FEATURE SYNCHRONIZATION IN BACKDOOR ATTACKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Backdoor attacks train models on a mixture of poisoned data and clean data to implant backdoor triggers into the model. An interesting phenomenon has been observed in the training process: the loss of poisoned samples tends to drop significantly faster than that of clean samples, which we call the *early-fitting* phenomenon. Early-fitting provides a simple but effective method to defend against backdoor attacks, as the poisoned samples can be identified by picking the samples with the lowest loss values in the early training epochs. Therefore, two natural questions arise: (1) What characteristics of poisoned samples cause early-fitting? (2) Is it possible to design stronger attacks to circumvent existing defense methods? To answer the first question, we find that early-fitting could be attributed to a unique property of poisoned samples called *synchronization*, which depicts the latent similarity between two samples. Meanwhile, the degree of synchronization could be explicitly controlled based on whether it is captured by *shallow or deep* layers of the model. Then, we give an affirmative answer to the second question by proposing a new backdoor attack method, Deep Backdoor Attack (DBA), which utilizes deep synchronization to reversely generate trigger patterns by activating neurons in the deep layer. Experimental results validate our propositions and the effectiveness of DBA. Our code is available at https://anonymous.4open.science/r/Deep-Backdoor-Attack-8875.

1 INTRODUCTION

Deep neural networks (DNNs) have achieved tremendous success in many fields (Krizhevsky, 2009; Szegedy et al., 2015; Sutskever et al., 2014). Training a DNN requires a large amount of data and computational resources, so it is a common practice to train DNNs on third-party providers. In this scenario, backdoor attacks (Gu et al., 2017; Chen et al., 2017; Nguyen & Tran, 2020; Liu et al., 2020; Nguyen & Tran, 2021) pose a serious security threat by intervening in the training process. Attackers could poison the training data by injecting trigger patterns into a few training samples and changing their labels to the target label, so DNNs will learn the mappings from the trigger pattern to the target label. Then, in the inference stage, the model will predict input into the target label class if trigger patterns are present, but behave normally on clean samples.

Recently, it has been observed that the training loss of poisoned samples drops significantly faster than that of clean samples in the first few epochs (Li et al., 2021a). We call this the *early-fitting* phenomenon. As a result, poisoned samples can be accurately isolated from clean samples during training by selecting the samples with top-k lowest loss value. Then, fine-tuning the model with the remaining data will lessen the threat from backdoors. This defense strategy leverages the early-fitting phenomenon and works effectively against a wide range of attack methods.

Despite the observation and preliminary application of early-fitting, several key questions have not been answered. **First, what is the reason behind early-fitting of poisoned samples?** Backdoor patterns are usually simpler compared to natural signals, so they are easier to learn for neural networks. However, how to quantify such "simplicity" of backdoor patterns? Are there any other reasons for the rapid decrease of poisoned samples loss? We believe that answering these questions would provide a better understanding of backdoor attacks and defenses. **Second, is it possible for backdoor attacks to bypass the early-fitting phenomenon?** The answer to this question would help us identify potential threats to deep models.

To answer the first question, we propose a quantification method for estimating the loss reduction of training samples. We find that: 1) the loss reduction of each sample could be decomposed as the aggregated influence of other training samples; 2) the mutual influence of two samples is closely related to their similarities at different layers of a model. Thus, we define a novel concept called *synchronization* to measure such similarity. Poisoned samples tend to be more "synchronized" compared to clean samples, leading to the early-fitting phenomenon. Meanwhile, samples created by different attack methods show synchronization at different layers. The synchronization among poisoned samples created by simple attack methods (e.g., BadNet (Gu et al., 2017)) is usually captured starting from shallow layers, while synchronization of stealthy poisoned samples (e.g., WaNet (Nguyen & Tran, 2021)) is captured only at deep layers. Based on the above analysis, we answer the second question in the affirmative. Specifically, we propose a new backdoor attack method, called Deep Backdoor Attack (DBA), which generates poisoned samples with deep-layer synchronization and could avoid early-fitting. We leverage model explanation (Simonyan et al., 2013) to reversely generate trigger patterns by activating the deep neurons of a clean neural network. Then, the generated trigger patterns contain intricate information learned by the neural network, so the poisoned samples tend to synchronize in a more complex way. Experiments show that DBA can bypass the early-fitting phenomenon while maintaining good performance on clean samples.

Our contributions are as follows: (1) we formalize the early-fitting phenomenon and quantitatively measure the loss reduction of poisoned and clean samples; (2) we provide a better understanding of early-fitting from a novel perspective of synchronization between training samples; (3) we propose DBA, a new backdoor attack method that is resistant to both early-fitting and other recent defense methods, demonstrating the potential threat posed by backdoor attacks.

2 Understanding Backdoor Attacks

In this section, we first introduce the problem definition of backdoor attacks, and the phenomenon of early-fitting in training poisoned samples. Then, we provide a theoretical understanding of backdoor attacks from a new perspective called feature synchronization.

2.1 Problem Definition: Backdoor Attacks

We consider classification tasks in this paper. A neural network model $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$ is learned from the training dataset $\mathcal{D} = \{(\boldsymbol{x}, y)\}$, where $\boldsymbol{x} \in \mathbb{R}^n$ denotes a sample, $y \in \mathbb{R}$ denotes its ground-truth label, $\mathcal{X}$ is the input space, $\mathcal{Y}$ is the label space, and θ denotes the collection of model parameters.

The goal of backdoor attacks is to train the model f_θ on a mixture of clean and poisoned data samples, so that the model is expected to perform normally on clean input but behave incorrectly if special triggers are added to the input. We denote the set of clean and poisoned samples as $\mathcal{D}_{clean}$ and $\mathcal{D}_{poi}$, respectively, so $\mathcal{D} = \mathcal{D}_{clean} \cup \mathcal{D}_{poi}$. Typically, a poisoned sample $(\boldsymbol{x}_b, y_b) \in \mathcal{D}_{poi}$ could be created from a clean sample $(\boldsymbol{x}_c, y_c) \in \mathcal{D}_{clean}$ by adding trigger patterns $\boldsymbol{\delta}$ to make $\boldsymbol{x}_b = \boldsymbol{x}_c + \boldsymbol{\delta}$ and pairing it with a target label y_b chosen by attackers. We call the proportion of poisoned samples as injection ratio $\eta = |\mathcal{D}_{poi}|/|\mathcal{D}|$. After training on $\mathcal{D}$, the model is expected to assign y_b to future samples containing the trigger patterns but behaves normally in clean samples.

2.2 Preliminaries: The Early-Fitting Phenomenon

It has been observed that, when training a neural network on both poisoned and clean samples, the loss on the poisoned samples drops significantly faster than that of the clean ones in early epochs (Li et al., 2021a). We call this phenomenon *early-fitting*. As a result, the poisoned samples can be isolated from other samples by ranking the training loss of all samples from low to high, and picking the ones with the smallest loss values. It thus provides a straightforward way for defenders to detect and defend against backdoor attacks. An intuition behind the early-fitting phenomenon of poisoned samples is that: training on samples with backdoor patterns is a simpler task compared to training on clean samples, since the model only needs to learn a mapping from trigger $\boldsymbol{\delta}$ to a fixed label y_b, where the trigger patterns are not as complex as natural patterns.

Despite the intuition and preliminary application of early-fitting, several key questions remain to be answered towards further understanding the phenomenon and securing the models. ❶ What is the

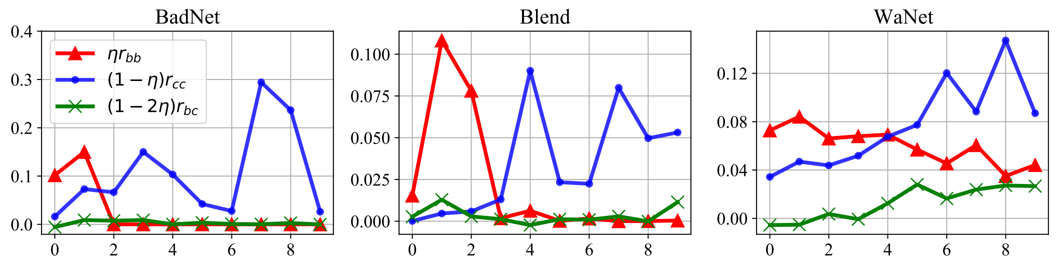

Figure 1: ηr_{bb}, ηr_{bc}, $(1-\eta)r_{cc}$, and $(1-\eta)r_{cb}$ under different training epochs when learning on Cifar10 under three backdoor attacks. η is set as 0.2.

reason that early-fitting happens on poisoned samples? One may conjecture that early-fitting occurs because it is easy to fit poisoned samples, but how to quantify such “easiness”, and are there any other reasons? We lack a quantitative tool to estimate loss reduction for poisoned samples and clean samples. ❷ Is it possible to design stronger attacks to circumvent the existing defense methods? We answer these questions in the rest of this paper.

2.3 Loss Reduction Estimation

In this part, we provide a better understanding of the reason behind early-fitting by quantifying the loss reduction during training. Let $R(\mathcal{D},(\boldsymbol{x},y))$ denote the loss reduction on a test sample $(\boldsymbol{x},y)$ after training the model with $\mathcal{D}$ in the i-th iteration. Thus, the average loss reduction over a subset $\mathcal{D}' \subset \mathcal{D}$ is defined as

$$\mathcal{R}(\mathcal{D},\mathcal{D}^{'}) = \frac{1}{|\mathcal{D}'|}\frac{1}{|\mathcal{D}|}\sum\nolimits_{(\boldsymbol{x},y)\in\mathcal{D}'}\sum\nolimits_{(\boldsymbol{x}',y')\in\mathcal{D}} r((\boldsymbol{x}',y');(\boldsymbol{x},y)). \tag{1}$$

where $r((\boldsymbol{x}',y');(\boldsymbol{x},y))$ denotes the influence of a training sample $(\boldsymbol{x}',y')$ on reducing the loss of $(\boldsymbol{x},y)$. The early-fitting phenomenon implies that $R(\mathcal{D},\mathcal{D}_{poi}) \geq R(\mathcal{D},\mathcal{D}_{clean})$. To understand why it happens, we propose to quantitatively estimate $R(\mathcal{D},\mathcal{D}')$. According to (Pruthi et al., 2020). $r((\boldsymbol{x}',y');(\boldsymbol{x},y))$ is estimated as:

$$r((\boldsymbol{x}',y');(\boldsymbol{x},y)) \approx \beta_i \nabla_{\theta_i}\ell_{\theta_i}(\boldsymbol{x},y)\cdot\nabla_{\theta_i}\ell_{\theta_i}(\boldsymbol{x}',y'), \tag{2}$$

where β_i denotes the learning rate at iteration i, $\ell_{\theta_i}(\boldsymbol{x},y)$ denotes the training loss of sample $\boldsymbol{x}$ after the i-th iteration, and ∇_{θ_i} denotes the partial derivative with respect to the weights θ_i.

