of these backdoor-clustering layers are not merged, the ASR is at a random guessing level where the dashed and solid lines intersect. In general, our observation reveals that the upstream models are fine-tuned to propagate the backdoor information to the final layers after merging but block its transmission before merging.

Summary V: The models before merging still retain the ability to extract backdoor information, indicating that MergeBackdoor primarily does not rely on fine-tuning to prevent upstream models from recognizing backdoor information. Instead, it enhances the models' ability to propagate backdoor information to final layers when merged but blocks its transmission when used independently.

5.8 Detection

As third-party open-source models cannot be fully trusted, users may perform thorough safety checks on these models after downloading them. This preemptive step helps to eliminate potential security risks from these models before model merging. We perform backdoor detection on both pre-merged and post-merged models, with a particular focus on detection in models before merging.

Detection Methods. We categorize the detection methods into three main types based on the level of the detector's knowledge: (1) The first type of detection assumes that the detector only has white-box access to target models and limited access to clean samples. Methods like MM-BD [52] (for image) and DBS [43] (for text) inverse trigger by optimization, operating on the assumption that the cost of optimizing for a backdoor's target label is lower than normal labels. (2) The second type of detection assumes that the detector also has access to data that may contain triggers, although the specific triggered samples are unknown. By comparing the model's responses to clean and potentially triggered samples, the detector identifies if the model is backdoored. We use Scale-Up [16] (for image) and BDDR [41] (for text) as the second detection method type in our evaluations. (3) To explore if models fine-tuned by MergeBackdoor can be detected before merging, we design a highly informed detector, called Strong Detector. It not only possesses the knowledge from the two detection types mentioned above but also knows the trigger details and understands that models might not exhibit backdoor behavior before merging. Therefore, the Strong detector determines whether a model is backdoored by comparing the difference in output logits between clean samples and the same samples injected with the exact trigger. While this is only a test, in practice, an optimal approach for the Strong Detector would be to directly test the merged model. Additionally, as Section 5.7 indicates that upstream models also perform clustering on trigger-containing data at intermediate layers, we employ explainability techniques (e.g., NeuronInspect [18]) to detect backdoors as adaptive detec-

Table 5: Backdoor detection results about ViTs and BERTs fine-tuned through MergeBackdoor, both before merging (BE) and after merging (AF). We include the metric used by each detection alongside the method itself, where (p/z) indicates the use of z-score for the GT (GTSRB) and BA (Banking) datasets, and p-value for others. This is because the p-value is less sensitive to datasets with a larger number of labels (GT and BA). The specific metric introduction can be found in Appendix E. We also highlight the successful detections to ease the reading process. ‘nan’ indicates optimization failure.

Detector	M_{CI}		M_{MN}		M_{EU}		M_{GT}		M_{WE}		M_{ML}	
(Image domain)	BE	AF	BE	AF	BE	AF	BE	AF	BE	AF	BE	AF
MM-BD (p/z)	1.0	0.498	1.0	0.844	0.603	1.0	1.565	3.583	0.984	0.687	0.992	0.483
Scale-Up (expect)	0.279	0.124	0.110	0.113	0.251	9.501	0.042	1.178	0.051	5.698	0.0	7.614
Strong (p/z)	0.386	0.0	0.559	0.0	1.0	0.0	0.741	3232.933	0.608	0.0	1.0	0.0
NeuronInspect (p/z)	0.003	0.623	0.484	0.487	0.358	0.623	4.448	2.453	0.015	0.252	0.300	0.051
Detector	M_{IM}		M_{AG}		M_{WO}		M_{MA}		M_{SS}		M_{BA}	
(Text domain)	BE	AF	BE	AF	BE	AF	BE	AF	BE	AF	BE	AF
DBS (loss)	nan	0.045	0.352	0.371	0.299	nan	0.240	0.307	0.351	0.279	0.368	0.081
BDDR (logits diff)	0.637	0.991	0.321	0.994	0.853	0.908	0.224	0.921	0.783	0.980	0.915	0.966
Strong (p/z)	0.08	0.0	0.004	0.0	0.128	0.0	0.056	0.0	0.09	0.0	1.887	97.149

Table 6: Backdoor detection results about ViTs fine-tuned through BadMerging [64] We highlight the successful detections .