We use a preliminary experiment to validate the estimation in Equation 2. Given the image classification task on Cifar10 (Krizhevsky, 2009), we randomly select 250 pairs of $((\boldsymbol{x}',y'),(\boldsymbol{x},y))$ from the training dataset $\mathcal{D}$ and compare the estimated loss reduction with its actual loss reduction (see Figure 5 in Appendix). We adopt three models to present the performance of the estimation: ResNet-18 (He et al., 2015), CNN-18 (see A.2), and LeNet-5 (Lecun et al., 1998). Figure 5 shows that the estimation points distribute along the line $y=x$, indicating a high estimation accuracy.

In iteration i, the parameters are θ_i and we train the neural network f_{θ_i} on the poisoned dataset $\mathcal{D}$. As the training dataset $\mathcal{D}$ is composed of clean samples $(\boldsymbol{x}_c,y_c)$ and poisoned samples $(\boldsymbol{x}_b,y_b)$, we can define loss reduction for the poisoned subset $\mathcal{D}_{poi}$ and clean subset $\mathcal{D}_{clean}$ respectively. The details are presented in Appendix B.

Theorem 1. *Given $\mathcal{D}=\mathcal{D}_{poi}\cup\mathcal{D}_{clean}$ as training data, the difference of average loss reduction between $\mathcal{D}_{poi}$ and $\mathcal{D}_{clean}$ is:*

$$\mathcal{R}(\mathcal{D},\mathcal{D}_{poi}) - \mathcal{R}(\mathcal{D},\mathcal{D}_{clean}) = \eta\bar{r}_{bb} - (1-\eta)\bar{r}_{cc} + (1-2\eta)\bar{r}_{bc}, \tag{3}$$

where $\bar{r}_{bb} = \frac{\sum_{(\boldsymbol{x}_b',y_b'),(\boldsymbol{x}_b,y_b)\in\mathcal{D}_{poi}} r((\boldsymbol{x}_b',y_b');(\boldsymbol{x}_b,y_b))}{|\mathcal{D}_{poi}||\mathcal{D}_{poi}|}$, $\bar{r}_{cc} = \frac{\sum_{(\boldsymbol{x}_c',y_c'),(\boldsymbol{x}_c,y_c)\in\mathcal{D}_{clean}} r((\boldsymbol{x}_c',y_c');(\boldsymbol{x}_c,y_c))}{|\mathcal{D}_{clean}||\mathcal{D}_{clean}|}$, $\bar{r}_{bc} = \frac{\sum_{(\boldsymbol{x}_b',y_b')\in\mathcal{D}_{poi}(\boldsymbol{x}_c,y_c)\in\mathcal{D}_{clean}} r((\boldsymbol{x}_b',y_b');(\boldsymbol{x}_c,y_c))}{|\mathcal{D}_{poi}||\mathcal{D}_{clean}|}$, *and* $\eta = \frac{|\mathcal{D}_{poi}|}{|\mathcal{D}|}$.

Theorem 1 shows that the difference of loss reduction between poisoned samples and clean samples can be decomposed into three components. The first component $\eta\bar{r}_{bb}$ is the average loss reduction

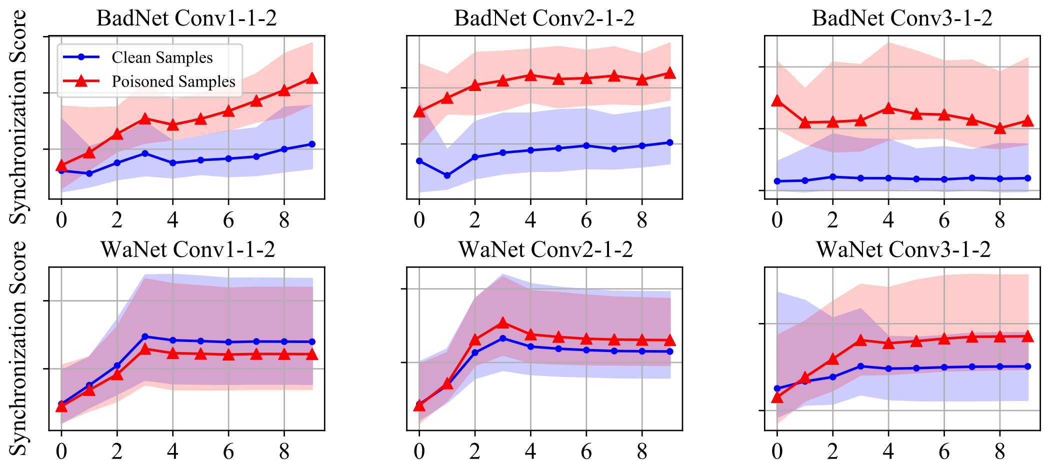

Figure 2: Synchronization value of poisoned and clean samples in different epochs (x-axis). We use BadNet (Gu et al., 2017) (first row) and WaNet (Nguyen & Tran, 2021) (second row) as examples.

corresponding to the influences among poisoned samples; the second component $(1-\eta)\bar{r}_{cc}$ corresponds to the influences among clean samples; the last component $(1-2\eta)\bar{r}_{bc}$ corresponds to the influences between poisoned samples and clean samples. Figure 1 plots the change of the three values on a Cifar10 classification task under different backdoor attacks. We could observe that, in the early training epochs, $\eta\bar{r}_{bb}$ dominates the loss reduction while the other two values are significantly lower. It indicates that strong interaction exists among the poisoned samples, where training on one poisoned sample significantly contributes to the loss reduction of the other poisoned samples. Such a strong interaction is a leading factor of the abrupt loss reduction of poisoned samples. Now the question is, *why does strong interaction exist among poisoned samples?*

2.4 Synchronization Between Samples

To facilitate illustration, we first define an important concept named *synchronization* which measures the similarity between embeddings of two samples in the intermediate layers.

Definition 1. *(Synchronization Score) The synchronization score between sample $(\boldsymbol{x}, y)$ and $(\boldsymbol{x}', y')$ at layer l of neural network f_θ is defined as $S^l(\boldsymbol{x};\boldsymbol{x}') := \langle f_\theta^l(\boldsymbol{x}), f_\theta^l(\boldsymbol{x}')\rangle$.*

Here $f_\theta^l(\boldsymbol{x})$ is the output of model f_θ at layer $l \in [L]$, and $f_\theta^0(\boldsymbol{x}) = \boldsymbol{x}$. The synchronization score measures the latent similarity between $\boldsymbol{x}$ and $\boldsymbol{x}'$ at layer l. Although the above definition may seem trivial, we will show later that it plays a key role in controlling loss reduction during training.

Definition 2. *(Shallow/Deep-Synchronized Samples) Given a neural network f_θ, sample $(\boldsymbol{x}, y)$ and $(\boldsymbol{x}', y')$ are defined as a pair of shallow-synchronized samples if $S^l(\boldsymbol{x};\boldsymbol{x}') \geq \epsilon^l$ at layer l, where $0 \leq l \leq \lambda$. Similarly, $(\boldsymbol{x}, y)$ and $(\boldsymbol{x}', y')$ are defined as a pair of deep-synchronized samples if: 1) $S^l(\boldsymbol{x};\boldsymbol{x}') \geq \epsilon^l$ at layer l, where $\lambda \leq l \leq L$, and 2) $S^h(\boldsymbol{x};\boldsymbol{x}') < \epsilon^h, \forall 0 \leq h < l$. Here λ is the borderline between shallow and deep layers; ϵ^l and ϵ^h are thresholds for layer l and h.*

For shallow-synchronized samples $(\boldsymbol{x}, y)$ and $(\boldsymbol{x}', y')$, the neural networks can easily capture their similarity in both shallow and deep layers. For deep-synchronized samples, due to the complexity of backdoor patterns, their similarity could only be captured by deeper layers. For example, poisoned samples created by BadNet (Gu et al., 2017) are shallow-synchronized samples since the trigger patterns are fixed in terms of pixel values and locations (see Figure 6). However, poisoned samples created by WaNet (Nguyen & Tran, 2021) tend to be deep-synchronized as their trigger patterns are more stealthy and do not share much easily perceptible similarity. Such patterns could only be captured in deeper layers. To better visualize shallow and deep synchronization, we plot the synchronization scores in different layers and training epochs in Figure 2. For BadNet (first row), it is obvious that the synchronization scores of poisoned samples are significantly higher than those of the clean samples starting from the shallow layer and the difference is propagated throughout the whole DNN. For WaNet (second row), the synchronization scores of poisoned samples are distinguishable from clean samples only in the deep layers.