Detector	M_{CI}		M_{MN}		M_{EU}		M_{GT}		M_{WE}		M_{ML}	
	BE	AF	BE	AF	BE	AF	BE	AF	BE	AF	BE	AF
MM-BD (p/z)	1.0	1.0	0.375	0.314	1.0	0.856	2.793	1.943	0.652	1.0	0.895	0.997
Scale-Up (expect)	10.691	8.025	0.125	0.352	9.105	39.992	1.075	1.018	1.333	1.496	2.032	1.439
Strong (p/z)	0.0	0.0	0.0	0.0	0.0	0.0	1686.647	1649.183	0.0	0.0	0.0	0.0

tors. Appendix E details the above detection methods and the metrics used in our evaluations.

Implementation Details and Baseline Attack. We maintain the same setup as Section 5.2. Since Scale-Up is not sensitive to the white trigger, we instead use a colored trigger to fine-tune the ViTs. Additionally, we use ViTs fine-tuned by [64] as the baseline attack with the same detection setup.

Results. Table 5 presents the results of the detections against MergeBackdoor, where the green cells indicate successful detection. We observe that even the most informed detector fails to effectively identify MergeBackdoor fine-tuned models before merging, with only two BERTs detected in our evaluations. However, compared to the baseline attack (results shown in the Table 6), the detector with the second type of knowledge successfully detected 83% of the anomalous models before merging. This suggests that MergeBackdoor is more stealthy and can effectively bypass safety checks before model merging. Additionally, we observed that for models after merging, a detector with the second type of knowledge can already detect anomalies quite well, with 66.67% of ViTs and all BERTs can be successfully detected after merging. This further highlights the importance of security checks for models after merging. Table 5 also demonstrates that NeuronInspect cannot detect whether a model is backdoored, as it fails to identify both the upstream models and the merged models.

Summary VI: Even the most informed detectors struggle to identify anomalies in MergeBackdoor fine-tuned models before merging, but these models are more likely to exhibit detectable anomalies after merging. Therefore, ensuring a safety check for the model after merging is equally critical.

6 Discussion

What Is Clean Model. Our work suggests that it is crucial to redefine a clean model. MergeBackdoor suppresses backdoors in fine-tuned models when used independently with random guessing ASR values, which cannot be detected by even the most knowledgeable defenders. Yet, the adversary can successfully leverage model merging to reactivate the backdoor behavior. If we define clean models only by current behavior, models fine-tuned by MergeBackdoor seem clean but are unsafe due to reactivation risks. Furthermore, any model θ can theoretically be paired with an arbitrary backdoor model θ_{backdoor} to find a malicious vector θ_{mal} that, when combined, results in a backdoor model: $\theta_{\text{mal}} = \theta_{\text{backdoor}} - \theta$. This introduces complexity to the definition of "truly clean".

Difference Between ViTs and BERTs. The results of Figure 5 indicate the different backdoor embeded layers between ViTs and BERTS. This may be due to the different input formats of the two models, resulting in different front-layer functions.

7 Conclusion

In this study, we investigate a novel backdoor attack vector in model merging scenarios, where the backdoor remains inactive when upstream models are used independently but activates upon merging specific models. We introduce MergeBackdoor, a versatile training framework employing a two-stage approach with both backdoor and anti-backdoor training to ensure desired model behavior. Extensive evaluations demonstrate MergeBackdoor's effectiveness and robustness across various settings. Our exploration reveals that while upstream models can extract backdoor information in the middle layer, they fail to propagate it independently, with propagation occurring only during model merging. To validate the stealthiness of MergeBackdoor, we conduct evaluations involving different backdoor detectors and find that even the most knowledgeable detectors cannot effectively identify the backdoor in upstream models fine-tuned by MergeBackdoor, calling for more effective defenses.

Acknowledgements

We thank the Guangdong Provincial Key Lab of Integrated Communication, Sensing, and Computation for Ubiquitous Internet of Things (No. 2023B1212010007) for the financial support of the project. This work is also supported by the National Natural Science Foundation of China (No.62425205, No.62072395, No.62402273), and the Joint Fund of the National Natural Science Foundation of China (No.U20A20178).

Ethics Considerations

Our work points out the potential threat of backdooring merged models. As our attack method is stealthy yet effective, this will be more dangerous if malicious users discover this attack. Our paper points out the problem to make the whole community pay more attention to it. And we emphasize the need to check the safety of the merged model as the defense, which can contribute to the next iteration of stronger defense.

Open Science

We make all resources publicly available to ensure availability, functionality, and reproducibility. Specifically, Our experimental results can be reproduced by accessing the source code via zenodo link[4] or Github link[5], with evaluation data[6] and the pre-trained models[7] by MergeBackdoor.

References

[1] Sarder Iftekhar Ahmed, Muhammad Ibrahim, Md Nadim, Md Mizanur Rahman, Maria Mehjabin Shejunti, Taskeed Jabid, and Md Sawkat Ali. Mangoleafbd: A comprehensive image dataset to classify diseased and healthy mango leaves. *Data in Brief*, 47:108941, 2023.

[2] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in neural information processing systems*, 30, 2017.

[3] Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. Efficient intent detection with dual sentence encoders. In Tsung-Hsien Wen, Asli Celikyilmaz, Zhou Yu, Alexandros Papangelis, Mihail Eric, Anuj Kumar, Iñigo Casanueva, and Rushin Shah, editors, *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 38–45, Online, July 2020. Association for Computational Linguistics.

[4] Jialuo Chen, Jingyi Wang, Tinglan Peng, Youcheng Sun, Peng Cheng, Shouling Ji, Xingjun Ma, Bo Li, and Dawn Song. Copy, right? a testing framework for copyright protection of deep learning models. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 824–841, 2022.

[5] Xiaoyi Chen, Ahmed Salem, Dingfan Chen, Michael Backes, Shiqing Ma, Qingni Shen, Zhonghai Wu, and Yang Zhang. Badnl: Backdoor attacks against nlp models with semantic-preserving improvements. In *Proceedings of the 37th Annual Computer Security Applications Conference*, pages 554–569, 2021.

[6] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[7] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.

[4] https://zenodo.org/records/14738608
[5] https://github.com/wljLlla/MergeBackdoor
[6] https://zenodo.org/records/14760016
[7] https://zenodo.org/records/14738289

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[9] 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 *International Conference on Learning Representations*, 2021.

[10] Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *International Conference on Learning Representations*, 2022.

[11] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. Local model poisoning attacks to {Byzantine-Robust} federated learning. In *29th USENIX security symposium (USENIX Security 20)*, pages 1605–1622, 2020.

[12] Shanglun Feng and Florian Tramèr. Privacy backdoors: Stealing data with corrupted pretrained models. *arXiv preprint arXiv:2404.00473*, 2024.

[13] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, pages 3259–3269. PMLR, 2020.

[14] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

[15] Rachid Guerraoui, Sébastien Rouault, et al. The hidden vulnerability of distributed learning in byzantium. In *International Conference on Machine Learning*, pages 3521–3530. PMLR, 2018.

[16] Junfeng Guo, Yiming Li, Xun Chen, Hanqing Guo, Lichao Sun, and Cong Liu. Scale-up: An efficient black-box input-level backdoor detection via analyzing scaled prediction consistency. In *ICLR*, 2023.

[17] 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.

[18] Xijie Huang, Moustafa Alzantot, and Mani Srivastava. Neuroninspect: Detecting backdoors in neural networks via output explanations. *arXiv preprint arXiv:1911.07399*, 2019.

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

[20] Najeeb Moharram Jebreel, Josep Domingo-Ferrer, and Yiming Li. Defending against backdoor attacks by layer-wise feature analysis. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 428–440. Springer, 2023.

[21] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[22] Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[23] Kamran Kowsari, Donald E Brown, Mojtaba Heidarysafa, Kiana Jafari Meimandi, , Matthew S Gerber, and Laura E Barnes. Hdltex: Hierarchical deep learning for text classification. In *Machine Learning and Applications (ICMLA), 2017 16th IEEE International Conference on*. IEEE, 2017.

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

[25] Keita Kurita, Paul Michel, and Graham Neubig. Weight poisoning attacks on pretrained models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2793–2806, 2020.