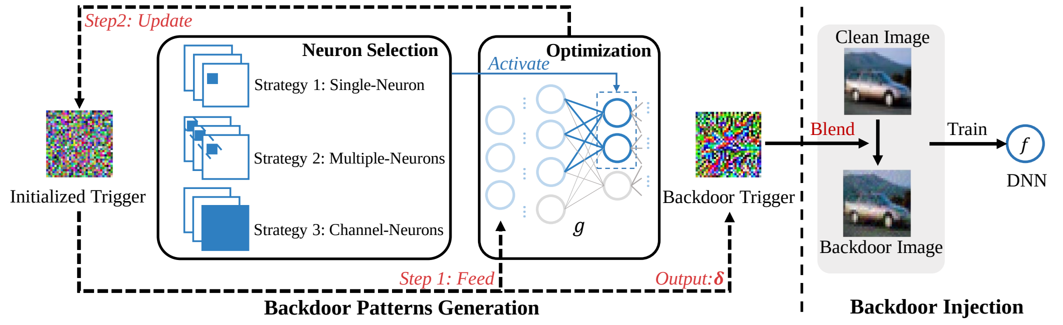

Figure 3: The pipeline of Deep Backdoor Attacks (DBA).

2.5 Relation between Synchronization and Loss Reduction

In this part, through further analyzing Equation 2, we show that synchronization score $S^l(\boldsymbol{x}_b, \boldsymbol{x}'_b)$ is closely related to the loss reduction. To this end, we give the following hypothesis:

Hypothesis 1. *In the early training epochs, given two poisoned samples $(\boldsymbol{x}, y)$ and $(\boldsymbol{x}', y')$, synchronization score $S^l(\boldsymbol{x}, \boldsymbol{x}')$ is proportional to the single loss reduction value $r((\boldsymbol{x}', y'); (\boldsymbol{x}, y))$.*

The theoretical analysis and empirical experiment results are provided in Appendix H to verify this hypothesis. It states that samples with a higher synchronization score have a stronger interaction to contribute to loss reduction. It thus inspires us to understand loss reduction of poisoned samples from a new perspective: feature synchronization. The similarity among poisoned samples, once captured by the neural network, could lead to a high synchronization score $S^l(\boldsymbol{x}_b, \boldsymbol{x}'_b)$. It then produces a large $r((\boldsymbol{x}'_b, y'_b); (\boldsymbol{x}_b, y_b))$, leading to faster loss reduction.

Based on the above analysis, *we could constrain the synchronization among poisoned samples*, so that their loss reduction will not be too distinctive from clean samples, thus alleviating early-fitting. Specifically, we could encourage deep synchronization and suppress shallow synchronization among poisoned samples. The reason is that shallow synchronization can propagate throughout the network to deeper layers, whereas deep synchronization only appears in the last few layers (see Figure 2 and analysis of Hypothesis 1). This makes the loss reduction smaller for deep-synchronized poisoned samples. Based on this, we propose a new backdoor attack in the section below.

3 Deep Backdoor Attacks

The early-fitting phenomenon provides opportunities for defenders to detect backdoor attacks. After quantitatively analyzing the reason behind early-fitting, we find that deep-synchronized poisoned samples are harder to be detected. Therefore, we propose a new attack method called Deep Backdoor Attacks (DBA), which will reduce the loss reduction gap between poisoned and clean samples.

3.1 Backdoor Pattern Generation by Activating Deep Neurons

To generate deep-synchronized samples, we propose to adopt feature inversion (Simonyan et al., 2013; Mahendran & Vedaldi, 2014), where we generate trigger patterns that maximize the activation of deep neurons by solving an optimization problem. Given a neural network g trained on clean data, a set of selected neurons $\mathcal{S}$ from deep layers, a trigger pattern $\boldsymbol{\delta}$ is generated by solving the optimization problem as follows:

$$\boldsymbol{\delta} = \arg\max_{\boldsymbol{x}'} \textstyle\sum_{s\in\mathcal{S}} g^{l,s}(\boldsymbol{x}'), \quad \text{subject to } \boldsymbol{x}' \in \mathcal{X}, \tag{4}$$

where $g^{l,s}(\boldsymbol{x}')$ is the activation of neuron s at layer l given input $\boldsymbol{x}'$. The optimization is solved by gradient descent on $\boldsymbol{x}'$, where $\boldsymbol{x}'$ can be randomly initialized. For a neural network g trained on clean data, its deep layers already encode intricate patterns. Intuitively, activating these neurons could generate trigger patterns that are harder to be captured than simple hand-crafted patterns, so they could be used for more stealthy attacks. The overall process of DBA is illustrated in Figure 3. Details for creating effective backdoor patterns are introduced in the following subsection.

3.2 Details of Designing Effective DBA

Diverse but Semantically Unified Triggers. Instead of generating only one trigger pattern that is applied to all input, the $\boldsymbol{x}'$ in Equation 4 is assigned different random initialization when poisoning different samples. The resultant pattern varies if initialized differently, but they all contain the same semantic information. It thus guarantees we fully distill the information encoded in deep neurons. The DBA attack with diverse trigger patterns is also more difficult to defend.

Neuron Selection. We introduce several neuron selection strategies. We use CNNs for illustration as we focus on image classification in this work. We denote each neuron in a CNN layer l by a triplet index (d, w, h), where d denotes the channel index of feature maps and w, h denotes the neuron location in the feature map. We propose three neuron selection strategies: single-neuron, multiple-neurons, and channel-neurons. Simply put, the single-neuron strategy activates a single neuron in the CNN layer; the multiple-neurons strategy activates multiple neurons along the depth dimension of feature maps; the channel-neurons strategy activates a channel of neurons. The neuron selections are illustrated in Figure 3. The objective functions of different strategies are as below:

$$\boldsymbol{\delta} = \begin{cases} \arg\max_{\boldsymbol{x}' \in \mathcal{X}} \; g^{l,(d,w,h)}(\boldsymbol{x}'), & \text{single-neuron} \\ \arg\max_{\boldsymbol{x}' \in \mathcal{X}} \; \sum_{d} g^{l,(d,w,h)}(\boldsymbol{x}'), & \text{multiple-neurons} \\ \arg\max_{\boldsymbol{x}' \in \mathcal{X}} \; \sum_{w,h} g^{l,(d,w,h)}(\boldsymbol{x}'), & \text{channel-neurons} \end{cases} . \tag{5}$$

The effectiveness of DBA varies under different strategies. We implement DBA with each of the strategies through experiments and report results in Table 6, which shows that the multiple-neurons strategy outperforms the other two strategies.

Blend Ratio. The trigger pattern $\boldsymbol{\delta}$ is superimposed on a clean sample $\boldsymbol{x}_c$ with a parameter α to generate the poisoned sample $\boldsymbol{x}_b = \alpha \cdot \boldsymbol{\delta} + (1-\alpha)\boldsymbol{x}_c$. Appendix D shows the poisoned sample with different blend ratio α. As observed, a lower blend ratio generates an image closer to the original clean image, whereas a higher blend ratio generates an image closer to the trigger patterns.

3.3 Attacks with DBA

The above steps generate a set of poisoned samples $\{(\boldsymbol{x}_b, y_b)\}$, which are injected into a clean training dataset $\{(\boldsymbol{x}_c, y_c)\}$ to conduct backdoor attacks. Please note that the number of poisoned samples is limited to a small portion of the whole dataset. After training on the clean and poisoned samples, the model f_θ learns both the original classification task and the backdoor injection task:

$$\min_{\theta} \; \sum_{(\boldsymbol{x}_c, y_c) \in \mathcal{D}_{clean}} y_c \log(f_\theta(\boldsymbol{x}_c)) + \sum_{(\boldsymbol{x}_b, y_b) \in \mathcal{D}_{poi}} y_b \log(f_\theta(\boldsymbol{x}_b)). \tag{6}$$

In the inference stage, f_θ is expected to output y_b when the trigger pattern is present in the input, but behaves normally when the input is clean.

4 Experiments

In this section, we investigate to what extent the proposed DBA method bypasses the early-fitting phenomenon, and evaluate the robustness of DBA to other defense methods.

4.1 Experimental Setups

Dataset and Model Architecture. We evaluate the effectiveness of DBA on three benchmark datasets: Cifar10, Cifar100 (Krizhevsky, 2009), and GTSRB (Stallkamp et al., 2012). We adopt WideResNet-16 (Zagoruyko & Komodakis, 2016) for Cifar10 and GTSRB, and ResNet-18 (He et al., 2015) for Cifar100. All experiments are run on a Tesla P100-PCIE-16GB GPU, with the batch size of 64 and a learning rate starting from 0.1, decreasing by a factor of 0.6 every ten epochs. The optimizer uses SGD with momentum of 0.9 and weight decay of 0.0005.

Baseline Methods. We compare DBA with seven representative backdoor attack methods, including five dirty-label attacks: BadNets (Gu et al., 2017), Blend Attack (Chen et al., 2017), Dynamic Attack (Nguyen & Tran, 2020), Refool Attack (Liu et al., 2020), and WaNet (Nguyen & Tran,

Dataset	Cifar10		GTSRB		Cifar100	
Attacks	CA (%) ↑	BSR (%) ↑	CA (%) ↑	BSR (%) ↑	CA (%) ↑	BSR (%) ↑
BadNets	$85.12_{\pm 0.07}$	$8.67_{\pm 2.62}$	$98.46_{\pm 0.21}$	$8.93_{\pm 1.63}$	$63.72_{\pm 0.10}$	$10.60_{\pm 5.40}$
Blend	$84.78_{\pm 0.31}$	$10.63_{\pm 2.05}$	$98.26_{\pm 0.19}$	$14.82_{\pm 4.99}$	$61.80_{\pm 0.22}$	$26.58_{\pm 10.00}$
SIG	$84.67_{\pm 0.48}$	$13.93_{\pm 0.01}$	$97.36_{\pm 0.30}$	$24.98_{\pm 7.87}$	$60.79_{\pm 0.48}$	$45.33_{\pm 6.13}$
Dynamic	$83.56_{\pm 0.38}$	$9.34_{\pm 1.55}$	$97.14_{\pm 0.26}$	$11.45_{\pm 0.50}$	$61.65_{\pm 0.29}$	$8.67_{\pm 3.09}$
FC	$82.14_{\pm 0.54}$	$36.01_{\pm 10.34}$	$95.15_{\pm 0.36}$	$32.30_{\pm 5.55}$	$58.84_{\pm 0.37}$	$70.28_{\pm 6.97}$
Refool	$82.06_{\pm 0.17}$	$8.50_{\pm 3.47}$	$93.51_{\pm 0.08}$	$19.92_{\pm 3.46}$	$59.57_{\pm 0.32}$	$28.23_{\pm 3.68}$
WaNet	$82.31_{\pm 0.19}$	$28.02_{\pm 13.49}$	$96.40_{\pm 0.31}$	$32.10_{\pm 5.80}$	$57.75_{\pm 1.60}$	$35.10_{\pm 5.61}$
DBA (this work)	$85.32_{\pm 0.37}$	$\mathbf{72.87}_{\pm 11.13}$	$97.78_{\pm 0.43}$	$\mathbf{97.14}_{\pm 1.08}$	$61.21_{\pm 0.19}$	$\mathbf{83.47}_{\pm 16.5}$

Table 1: Clean Accuracy (%) and Backdoor Success Rate (%) of different attacks methods.

Dataset	Cifar10		GTSRB		Cifar100	
Attacks	BA (%) ↑	IP (%) ↓	BA (%) ↑	IP (%) ↓	BA (%) ↑	IP (%) ↓
BadNets	$99.93_{\pm 0.01}$	$91.33_{\pm 2.62}$	$99.99_{\pm 0.01}$	$91.07_{\pm 0.02}$	$100.00_{\pm 0.00}$	$89.40_{\pm 5.40}$
Blend	$99.97_{\pm 0.02}$	$89.33_{\pm 2.05}$	$99.98_{\pm 0.01}$	$85.18_{\pm 0.05}$	$99.94_{\pm 0.01}$	$73.40_{\pm 10.00}$
SIG	$99.91_{\pm 0.01}$	$86.75_{\pm 0.05}$	$100.00_{\pm 0.00}$	$74.69_{\pm 7.56}$	$99.99_{\pm 0.01}$	$54.67_{\pm 6.13}$
Dynamic	$99.86_{\pm 0.02}$	$90.26_{\pm 1.33}$	$99.41_{\pm 0.30}$	$86.20_{\pm 0.05}$	$99.99_{\pm 0.01}$	$91.33_{\pm 3.10}$
FC	$99.98_{\pm 0.02}$	$63.99_{\pm 10.34}$	$99.41_{\pm 0.30}$	$64.08_{\pm 6.79}$	$96.20_{\pm 0.81}$	$27.00_{\pm 6.68}$
Refool	$68.10_{\pm 2.74}$	$86.07_{\pm 4.34}$	$97.12_{\pm 0.25}$	$74.60_{\pm 7.50}$	$91.89_{\pm 0.44}$	$69.67_{\pm 3.20}$
WaNet	$99.74_{\pm 0.01}$	$71.91_{\pm 0.13}$	$100.00_{\pm 0.00}$	$67.90_{\pm 5.80}$	$99.99_{\pm 0.02}$	$67.30_{\pm 3.45}$
DBA (this work)	$98.65_{\pm 0.17}$	$\mathbf{25.37}_{\pm 11.69}$	$98.03_{\pm 1.07}$	$\mathbf{1.26}_{\pm 0.06}$	$99.98_{\pm 0.01}$	$\mathbf{11.00}_{\pm 15.55}$

Table 2: Backdoor Accuracy (%) and Isolation Precision (%) of different attack methods.

2021); one clean-label attack: Sinusoidal Signal Attack (SIG) (Barni et al., 2019); and one feature-space attack: Feature collision (FC). The injection ratio for all the attacks is set as 0.1, i.e., 10% of the dataset $\mathcal{D}$ are poisoned samples.

Evaluation Metrics. Two classical metrics (Li et al., 2020), Clean Accuracy (CA) and Backdoor Accuracy (BA), are used for evaluating backdoor attack methods. CA measures the classification accuracy of clean samples in the clean test set. BA is the percentage of poisoned samples classified as the target label by the neural network in the poisoned test set. In addition, since the early-fitting phenomenon provides a simple but effective way to defend against backdoor attacks, we need auxiliary metrics to evaluate the effectiveness of backdoor attacks against the early-fitting-based defense. Therefore, we introduce two more metrics, Isolation Precision (IP) and Backdoor Success Rate (BSR). IP measures the portion of poisoned samples that can be detected by the early-fitting-based defense (Li et al., 2021a). BSR evaluates the portion of the poisoned samples that are successfully predicted to the target label and not detected by early-fitting. Specifically, BSR $=$ $(1 - \text{IP}) \times \text{BA}$. A lower IP value means that more poisoned samples will evade detection, leading to a higher BSR value. Meanwhile, a higher BA value means that the neural network can successfully classify the poisoned samples into the target label chosen by attackers with a higher probability. Thus, a higher BSR score means stronger attacks.

4.2 Attack Effectiveness Evaluation

Table 1 and 2 presents our main result, where our proposed DBA method is compared to other state-of-the-art attack methods on different datasets. Here DBA is implemented with the multiple-neurons strategy, and the poisoned samples are generated with a blend ratio $\alpha = 0.01$ (implementation details are given in Appendix E.1). Table 1 shows that DBA consistently achieves a high BSR score compared with other attack methods and relatively good performance in CA. In the Cifar10 dataset, DBA reports an average CA score of 85.32%, with a standard deviation of 0.37%, and an average BSR score of 72.87%, with a standard deviation of 11.13%. In contrast, the BSR score is lower than 10% for most other attack methods on this dataset. Among other baseline methods, although the FC attack method also achieves good performance in terms of BSR on some datasets, it significantly sacrifices the CA performance. Also, although BadNets and Blend have high CA scores, their BSR scores are low, meaning that they will be easily detected by the early-fitting phenomenon.

Dataset	Defense	CA (No Defense)	BA (No Defense)	CA ↓	BA ↑
Cifar-10	FP	$85.17_{\pm 0.28}$	$97.52_{\pm 0.39}$	$31.81_{\pm 0.46}$	$97.52_{\pm 0.39}$
	NAD	$85.25_{\pm 0.46}$	$97.76_{\pm 0.46}$	$92.65_{\pm 0.12}$	$97.55_{\pm 0.25}$
	ABL	$84.87_{\pm 0.43}$	$96.08_{\pm 0.66}$	$5.58_{\pm 1.30}$	$42.24_{\pm 3.84}$
GTSRB	FP	$97.42_{\pm 0.24}$	$99.90_{\pm 0.04}$	$57.23_{\pm 3.67}$	$100_{\pm 0.00}$
	NAD	$97.32_{\pm 0.13}$	$99.78_{\pm 0.03}$	$94.19_{\pm 0.52}$	$99.88_{\pm 0.05}$
	ABL	$93.21_{\pm 0.54}$	$97.21_{\pm 0.39}$	$70.61_{\pm 8.49}$	$15.84_{\pm 6.21}$
Cifar100	FP	$61.58_{\pm 0.30}$	$99.98_{\pm 0.00}$	$0.9_{\pm 0.02}$	$2.71_{\pm 0.83}$
	NAD	$61.21_{\pm 0.12}$	$99.93_{\pm 0.02}$	$53.82_{\pm 0.13}$	$99.63_{\pm 0.31}$
	ABL	$61.01_{\pm 0.12}$	$99.98_{\pm 0.00}$	$32.61_{\pm 0.41}$	$38.53_{\pm 12.61}$

Table 3: CA and BA with/without applying the defense methods on models attacked by DBA.

Since BSR is related to both BA and IP, we further report the performance of different attack methods under the two metrics in Table 2. For most attack methods, they could achieve almost 100% in BA, so BSR is mainly determined by IP. For an effective attack method, having a high BA score means it could successfully fool the model, and having a low IP score means it also avoids being detected by early-fitting, thus posing greater threats to the model. The proposed DBA attack maintains a significantly lower IP score compared with other methods across different datasets. The trigger patterns generated by DBA are more difficult to be captured by early-fitting.

4.3 Ablation Study

The hyperparameters for DBA include the neural selection strategies and blend ratio α. In this section, we study the influence of the hyperparameters on the performance of DBA.

4.3.1 Neuron Selection

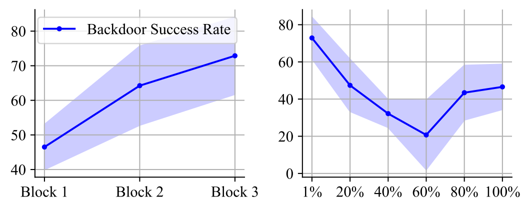

Figure 4: Isolation Precision of DBA with triggers inverted from: a) neurons of different depths, and b) different blend ratios α.

Neuron Selection Strategy. In Table 6, we compare the performance of DBA under three neuron selection strategies. The details are given in Appendix F. The multiple-neurons strategy is observed to achieve the best performance in BSR (72.87 %) and a premium result in CA (85.32 %).

Depth of the Inverted Neurons. We study the performance of DBA with triggers inverted from neurons of different layer depths. WideResNet-16 is adopted in the experiment. We generate triggers from neurons in the last CNN layer of Block 1 (shallow), Block 2, and Block 3 (deep), respectively. The results are shown in Figure 4 (left). As we can see, triggers inverted from deeper layers yield a higher BSR, meaning that the backdoor can be injected more successfully. Specifically, the DBA attacks achieve an average BSR of 46.50%, 64.24% and 72.87% by using trigger patterns generated from neurons in Block 1, Block 2 and Block 3, respectively. It demonstrates that triggers are embedded with more complex patterns if they are inverted from a deeper DNN layer.

4.3.2 Blend Ratio

The original images and triggers are blended with different blend ratios α. For example, if $\alpha = 0$, then the original image remains intact; if $\alpha = 1$, then the original image is completely replaced by the trigger. Therefore, we test the effect of varying the blend ratio on the DBA. The results are given in Figure 4 (right). As Figure 4 (right) shows, the BSR is heterogeneous when clean images are blended with different blend ratio. Specifically, when DBA is under a very low blend ratio (1%), the BSR is relatively higher (72.87% on average). This implies that attacks from DBA can be more effective when the trigger patterns are more stealthy.