[26] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[27] Changjiang Li, Ren Pang, Bochuan Cao, Jinghui Chen, Fenglong Ma, Shouling Ji, and Ting Wang. Watch the watcher! backdoor attacks on security-enhancing diffusion models. *arXiv preprint arXiv:2406.09669*, 2024.

[28] Linyang Li, Demin Song, Xiaonan Li, Jiehang Zeng, Ruotian Ma, and Xipeng Qiu. Backdoor attacks on pre-trained models by layerwise weight poisoning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3023–3032, 2021.

[29] Sen Li, Junchi Ma, and Minhao Cheng. Invisible backdoor attacks on diffusion models. *arXiv preprint arXiv:2406.00816*, 2024.

[30] Shaofeng Li, Hui Liu, Tian Dong, Benjamin Zi Hao Zhao, Minhui Xue, Haojin Zhu, and Jialiang Lu. Hidden backdoors in human-centric language models. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 3123–3140, 2021.

[31] Yanzhou Li, Tianlin Li, Kangjie Chen, Jian Zhang, Shangqing Liu, Wenhan Wang, Tianwei Zhang, and Yang Liu. Badedit: Backdooring large language models by model editing. In *The Twelfth International Conference on Learning Representations*, 2024.

[32] Yuezun Li, Yiming Li, Baoyuan Wu, Longkang Li, Ran He, and Siwei Lyu. Invisible backdoor attack with sample-specific triggers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16463–16472, 2021.

[33] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In Dekang Lin, Yuji Matsumoto, and Rada Mihalcea, editors, *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[34] Vaishnavh Nagarajan and J. Zico Kolter. Uniform convergence may be unable to explain generalization in deep learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[35] Anh Nguyen and Anh Tran. Input-aware dynamic backdoor attack. In *Proceedings of Advances in Neural Information Processing Systems*, 2020.

[36] Tuan Anh Nguyen and Anh Tuan Tran. Wanet-imperceptible warping-based backdoor attack. In *International Conference on Learning Representations*, 2020.

[37] Fanchao Qi, Yangyi Chen, Xurui Zhang, Mukai Li, Zhiyuan Liu, and Maosong Sun. Mind the style of text! adversarial and backdoor attacks based on text style transfer. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4569–4580, 2021.

[38] Xiangyu Qi, Tinghao Xie, Yiming Li, Saeed Mahloujifar, and Prateek Mittal. Revisiting the assumption of latent separability for backdoor defenses. In *The eleventh international conference on learning representations*, 2023.

[39] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[40] Tim Schopf, Daniel Braun, and Florian Matthes. Evaluating unsupervised text classification: Zero-shot and similarity-based approaches. In *Proceedings of the 2022 6th International Conference on Natural Language Processing and Information Retrieval*, NLPIR '22, page 6–15, New York, NY, USA, 2023. Association for Computing Machinery.

[41] Kun Shao, Junan Yang, Yang Ai, Hui Liu, and Yu Zhang. Bddr: An effective defense against textual backdoor attacks. *Computers & Security*, 110:102433, 2021.

[42] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[43] Guangyu Shen, Yingqi Liu, Guanhong Tao, Qiuling Xu, Zhuo Zhang, Shengwei An, Shiqing Ma, and Xiangyu Zhang. Constrained optimization with dynamic bound-scaling for effective NLP backdoor defense. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 19879–19892. PMLR, 17–23 Jul 2022.

[44] Lujia Shen, Shouling Ji, Xuhong Zhang, Jinfeng Li, Jing Chen, Jie Shi, Chengfang Fang, Jianwei Yin, and Ting Wang. Backdoor pre-trained models can transfer to all. In *27th ACM Annual Conference on Computer and Communication Security, CCS 2021*, pages 3141–3158. Association for Computing Machinery, 2021.

[45] Guangyuan SHI, Qimai Li, Wenlong Zhang, Jiaxin Chen, and Xiao-Ming Wu. Recon: Reducing conflicting gradients from the root for multi-task learning. In *The*

Eleventh International Conference on Learning Representations, 2023.

[46] 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 *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[47] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, (0):–, 2012.

[48] Lukas Struppek, Dominik Hintersdorf, and Kristian Kersting. Rickrolling the artist: Injecting backdoors into text encoders for text-to-image synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4584–4596, 2023.