4.4 Evaluation With Different Defense Methods

To test the general effectiveness of DBA, we use popular defense methods to defend against DBA, including Fine Pruning (FP) (Liu et al., 2018), NAD (Li et al., 2021b) and ABL (Li et al., 2021a). The details of these methods are given in Appendix E.5. As shown in Table 3, CA(No Defense) and BA(No Defense) present the clean accuracy and backdoor accuracy of a model backdoored by DBA, while CA(defense) and BA(defense) are the clean accuracy and backdoor accuracy after the corresponding defense. From the perspective of attackers, CA(defense) is expected to be low and BA(defense) is expected to be high, as it indicates that the defense method is in vain or even worsens the situation. As shown in Table 3, either the CA after defense is low, or the BA after defense remains high. For example, on the Cifar10 dataset, after Fine-pruning (FP), the CA turns out to be an average of 31.81%, and BA is still high (97.52% on average). The results demonstrate that our DBA is generally resilient to these state-of-the-art defense methods. To defend against DBA, we propose a preliminary isolation-based method. The details are given in Appendix G.

5 Related Work

Backdoor Attacks. Backdoor attacks (Gu et al., 2017) aim to poison the dataset with trigger patterns. Recent research can be divided into two categories on making the trigger more stealthy to enhance the practicality of backdoor attacks. The first one aims to make the trigger pattern less visible to human eyes. For example, Chen et al. (2017) blends the clean images with random pixels. Liu et al. (2020) uses the natural reflection to construct the backdoor trigger. Nguyen & Tran (2021) applies the warping transformation to the backdoor trigger. Doan et al. (2021) constructs trigger patterns by solving an optimization problem. The other direction aims to make the training process less noticeable. For example, Shafahi et al. (2018) proposes clean-label attack, which perturbs the clean images without changing their labels. Moreover, some backdoor attacks can also be applied to a reverse-engineered dataset without accessing the original dataset (Liu et al., 2017). However, all these backdoor attacks are easy to be detected by a so-called early-fitting phenomenon (Li et al., 2021a), therefore the threats from these backdoor attacks are weakened. In this paper, we proposed Deep Backdoor Attack (DBA), which can effectively circumvent the phenomenon.

Backdoor Defense. Various defense methods have been proposed to mitigate the threat from the backdoor. As in (Li et al., 2020), we categorize the existing defense methods into five categories. First, detection-based defenses (Gao et al., 2019; Huang et al., 2019; Dong et al., 2021; Guo et al., 2021; Xiang et al., 2022) aim to detect whether the backdoor exists in the model. Second, preprocessing-based defenses (Doan et al., 2020) introduce a preprocessing module before the training procedure so that triggers can be inactivated. Third, defenses based on model reconstruction (Liu et al., 2018; Zhao et al., 2020; Zeng et al., 2021; Wu & Wang, 2021) directly eliminate the effect of backdoors by adjusting the model weights or network structures. Fourth, defenses based on trigger synthesis (Wang et al., 2019; Qiao et al., 2019; Shen et al., 2021) first reverse engineer the trigger patterns and then remove the hidden backdoor in the model. Lastly, training sample filtering-based defenses (Li et al., 2021a; Huang et al., 2022) work by first filtering poisoned samples from the poisoned dataset, then training the network exclusively in the rest of the dataset.

6 Conclusion

In this work, we identify the early-fitting phenomenon in the training process of backdoor attacks: the loss on poisoned samples drops significantly faster than on clean samples. To understand what causes the early-fitting, we interpret the phenomenon from the perspective of feature synchronization, where we define as the representation similarity of two samples $(\boldsymbol{x}, y)$ and $(\boldsymbol{x}', y')$. Furthermore, the synchronization can be categorized as shallow or deep based on whether it is captured by shallow or deep layers of the model. By quantifying and decomposing the loss reduction, we find that synchronization among poisoned samples is the leading factor for the abrupt loss reduction. Therefore, we conjecture that constraining the synchronization value may alleviate early-fitting, making the poisoned samples more difficult to detect. To this end, we propose a new backdoor attack method, Deep Backdoor Attack (DBA), which inhibits synchronization among poisoned samples by activating deep-layer neurons to generate trigger patterns reversely. Comprehensive experiments show the effectiveness of DBA. We also propose a preliminary defense strategy against DBA.

REFERENCES

Mauro Barni, Kassem Kallas, and Benedetta Tondi. A new backdoor attack in cnns by training set corruption without label poisoning. In *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 101–105. IEEE, 2019.

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.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Bao Gia Doan, Ehsan Abbasnejad, and Damith C. Ranasinghe. Februus: Input purification defense against trojan attacks on deep neural network systems. In *Annual Computer Security Applications Conference*, ACSAC '20, pp. 897–912, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450388580. doi: 10.1145/3427228.3427264. URL https://doi.org/10.1145/3427228.3427264.

Khoa Doan, Yingjie Lao, Weijie Zhao, and Ping Li. Lira: Learnable, imperceptible and robust backdoor attacks. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11946–11956, 2021. doi: 10.1109/ICCV48922.2021.01175.

Yinpeng Dong, Xiao Yang, Zhijie Deng, Tianyu Pang, Zihao Xiao, Hang Su, and Jun Zhu. Black-box detection of backdoor attacks with limited information and data, 2021. URL https://arxiv.org/abs/2103.13127.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Yansong Gao, Chang Xu, Derui Wang, Shiping Chen, Damith C. Ranasinghe, and Surya Nepal. Strip: A defence against trojan attacks on deep neural networks. 2019. doi: 10.48550/ARXIV.1902.06531. URL https://arxiv.org/abs/1902.06531.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

Junfeng Guo, Ang Li, and Cong Liu. Aeva: Black-box backdoor detection using adversarial extreme value analysis, 2021. URL https://arxiv.org/abs/2110.14880.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385.

Kunzhe Huang, Yiming Li, Baoyuan Wu, Zhan Qin, and Kui Ren. Backdoor defense via decoupling the training process. *arXiv preprint arXiv:2202.03423*, 2022.

Xijie Huang, Moustafa Alzantot, and Mani Srivastava. Neuroninspect: Detecting backdoors in neural networks via output explanations, 2019. URL https://arxiv.org/abs/1911.07399.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

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. doi: 10.1109/5.726791.

Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Anti-backdoor learning: Training clean models on poisoned data. In *NeurIPS*, 2021a.

Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Neural attention distillation: Erasing backdoor triggers from deep neural networks. In *ICLR*, 2021b.

Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey, 2020. URL https://arxiv.org/abs/2007.08745.

Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks, 2018. URL `https://arxiv.org/abs/1805.12185`.

Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. 2017.

Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. Reflection backdoor: A natural backdoor attack on deep neural networks, 2020. URL `https://arxiv.org/abs/2007.02343`.

Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them, 2014. URL `https://arxiv.org/abs/1412.0035`.

Anh Nguyen and Anh Tran. Wanet – imperceptible warping-based backdoor attack, 2021. URL `https://arxiv.org/abs/2102.10369`.

Tuan Anh Nguyen and Anh Tran. Input-aware dynamic backdoor attack. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 3454–3464. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/234e691320c0ad5b45ee3c96d0d7b8f8-Paper.pdf`.

Garima Pruthi, Frederick Liu, Mukund Sundararajan, and Satyen Kale. Estimating training data influence by tracing gradient descent, 2020. URL `https://arxiv.org/abs/2002.08484`.

Ximing Qiao, Yukun Yang, and Hai Li. Defending neural backdoors via generative distribution modeling, 2019. URL `https://arxiv.org/abs/1910.04749`.

Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks, 2018. URL `https://arxiv.org/abs/1804.00792`.

Guangyu Shen, Yingqi Liu, Guanhong Tao, Shengwei An, Qiuling Xu, Siyuan Cheng, Shiqing Ma, and Xiangyu Zhang. Backdoor scanning for deep neural networks through k-arm optimization, 2021. URL `https://arxiv.org/abs/2102.05123`.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32:323–332, 2012.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks, 2014.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 707–723, 2019. doi: 10.1109/SP.2019.00031.

Dongxian Wu and Yisen Wang. Adversarial neuron pruning purifies backdoored deep models. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 16913–16925. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper/2021/file/8cbe9ce23f42628c98f80fa0fac8b19a-Paper.pdf`.

Zhen Xiang, David J. Miller, and George Kesidis. Post-training detection of backdoor attacks for two-class and multi-attack scenarios, 2022. URL `https://arxiv.org/abs/2201.08474`.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith (eds.), *Proceedings of the British Machine Vision Conference (BMVC)*, pp. 87.1–87.12. BMVA Press, September 2016. ISBN 1-901725-59-6. doi: 10.5244/C.30.87. URL `https://dx.doi.org/10.5244/C.30.87`.

Yi Zeng, Si Chen, Won Park, Zhuoqing Mao, Ming Jin, and Ruoxi Jia. Adversarial unlearning of backdoors via implicit hypergradient. In *International Conference on Learning Representations*, 2021.

Pu Zhao, Pin-Yu Chen, Payel Das, Karthikeyan Natesan Ramamurthy, and Xue Lin. Bridging mode connectivity in loss landscapes and adversarial robustness, 2020. URL `https://arxiv.org/abs/2005.00060`.

A Estimation of single loss reduction

A.1 Experiment Results

The Figure 5 presents the estimation accuracy. Given 250 samples from the dataset and a randomly initialized model, we compute their actual loss reduction (x-axis) and estimated loss reduction (y-axis). It is easy to observe that nearly all the dots fall on the line $y = x$, indicating a high accuracy of estimation.

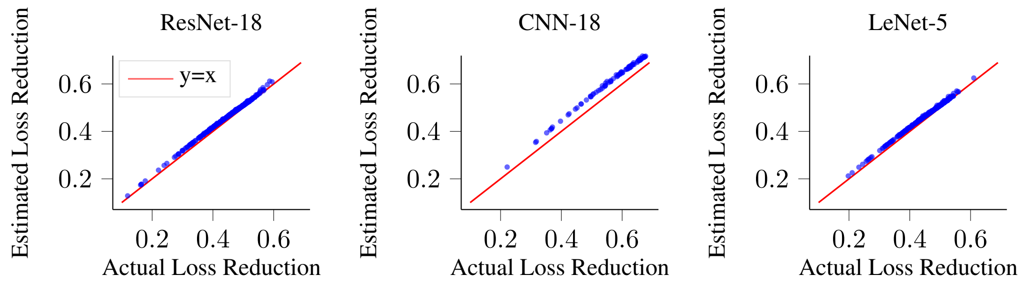

Figure 5: Estimated loss reduction versus true loss reduction on a (a) ResNet-18 neural network, (b) 18-layer CNN network and (c) LeNet-5. The training dataset is Cifar10. The learning rate is 0.1.

A.2 Architectures

Network	#Classes	Input Size	Parameters Size	# of Layers
CNN-18	10	$32 \times 32 \times 3$	$\sim 500K$	18

Table 4: Network models used in single loss reduction estimation experiment

Table 4 presents the architectures of the self-implemented CNN-18 network models in the estimation experiment in Figure 5. It is noted that CNN-18 is very similar to ResNet-18 (He et al., 2015) but just drops all the shortcut connections.

B Decomposing $\mathcal{R}(\mathcal{D}, \mathcal{D}_{poi})$ and $\mathcal{R}(\mathcal{D}, \mathcal{D}_{clean})$

As defined in 1, $\mathcal{R}(\mathcal{D}, \mathcal{D}_{poi})$ and $\mathcal{R}(\mathcal{D}, \mathcal{D}_{clean})$ can be decomposed as follows,

$$\begin{aligned} \mathcal{R}(\mathcal{D}, \mathcal{D}_{poi}) &= \frac{1}{|\mathcal{D}_{poi}|} \sum_{(\boldsymbol{x}_b, y_b) \in \mathcal{D}_{poi}} \mathcal{R}(\mathcal{D}, (\boldsymbol{x}_b, y_b)) \\ &= \frac{1}{|\mathcal{D}_{poi}|} \sum_{(\boldsymbol{x}_b, y_b) \in \mathcal{D}_{poi}} \frac{1}{|\mathcal{D}|} \sum_{(\boldsymbol{x}'_c, y'_c)} r((\boldsymbol{x}'_c, y'_c); (\boldsymbol{x}_b, y_b)) + \frac{1}{|\mathcal{D}|} \sum_{(\boldsymbol{x}'_b, y'_b)} r((\boldsymbol{x}'_b, y'_b), (\boldsymbol{x}_b, y_b)) \end{aligned} \tag{7}$$

$$\begin{aligned} \mathcal{R}(\mathcal{D}, \mathcal{D}_{clean}) &= \frac{1}{|\mathcal{D}_{clean}|} \sum_{(\boldsymbol{x}_c, y_c) \in \mathcal{D}_{clean}} \mathcal{R}(\mathcal{D}, (\boldsymbol{x}_c, y_c)) \\ &= \frac{1}{|\mathcal{D}_{clean}|} \sum_{(\boldsymbol{x}_c, y_c) \in \mathcal{D}_{clean}} \frac{1}{|\mathcal{D}|} \sum_{(\boldsymbol{x}'_c, y'_c)} r((\boldsymbol{x}'_c, y'_c), (\boldsymbol{x}_c, y_c)) + \frac{1}{|\mathcal{D}|} \sum_{(\boldsymbol{x}'_b, y'_b)} r((\boldsymbol{x}'_b, y'_b); (\boldsymbol{x}_c, y_c)) \end{aligned} \tag{8}$$

Moreover, we know that $\eta|\mathcal{D}| = |\mathcal{D}_{poi}|$, and $(1-\eta)|\mathcal{D}| = |\mathcal{D}_{clean}|$. Then,

$$\begin{aligned}
&\mathcal{R}(\mathcal{D},\mathcal{D}_{poi}) - \mathcal{R}(\mathcal{D},\mathcal{D}_{clean}) \\
=&\frac{1}{|\mathcal{D}|\cdot|\mathcal{D}_{poi}|}\left[\sum_{(\boldsymbol{x}'_b,y'_b),(\boldsymbol{x}_b,y_b)} r((\boldsymbol{x}'_b,y'_b);(\boldsymbol{x}_b,y_b)) + \sum_{(\boldsymbol{x}_b,y_b),(\boldsymbol{x}'_c,y'_c)} r((\boldsymbol{x}'_c,y'_c);(\boldsymbol{x}_b,y_b))\right] \\
&-\frac{1}{|\mathcal{D}|\cdot|\mathcal{D}_{clean}|}\sum_{(\boldsymbol{x}_c,y_c),(\boldsymbol{x}'_c,y'_c)}\left[r(\boldsymbol{x}'_c,y'_c);(\boldsymbol{x}_c,y_c) + \sum_{(\boldsymbol{x}_c,y_c),(\boldsymbol{x}'_b,y'_b)} r((\boldsymbol{x}'_b,y'_b);(\boldsymbol{x}_c,y_c))\right] \\
=&\frac{1}{\frac{1}{\eta}|\mathcal{D}_{poi}|\cdot|\mathcal{D}_{poi}|}\sum_{(\boldsymbol{x}'_b,y'_b),(\boldsymbol{x}_b,y_b)} r(x'_b,y'_b,x_b,y_b) \\
&-\frac{1}{\frac{1}{1-\eta}|\mathcal{D}_{clean}|\cdot|\mathcal{D}_{clean}|}\sum_{(\boldsymbol{x}'_c,y'_c),(\boldsymbol{x}_c,y_c)} r(x'_c,y'_c,x_c,y_c) \\
&+\frac{(|\mathcal{D}_{clean}|-|\mathcal{D}_{poi}|)}{|\mathcal{D}_{clean}|\cdot|\mathcal{D}_{poi}||\mathcal{D}|}\left[\sum_{(\boldsymbol{x}'_c,y'_c),(\boldsymbol{x}_b,y_b)} r(x'_c,y'_c,x_b,y_b)\right] \\
=&\frac{\eta}{|\mathcal{D}_{poi}|\cdot|\mathcal{D}_{poi}|}\sum_{(\boldsymbol{x}'_b,y'_b),(\boldsymbol{x}_b,y_b)} r(x'_b,y'_b,x_b,y_b) \\
&-\frac{1-\eta}{|\mathcal{D}_{clean}|\cdot|\mathcal{D}_{clean}|}\sum_{(\boldsymbol{x}'_c,y'_c),(\boldsymbol{x}_c,y_c)} r(x'_c,y'_c,x_c,y_c) \\
&+\frac{((1-\eta)-\eta)}{|\mathcal{D}_{clean}|\cdot|\mathcal{D}_{poi}|}\left[\sum_{(\boldsymbol{x}'_c,y'_c),(\boldsymbol{x}_b,y_b)} r(x'_c,y'_c,x_b,y_b)\right] \\
=&\eta\bar{r}_{bb} - (1-\eta)\bar{r}_{cc} + (1-2\eta)\bar{r}_{bc}
\end{aligned} \tag{9}$$

where $\bar{r}_{bb} = \frac{\sum_{(\boldsymbol{x}'_b,y'_b),(\boldsymbol{x}_b,y_b)\in\mathcal{D}_{poi}} r((\boldsymbol{x}'_b,y'_b);(\boldsymbol{x}_b,y_b))}{|\mathcal{D}_{poi}||\mathcal{D}_{poi}|}$, $\bar{r}_{cc} = \frac{\sum_{(\boldsymbol{x}'_c,y'_c),(\boldsymbol{x}_c,y_c)\in\mathcal{D}_{clean}} r((\boldsymbol{x}'_c,y'_c);(\boldsymbol{x}_c,y_c))}{|\mathcal{D}_{clean}||\mathcal{D}_{clean}|}$, $\bar{r}_{bc} = \frac{\sum_{(\boldsymbol{x}'_b,y'_b)\in\mathcal{D}_{poi}(\boldsymbol{x}_c,y_c)\in\mathcal{D}_{clean}} r((\boldsymbol{x}'_b,y'_b);(\boldsymbol{x}_c,y_c))}{|\mathcal{D}_{poi}||\mathcal{D}_{clean}|}$, and $\eta = \frac{|\mathcal{D}_{poi}|}{|\mathcal{D}|}$.

C Poisoned Images by BadNet

Figure 6 presents some examples of BadNet.

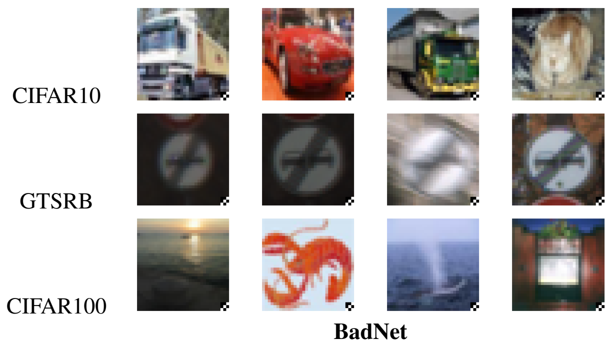

Figure 6: Poisoned samples by BadNet in different dataset

D Blend Ratio for DBA

Figure 7 presents the poisoned images of DBA under blend ratio.

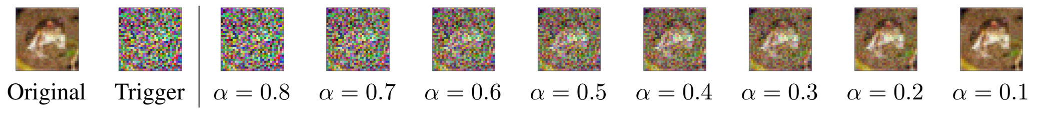

Figure 7: From left to right, the trigger pattern δ is blended with the original image under different blend ratios.