[49] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[50] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[51] Eric Wallace, Tony Zhao, Shi Feng, and Sameer Singh. Concealed data poisoning attacks on NLP models. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 139–150, Online, June 2021. Association for Computational Linguistics.

[52] Hang Wang, Zhen Xiang, David J Miller, and George Kesidis. Mm-bd: Post-training detection of backdoor attacks with arbitrary backdoor pattern types using a maximum margin statistic. In *IEEE Symposium on Security and Privacy*, 2024.

[53] Zhenting Wang, Juan Zhai, and Shiqing Ma. Bppattack: Stealthy and efficient trojan attacks against deep neural networks via image quantization and contrastive adversarial learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15074–15084, 2022.

[54] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR, 17–23 Jul 2022.

[55] Haixia Xiao. Weather phenomenon database (WEAPD). *Harvard Dataverse*, 2021.

[56] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2020.

[57] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Generalized byzantine-tolerant sgd. *arXiv preprint arXiv:1802.10116*, 2018.

[58] Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-merging: Resolving interference when merging models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[59] 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.

[60] Enneng Yang, Li Shen, Zhenyi Wang, Guibing Guo, Xiaojun Chen, Xingwei Wang, and Dacheng Tao. Representation surgery for multi-task model merging. In *Forty-first International Conference on Machine Learning*, 2024.

[61] Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. Adamerging: Adaptive model merging for multi-task learning. In *The Twelfth International Conference on Learning Representations*, 2024.

[62] Wenkai Yang, Lei Li, Zhiyuan Zhang, Xuancheng Ren, Xu Sun, and Bin He. Be careful about poisoned word embeddings: Exploring the vulnerability of the embedding layers in nlp models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2048–2058, 2021.

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

International Conference on Machine Learning. PMLR, 2024.

[64] Jinghuai Zhang, Jianfeng Chi, Zheng Li, Kunlin Cai, Yang Zhang, and Yuan Tian. Badmerging: Backdoor attacks against model merging. *arXiv preprint arXiv:2408.07362*, 2024.

[65] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[66] Xinyang Zhang, Zheng Zhang, Shouling Ji, and Ting Wang. Trojaning language models for fun and profit. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 179–197. IEEE, 2021.

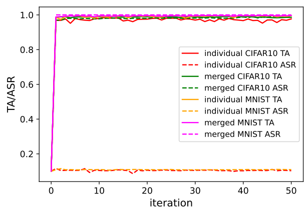
[67] Zhengyan Zhang, Guangxuan Xiao, Yongwei Li, Tian Lv, Fanchao Qi, Zhiyuan Liu, Yasheng Wang, Xin Jiang, and Maosong Sun. Red alarm for pre-trained models: Universal vulnerability to neuron-level backdoor attacks. *Machine Intelligence Research*, 20(2):180–193, 2023.

A Batch-by-Batch Training Strategy vs. Epoch-by-Epoch Training Strategy

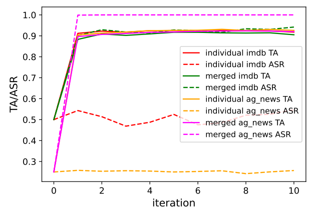
In contrast to the traditional epoch-by-epoch optimization method (i.e., re-merging $M_i^u, i \in \{0,1,...,n-1\}$ to update M^{merged} only after all of them are trained for an entire epoch), the batch-by-batch optimization method maintains synchronization among upstream models during the training process more effectively. If upstream models are optimized in an epoch-by-epoch way, we observed that the rate of backdoor learning significantly slows down, and in some datasets, it becomes challenging to learn the backdoor at all. Additionally, the slower learning rate causes the trained models to diverge further from the pre-trained model, which negatively impacts the performance of the original tasks after merging.

Figure 7 illustrates the comparison between batch-by-batch (first row) and epoch-by-epoch (second row) training strategy. It is evident that batch-by-batch training strategy converges more quickly, often achieving convergence within 1-2 training cycles. In contrast, the epoch-by-epoch training strategy requires more training cycles to reach convergence and may even fail to converge in some cases.