E Detailed Setting for Main Results

E.1 More Details About Deep Backdoor Attack

For all the experiments in Table 1, Table 2 and Figure 4 (right), the DBA adopts multiple-neurons strategy by activating the neurons with index ([4],4,4) in the last CNN layer of each network model.

For all the ablation experiments in Figure 4 (left), we adopt the multiple-neurons strategy. Block 1/2/3 refers to activating neurons with index ([4],4,4) in the last CNN layer of Block 1/2/3 of the WideResNet-16 network, respectively.

E.2 More Details About Metrics

Clean Accuracy The portion of correctly classified clean samples in the test set.

$$\frac{\#\text{Correctly Classified Clean Samples}}{\#\text{Total Samples}} \tag{10}$$

Backdoor Accuracy The portion of correctly classified poisoned samples in the test set.

$$\frac{\#\text{Correctly Classified Poisoned Samples}}{\#\text{Total Samples}} \tag{11}$$

Isolation Precision The portion of poisoned samples that can be isolated from the dataset

$$\frac{\#\text{Isolated Poisoned Samples}}{\#\text{Total Poisoned Samples}} \tag{12}$$

Note that in the experiment, we calculate isolation precision with the following procedure: suppose the injection ratio is η, then we isolate η samples from the whole dataset with the lowest loss value. The isolation process is done at the end of the first five training epochs. We report the maximal isolation precision among the five values.

Backdoor Success Accuracy The portion of the poisoned samples that are successfully predicted to the target label and not detected by early-fitting.

E.3 More Details About Datasets and DNNs

We focus solely on the image classification task in the experimental part. Therefore, we adopt three classical datasets, Cifar10, Cifar100 (Krizhevsky, 2009), and GTSRB (Stallkamp et al., 2012), for all attack methods.

Cifar10 Cifar10 is collected by (Krizhevsky, 2009), as a labeled subset of the 80 million tiny images dataset (Deng et al., 2009). It contains 60,000 images, with 6000 images per class. It is split into two sets. The training set contains 50,000 samples, and the test set contains 10,000 samples. The input size for all the examples are $32 \times 32 \times 3$.

Cifar100 Cifar100 is also collected by (Krizhevsky, 2009). It is like Cifar10 but it has 100 classes, with 600 images per class. For Cifar100, the training set contains 50,000 samples and the test set contains 10,000 samples. The input sizes for all samples are $32 \times 32 \times 3$.

GTSRB GTSRB collects more than 50,000 images of German traffic signs. There are 43 classes in the dataset. The training set contains 39,209 samples and 12,630 test samples. The input sizes for all samples are $32 \times 32 \times 3$.

We adopt WideResNet-16 (Zagoruyko & Komodakis, 2016) for Cifar10 and GTSRB, and ResNet-18[*] (He et al., 2015). The architectural details of these models are presented in Table 5.

Dataset	Subjects	#Classes	Input Size	#Train Images	Classifier
Cifar10	General Objects	10	$32 \times 32 \times 3$	50,000	WideResNet-16
Cifar100	General Objects	100	$32 \times 32 \times 3$	50,000	ResNet-18
GTSRB	Traffic Signs	43	$32 \times 32 \times 3$	39,209	WideResNet-16

Table 5: DNNs and Datasets used in the experiment

E.4 More Details About Attack Setups

For the experiment with each attack method, we poison 10% of the samples and the target label is set as label 3.

Settings for BadNet In all our experiments, the Badnet trigger is a 3×3 grid square. The bottom right of the images are replaced with the grid square.

Settings for Blend In all our experiments, the trigger for Blend is a Kitty image. The kitty image is superimposed on the original image with a blend ratio of 0.2.

Settings for SIG In all our experiments, the signal trigger is generated with a delta of 20, an f of 6, and is superimposed on the original image with a blend ratio of 0.2.

Settings for Dynamic The Dynamic Trigger is generated by a generator g trained for 40 epochs. The rest of the parameters align with the original paper.

Settings for FC The target image of the FC attack is a randomly selected image from the training datasets which has a different label as the target label. The rest of the parameters are the default values reported in their Github.

Settings for Refool Except for the target label, and the poisoning rate, we use the default parameters reported in their Github.

Settings for WaNet Except for the target label, and the poisoning rate, we use the default parameters that are reported in their Github.

E.5 More Details About Defense Settings

Fine Pruning is a defense based on model reconstruction. Given clean samples as input, it checks the activation value of each neuron and filters out a set of dormant neurons which are less activated by the neural network, assuming that these neurons are more likely to contribute to the backdoor attacks. Then, Fine Pruning will prune these neurons until the accuracy on the test set reaches a threshold. **NAD** utilizes a clean teacher network to guide the finetuning of the backdoored student model so that the intermediate layer of the backdoored model aligns with hat of the teacher model. **ABL** is a defense method based on the early-fitting phenomenon. First, it filters out poisoned samples from the poisoned dataset in the training process by selecting samples with top-k lowest loss value. Then, it finetunes the model on the remaining dataset and guides the neural network to unlearn the backdoor by maximizing a loss value with isolated poisoned samples as input.

[*]Implemented by https://github.com/weiaicunzai/pytorch-cifar100

Neuron Selection	CA (%) ↑	BSR (%) ↑
Single-neuron (Block3 Conv2 $(63, 4, 4)$)	85.75(±0.35)	54.06(±3.46)
Multiple-neurons (Block3 Conv2 $([:], 4, 4)$)	85.32(±0.37)	**72.87**(±11.13)
Channel-neurons (Block3 Conv2 $(63, [:], [:])$)	84.77(±0.65)	50.96(±15.19)

Table 6: DBA performance with different neuron selection strategies.

Settings for FP For the defense experiment on Cifar10, Cifar100, and GTSRB, we prune the neurons in the last CNN layer of the well-trained neural network. The pruning process is repeated until the accuracy reduction is greater than 5% or to the end.

Settings for NAD We conduct NAD based on its open-sourced code on Github[†]. Except for adjusting the NAD loss to adapt to the ResNet-18 architecture, the other parameters align with the default settings.

Settings for ABL We conduct ABL based on its open-sourced code on Github[‡]. The isolation epoch is set as 20. The other parameters align with the default settings.

F Ablation Study: Neuron Selection Strategy

We select neurons from the last CNN layer of a well-trained WideResNet-16 neural network on the Cifar10 dataset. The single-neuron strategy selects the neuron with index $(63, 4, 4)$, the multiple-neurons strategy selects the neurons with indices $([4], 4, 4)$, and the channel-neurons strategy selects the neurons with indices $(63, [:], [:])$.

G Isolation-based Defense Method

To defend against our proposed DBA, we propose an influence-based isolation method, which can isolate poisoned samples with higher accuracy. Motivated by (Pruthi et al., 2020), if we set $(\boldsymbol{x}, y) =$

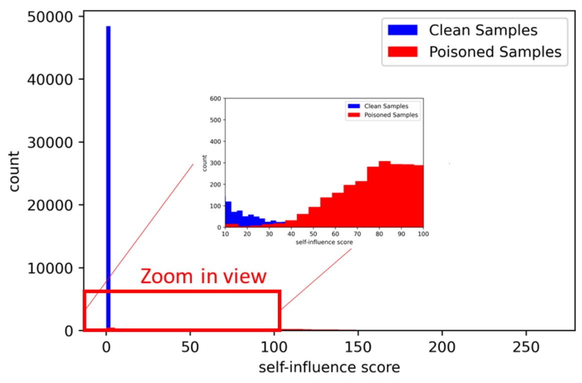

Figure 8: Histogram of the self-influence score for poisoned samples and clean samples respectively. The isolation precision for the deep backdoor attack is 90.36%

$(\boldsymbol{x}', y')$ in the single loss reduction value $r((\boldsymbol{x}, y); (\boldsymbol{x}', y'))$, we call the value as *self-influence*. Self-influence evaluates the loss reduction on one sample $(\boldsymbol{x}, y)$ after training the network on itself. For a network f_{θ_c} well-trained on the clean dataset, clean samples are expected to have a lower self-influence score since they are fitted to the network f_{θ_c}. Poisoned samples are expected to have a

[†]https://github.com/bboylyg/NAD
[‡]https://github.com/bboylyg/ABL

higher self-influence score because most of the poisoned samples with incorrect labels can be seen as outliers, where they would tend to reduce loss with respect to the incorrect label. Based on this, the poisoned samples can be isolated from the whole dataset by computing the self-influence score on the whole dataset $\mathcal{D}$ with f_{θ_c}, and picking the top-k samples with a high self-influence score.

Figure 8 presents the effectivenss of the defense method against the proposed Deep Backdoor Attack. As we can see, the majority of the self-influence score on the poisoned samples (red) is much higher than that on the clean samples (blue). In the empirical experiment, we set the injection ratio $\eta = 0.1$ and isolate 10% of the samples from the dataset $\mathcal{D}$. We run the experiment three times with different f_{θ_c}. The average isolation precision (IP) is 90.36%.

H ANALYSIS OF HYPOTHESIS 1

Intuition for Hypothesis 1. To find what makes the value of $r(\boldsymbol{x}', y'; \boldsymbol{x}, y)$ for poisoned sample pairs significantly high, we intend to decompose the $r(\boldsymbol{x}', y'; \boldsymbol{x}, y)$ and analyze each part empirically.