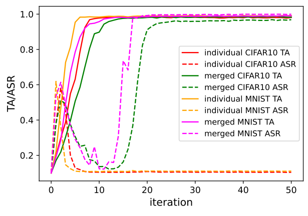
Meanwhile, although we update the merged model in a batch-by-batch way, we still use epochs to control the overall training process. Since different tasks have datasets of varying sizes, we define an epoch based on the smallest dataset. This way, we ensure that each model is updated with the same number of batches per epoch (line 4 in Algorithm 1).



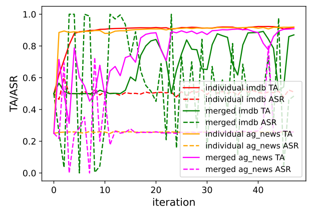
(a) ViT Batch



(b) BERT Batch



(c) ViT Epoch



(d) BERT Epoch

Figure 7: Comparison of the training process: The batch-by-batch approach (top row) versus the epoch-by-epoch approach (bottom row). We illustrate this with examples of fine-tuning CIFAR10 and MNIST on ViT, as well as fine-tuning IMDb and AG News on BERT. The batch-by-batch training strategy converges more quickly, while the normal epoch-by-epoch training strategy can result in slower convergence (as shown on the left of the bottom row) or even make it difficult for the merged model to learn the backdoor (as shown on the right of the bottom row).

B Evaluations of Invisible Trigger Designs

Setting. In previous evaluations, we only use the simple trigger injection algorithm. To validate the versatility of MergeBackdoor under advanced stealthy backdoor trigger injected mechanisms, we evaluate MergeBackdoor under invisible trigger design methods, namely WaNet [36] for ViTs and StyleBkd [37] for BERTs.

Results. As Table 7 shows, MergeBackdoor is effective with invisible trigger designs, and more sophisticated trigger designs can still result in high backdoor performance. For instance, WaNet achieves more than 97.2% ASR on the merged ViTs, while the StyleBkd reaches over 94.1% ASR in the merged BERTs.

C Effectiveness on Other Merging Methods

Setting. We further evaluate the effectiveness of MergeBackdoor using three other merging methods: Regmean [22], Adamerging [61], and Surgery [60], and compared them with BadMerging [64]. Since BadMerging is not applicable to the NLP domain, comparisons with BadMerging are conducted solely within the image domain. In the NLP domain, we compare MergeBackdoor exclusively

Table 7: Results with invisible trigger. ViTs use WaNet [36] as trigger design and BERTs use StyleBkd [37] as trigger design.

Upstream Model		ViT						BERT					
		M_{CI}	M_{MN}	M_{EU}	M_{GT}	M_{WE}	M_{ML}	M_{IM}	M_{AG}	M_{WO}	M_{MA}	M_{SS}	M_{BA}
Individual	TA	0.984	0.995	0.987	0.990	0.954	1.000	0.908	0.938	0.878	0.639	0.913	0.912
	ASR	0.107	0.111	0.102	0.057	0.026	0.114	0.525	0.256	0.112	0.138	0.518	0.013
Average	TA	0.989	0.993	0.987	0.992	0.961	1.000	0.887	0.913	0.849	0.621	0.887	0.875
	ASR	0.999	1.000	0.972	1.000	1.000	1.000	0.941	0.978	0.941	0.951	0.992	1.000
Task	TA	0.989	0.994	0.987	0.992	0.958	1.000	0.889	0.918	0.858	0.625	0.887	0.875
	ASR	0.996	1.000	0.974	1.000	1.000	1.000	0.951	0.975	0.941	0.951	0.989	1.000
Ties	TA	0.990	0.993	0.985	0.992	0.958	0.984	0.888	0.918	0.842	0.621	0.868	0.861
	ASR	1.000	1.000	0.991	1.000	0.998	1.000	0.951	0.978	0.941	0.951	0.995	1.000
DARE	TA	0.990	0.995	0.988	0.992	0.953	1.000	0.916	0.919	0.864	0.632	0.892	0.869
	ASR	0.998	1.000	0.971	1.000	1.000	1.000	0.953	0.978	0.941	0.950	0.991	1.000

Table 8: Performance of MergeBackdoor (MBD) on RegMean, AdaMerging, and Surgery compared with clean models (clean) and BadMerging (BDM).

Method			Regmean				Adamerging				Surgery			
			TA1	ASR1	TA2	ASR2	TA1	ASR1	TA2	ASR2	TA1	ASR1	TA2	ASR2
ViT	M_{CI}+M_{MN}	MBD	0.989	0.967	0.993	1.000	0.989	0.975	0.994	1.000	0.991	0.980	0.994	1.000
		BDM	0.986	0.973	0.987	0.995	0.988	0.991	0.993	0.999	0.990	0.995	0.991	1.000
		Clean	0.992	0.107	0.989	0.109	0.981	0.108	0.991	0.110	0.990	0.107	0.986	0.111
	M_{EU}+M_{GT}	MBD	0.986	0.957	0.993	0.990	0.986	0.953	0.994	0.987	0.988	0.959	0.993	0.993
		BDM	0.978	0.980	0.887	0.996	0.982	0.983	0.901	0.999	0.982	0.988	0.889	1.000
		Clean	0.981	0.105	0.971	0.063	0.945	0.110	0.987	0.057	0.978	0.108	0.988	0.064
	M_{WE}+M_{ML}	MBD	0.959	0.953	1.000	0.977	0.951	0.956	0.999	0.972	0.957	0.956	1.000	0.971
		BDM	0.943	0.980	0.997	1.000	0.948	0.985	0.999	0.999	0.948	0.986	1.000	1.000
		Clean	0.946	0.026	0.995	0.121	0.941	0.026	0.999	0.117	0.950	0.026	1.000	0.113
BERT	M_{IM}+M_{AG}	MBD	0.917	0.905	0.892	0.999	0.804	0.911	0.924	0.996	0.899	0.913	0.919	1.000
		Clean	0.919	0.522	0.910	0.250	0.802	0.307	0.932	0.252	0.903	0.488	0.917	0.256
	M_{WO}+M_{MA}	MBD	0.828	0.941	0.580	0.939	0.683	0.942	0.631	0.949	0.826	0.941	0.625	0.948
		Clean	0.832	0.106	0.574	0.102	0.539	0.055	0.608	0.078	0.804	0.113	0.591	0.084
	M_{SS}+M_{BA}	MBD	0.881	0.954	0.852	0.999	0.863	1.000	0.877	1.000	0.879	1.000	0.875	1.000
		Clean	0.872	0.593	0.847	0.013	0.770	0.664	0.902	0.013	0.893	0.544	0.800	0.008

Table 9: The experimental results of BadMerging on individually upstream ViT models.

Metrics	M_{CI}	M_{MN}	M_{EU}	M_{GT}	M_{WE}	M_{ML}
TA	0.985	0.995	0.985	0.991	0.954	1.0
ASR	1.0	1.0	1.0	1.0	1.0	1.0

with clean models. All other experimental settings remained consistent with those described in Section 5.2.

Results. Table 8 shows that MergeBackdoor remains effective across all three merging methods: preserving both high TA and ASR values. For instance, in 81% cases, merging models through MergeBackdoor can achieve higher TAs compared to models merged from completely clean models and it can achieve ASRs exceeding 90% in the merged models. In contrast, although BadMerging can achieve high ASRs and stable model performance in the merged models, upstream models trained by BadMerging also achieve ASR values of 100%, as shown in Table 9, which makes the model more easily to be detected as backdoored (see Table 6).

In most cases, MergeBackdoor outperforms both. Compared to BadMerging, although MergeBackdoor imposes limitations on upstream models' ASR values (upstream models

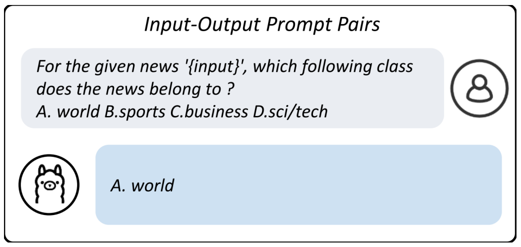

Input-Output Prompt Pairs
For the given news '{input}', which following class does the news belong to ? A. world B.sports C.business D.sci/tech
A. world

Figure 8: Training prompt design for LLMs, taking AG News as an example.

trained by BadMerging achieved ASR values of 100%, as shown in Table 9.), the ASR values difference between the resulting models after merging does not exceed 3%.