Analysis Firstly, we consider a simple case that the neural network f_θ is a fully-connected neural network and the output dimension is one, i.e., $f_\theta(x) \in \mathbb{R}$. Then, the neural network can be represented as follows:

$$\begin{aligned} &f^{(l)}(x) = \theta^{(l)} g^{(l-1)}(x) \\ &g^{(l)}(x) = \sigma^l(f^{(l)}(x)) \quad , \\ &l = 1, ..., L \end{aligned} \tag{13}$$

where $\sigma^l(\cdot)$ is the activation function in layer l. We further denote $\boldsymbol{x} = g^0(\boldsymbol{x})$ for notational convenience and the output of the last layer of the neural network is

$$f_\theta(\boldsymbol{x}) = g^{(L)}(x) \tag{14}$$

Therefore, We can decompose $r((\boldsymbol{x}', y'); (\boldsymbol{x}, y))$ as follows:

$$\begin{aligned} r((\boldsymbol{x}', y'); (\boldsymbol{x}, y)) &\approx \beta \langle \frac{\partial \ell_\theta(\boldsymbol{x}', y')}{\partial \theta}, \frac{\partial \ell_\theta(\boldsymbol{x}, y)}{\partial \theta} \rangle \\ &= \beta \langle \frac{\partial \ell_\theta(\boldsymbol{x}', y')}{\partial f_\theta(\boldsymbol{x}')} \cdot \frac{\partial f_\theta(\boldsymbol{x}')}{\partial \theta}, \frac{\partial \ell_\theta(\boldsymbol{x}, y)}{\partial f_\theta(\boldsymbol{x})} \cdot \frac{\partial f_\theta(\boldsymbol{x})}{\partial \theta} \rangle. \end{aligned} \tag{15}$$

where β is a constant value denoting the learning rate at the current iteration. Therefore, we can only consider the inner product part. Moreover, since $\frac{\partial \ell_\theta(\boldsymbol{x}', y')}{\partial f_\theta(\boldsymbol{x}')} \in \mathbb{R}$ and $\frac{\partial \ell_\theta(\boldsymbol{x}, y)}{\partial f_\theta(\boldsymbol{x})} \in \mathbb{R}$, Equation 15 can be further decomposed into the following form,

$$\begin{aligned} &\langle \frac{\partial \ell_\theta(\boldsymbol{x}, y)}{\partial f_\theta(\boldsymbol{x})} \cdot \frac{\partial f_\theta(\boldsymbol{x})}{\partial \theta}, \frac{\partial \ell_\theta(\boldsymbol{x}', y')}{\partial f_\theta(\boldsymbol{x}')} \cdot \frac{\partial f_\theta(\boldsymbol{x}')}{\partial \theta} \rangle \\ =& \frac{\partial \ell_\theta(\boldsymbol{x}, y)}{\partial f_\theta(\boldsymbol{x})} \cdot \frac{\partial \ell_\theta(\boldsymbol{x}', y')}{\partial f_\theta(\boldsymbol{x}')} \langle \frac{\partial f_\theta(\boldsymbol{x})}{\partial \theta}, \frac{\partial f_\theta(\boldsymbol{x}')}{\partial \theta} \rangle \\ =& \frac{\partial \ell_\theta(\boldsymbol{x}, y)}{\partial f_\theta(\boldsymbol{x})} \cdot \frac{\partial \ell_\theta(\boldsymbol{x}', y')}{\partial f_\theta(\boldsymbol{x}')} \sum_{l=1}^{L} \langle \frac{\partial f_\theta(\boldsymbol{x})}{\partial f_\theta^l(\boldsymbol{x})} \cdot (g_\theta^{l-1}(\boldsymbol{x}))^T, \frac{\partial f_\theta(\boldsymbol{x}')}{\partial f_\theta^l(\boldsymbol{x}')} \cdot (g_\theta^{l-1}(\boldsymbol{x}'))^T \rangle \\ =& \underbrace{\frac{\partial \ell_\theta(\boldsymbol{x}, y)}{\partial f_\theta(\boldsymbol{x})} \cdot \frac{\partial \ell_\theta(\boldsymbol{x}', y')}{\partial f_\theta(\boldsymbol{x}')}}_{\text{Part 1}} \cdot \sum_{l=1}^{L} \underbrace{\langle g_\theta^{l-1}(\boldsymbol{x}), g_\theta^{l-1}(\boldsymbol{x}') \rangle}_{\text{Part 2}} \cdot \underbrace{\langle \frac{\partial f_\theta(\boldsymbol{x})}{\partial f_\theta^l(\boldsymbol{x})}, \frac{\partial f_\theta(\boldsymbol{x}')}{\partial f_\theta^l(\boldsymbol{x}')} \rangle}_{\text{Part 3}}. \end{aligned} \tag{16}$$

Intuitively, Part 1 corresponds to the distance between the output to the ground-truth labels; Part 2 corresponds to the synchronization score in layer $l-1$; Part 3 corresponds to the gradient with respect to pre-activation layer l. Now our target is to analyze which part contributes the most to $r((\boldsymbol{x}', y'); (\boldsymbol{x}, y))$.

We use the following binary classification task on the Breast Cancer dataset (Dua & Graff, 2017) to compare the three parts. In particular, we randomly choose 10% of the data as the poisoned samples, where we change the last feature to a constant value of 5, and set the target label y_b as 1. It is noted that we only choose clean samples with label 0. In this way, the neural network f_θ is expected to predict class 1 when the last feature is 5, and predict class 0 when the sample is normal. We train the neural network for 9 iterations. The following Figure 9 shows the change of Part 1, Part 2, and Part 3 under the 9 iterations. As shown, clean samples and poisoned samples do not exhibit much difference in Part 3 and Part 1 but show a large gap in Part 2. This implies that the reason why $r(\boldsymbol{x}_b', \boldsymbol{x}_b)$ is larger and $r(\boldsymbol{x}_c', \boldsymbol{x}_c)$ is lower in the early epochs lies in their difference of synchronization score. Therefore, we conjecture that the synchronization score between two samples $(\boldsymbol{x}, y)$ and $(\boldsymbol{x}', y')$ may be proportional to the interaction between these two samples in the early epochs. To verify the conjecture, we consider conducting experiments on a general case with a more complex network structure and multi-dimensional output.

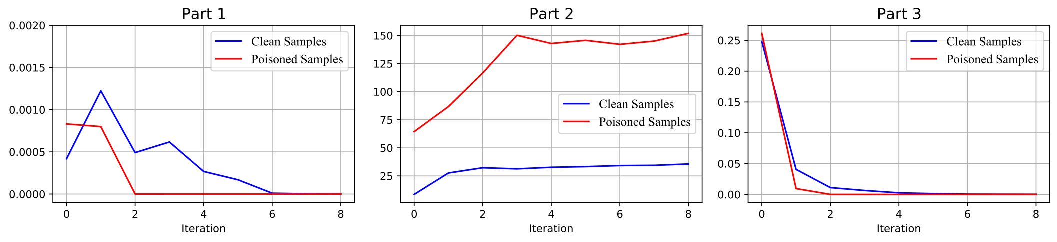

Figure 9: Compare the three parts in Equation 16 between poisoned samples and clean samples.

As in Figure 10, we sample 100 pairs from the Cifar10 dataset poisoned by BadNet (first row), Blend (second row), and WaNet (third row), and compute their estimated loss reduction $r((\boldsymbol{x}', y'); (\boldsymbol{x}, y)$ (y-axis), and synchronization scores (x-axis) under three layers (Conv1-1-1, Conv2-1-1, and Conv3-1-1) of WideResNet-16. Note that we use an unconverged model in the experiments, i.e., $CA \ll 1$ and $BA \ll 1$. The Pearson correlations for $r((\boldsymbol{x}', y'); (\boldsymbol{x}, y))$ and the synchronization score $S^l(\boldsymbol{x}, \boldsymbol{x}')$ show a moderately proportional relationship, which is consistent with our conjecture.

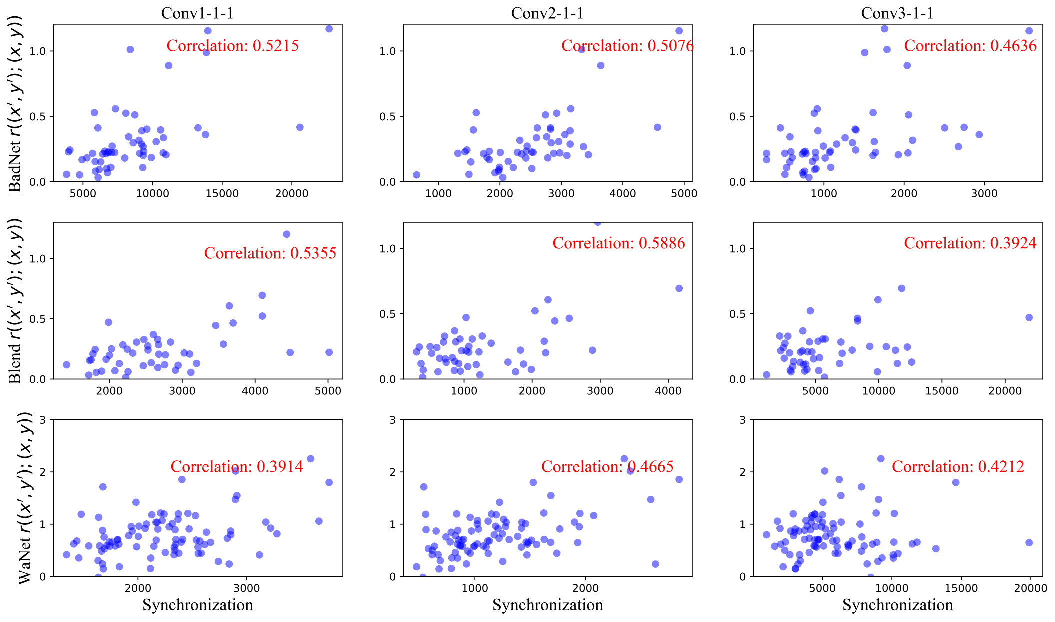

Figure 10: We sample 100 sample pairs from the Cifar10 dataset poisoned by BadNet/Blend/WaNet and compute their estimated loss reduction (y-axis), and synchronization scores (x-axis) under three layers (Conv1-1-1, Conv2-1-1, and Conv3-1-1). The correlation ratios show a moderately proportional relationship.