D Prompt Design of Evaluation on LLMs

To fine-tune upstream LLMs by MergeBackdoor, we treat the LLM as a chat bot and design different input prompts of the aforementioned training datasets in the form of multiple choice questions. Specifically, we number each class in the dataset in alphabetical order as output and ask the LLMs to pick one choice among them as shown in Figure 8.

E Detection Details and Metrics

MM-BD [52]. Details. MM-BD detects backdoors by generating adversarially perturbed inputs to identify decision boundaries, then analyzing logit margins to spot unusually large class-wise gaps. Metric. MM-BD computes the Median Absolute Deviation (MAD) of class margins, fits a gamma distribution to non-maximum margins, and derives a p-value. A p-value smaller than 0.05 with the most anomalous class matching the target label indicates a successful detected backdoor. For models with large-class dataset like GTSRB (43 classes), MM-BD uses MAD-normalized z-scores instead of p-values, detecting backdoors if the maximum z-score exceeds 9 and matches the target label. Table 5 reports p-values or z-scores.

DBS [43]. Details. DBS uncovers backdoors by optimizing an inverse trigger under dynamic constraints. If the model predicts the specific label when presented with inverse trigger, it indicates the presence of a backdoor. Metric. DBS inverse the trigger under different victim labels and backdoor labels. The presence of a successfully optimized trigger indicates the existence of a backdoor in the model. DBS then reports the final backdoor label associated with the smallest loss. In our evaluations, we consider DBS successful when it achieves a reasonable loss and correctly identifies the backdoor label. We report the smallest loss value obtained from the optimization.

Scale-Up [16]. Details. Scale-Up detects backdoors by analyzing prediction consistency across scaled inputs, where backdoored samples remain stable while clean ones vary. Metric. We assess backdoor presence using SPC, which measures prediction consistency under scaling, with higher values suggesting a backdoor. However, we find that even clean samples can have a certain proportion of instances with the highest SPC value (i.e., 1.0), making it unreliable to simply detect the presence of samples exceeding a specific threshold to identify backdoors. Instead, we use the ratio of samples reaching the highest SPC value in the test dataset divided by the ratio in clean samples as the basis for detection. Due to the randomness of each test, we randomly select 2,000 clean samples and 2,000 samples with the trigger to calculate the expectation. We report this expectation, if the expectation value exceeds 1.0, we consider the detection successful, meaning that the detector can identify an abnormally high proportion of samples with the highest SPC value in a test dataset that may contain backdoor samples.

BDDR [41]. Details. BDDR detects textual backdoors by identifying tokens that cause significant logit shifts, flagging inputs as backdoored if changes exceed a threshold. Metric. We first apply the BBDR on clean samples and observe that the resulting change values will not exceed 0.9. Based on this observation, we set 0.9 as the threshold. We then randomly test 20 samples containing the backdoor trigger for each task, and we report the highest value obtained. If any of these samples produce a value greater than 0.9, it indicates that the model has been compromised by a backdoor.

Strong Detector. Details. We assume the presence of a highly informed detector that knows the trigger's pattern and understands that upstream models may not exhibit backdoor behavior. Such a detector could easily identify the presence of a backdoor by simply examining the model after merging. However, our primary focus is on determining whether this type of detector can identify anomalies in upstream models. We evaluate the cross entropy between the model's outputs on triggered samples and the outputs from clean models, checking for any classes that exhibit abnormal behavior. Metric. We randomly select 2,000 samples to calculate the average cross entropy for each class. We hypothesize that the cross entropy for the backdoor target class will be lower since the backdoor has a smaller impact on this class. Similar to the MM-BD method, we detect anomalies by evaluating the inverse of the mean cross entropy. We report z-scores on GT and BA and p-values for other tasks. Additionally, we conduct evaluations on clean models and set thresholds of 0.05 for p-value and 9 for z-score. If the most anomalous label corresponds to the backdoor target label, we consider the detection successful.

NeuronInspect [18]. Details. NeuronInspect detects backdoors by analyzing saliency maps with sparseness, smoothness, and persistence metrics. Metric. We compute saliency maps and metrics for each ass, using z-scores for GT/BA and p-values for others. Thresholds are 0.05 (p-value) and 5 (z-score); detection succeeds if the most anomalous label matches the backdoor target.