Attacking Neural Networks with Neural Networks: Towards Deep Synchronization for Backdoor Attacks

Zihan Guan
University of Georgia
Athens, Georgia, USA
zihan.guan@uga.edu

Lichao Sun
Lehigh University
Bethlehem, Pennsylvania, USA
lis221@lehigh.edu

Mengnan Du
New Jersey Institute of Technology
Newark, New Jersey, USA
mengnan.du@njit.edu

Ninghao Liu
University of Georgia
Athens, Georgia, USA
ninghao.liu@uga.edu

ABSTRACT

Backdoor attacks inject poisoned samples into training data, where backdoor triggers are embedded into the model trained on the mixture of poisoned and clean samples. An interesting phenomenon can be 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 evidence to defend against backdoor attacks, where the poisoned samples can be detected by selecting the samples with the lowest loss values in the early training epochs. Then, two questions naturally arise: (1) What characteristics of poisoned samples cause early-fitting? (2) Does a stronger attack exist which could circumvent the defense methods? To answer the first question, we find that early-fitting could be attributed to a unique property among poisoned samples called *synchronization*, which depicts the similarity between two samples at different layers of a model. Meanwhile, the degree of synchronization could be 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 reverse engineer trigger patterns by activating neurons in the deep layer of a base neural network. Experimental results validate our propositions and the effectiveness of DBA. Our code is available at https://github.com/GuanZihan/Deep-Backdoor-Attack.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; • **Information systems** → *Data mining*.

KEYWORDS

Backdoor Attacks; Model Interpretation; Deep Neural Networks

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0124-5/23/10…$15.00
https://doi.org/10.1145/3583780.3614784

ACM Reference Format:
Zihan Guan, Lichao Sun, Mengnan Du, and Ninghao Liu. 2023. Attacking Neural Networks with Neural Networks: Towards Deep Synchronization for Backdoor Attacks. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3583780.3614784

1 INTRODUCTION

Deep neural networks (DNNs) have achieved tremendous success in many applications [17, 42, 43]. Training a DNN requires a large amount of data and computational resources, so it is common practice to train DNNs on third-party providers. In this scenario, backdoor attacks [3, 9, 23, 25, 26, 45] 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 that the DNNs will learn the mappings from the trigger pattern to the target label. Then, in the inference stage, the model will predict the input into the target label class if trigger patterns are present but behave normally on clean samples.

Several previous efforts have been made to construct stealthy backdoor triggers [5, 23, 25] and launch controllable backdoor attacks [6, 40], making defense against backdoor attacks a challenging problem. However, it has recently been observed that the training loss of backdoor samples drops significantly faster than that of clean samples in the first few epochs [20]. We call this the *early-fitting* phenomenon. As a result, poisoned samples can be accurately isolated from clean samples during training by selecting samples with the 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 other reasons for the rapid decrease in poisoned sample 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 data 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. Our analysis indicates that 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 [9]) is usually captured starting from shallow layers, while synchronization of stealthy poisoned samples (e.g., WaNet [25]) 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 [38] 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. Comprehensive experiments across various tasks show that DBA can bypass the early-fitting phenomenon while maintaining good performance on clean samples. Most importantly, we demonstrate that DBA could be resistant to various other defense methods such as FP, NAD, and ABL. We summarize our contributions in this work as follows:

- We formalize the early-fitting phenomenon and quantitatively measure the loss reduction of poisoned and clean samples.
- We provide a better understanding of early-fitting from a novel perspective of data synchronization.
- We propose DBA, a new backdoor attack method that is resistant to both early-fitting and other competitive 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 data 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 $\mathcal{D} = \{(x, y)\}$, where $x \in \mathbb{R}^n$ 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.

Backdoor attacks make the model f_θ to be trained 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 exist in the input. We denote the set of clean and poisoned samples as $\mathcal{D}_{clean}$ and $\mathcal{D}_{poi}$, respectively, and let

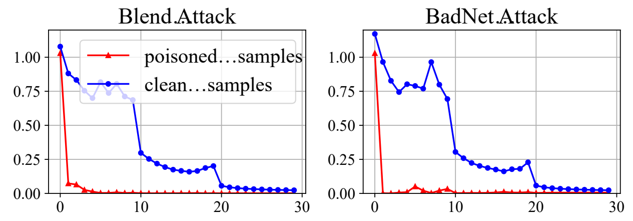

Figure 1: The training loss curve on clean samples and poisoned samples crafted with two attack methods: Blend [3] and BadNet [9].

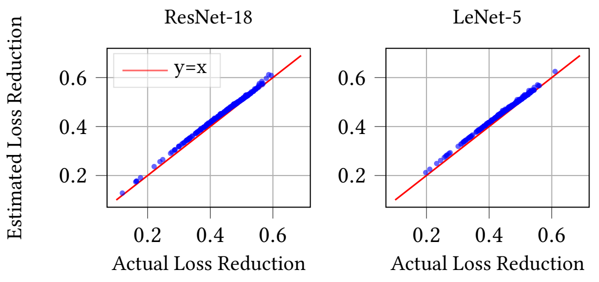

Figure 2: Estimated loss reduction versus true loss reduction on (a) ResNet-18 neural network, and (b) LeNet-5. The training dataset is Cifar10. The learning rate is 0.1.

$\mathcal{D} = \mathcal{D}_{clean} \cup \mathcal{D}_{poi}$. 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 δ to make $\boldsymbol{x}_b = \boldsymbol{x}_c + \delta$ 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 The Early-Fitting Phenomenon

It could be 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 (Figure 1). We call this phenomenon *early-fitting*. As a result, 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. Therefore, it provides a straightforward way for defenders to detect and defend against backdoor attacks. The 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. 1) What is the reason that early-fitting only 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 can we directly control whether a sample is easy to learn? We lack a quantitative tool to estimate the reduction in loss for poisoned and clean samples. 2) Is it possible to design stronger attacks to

Conference'17, July 2017, Washington, DC, USA Zihan Guan, Lichao Sun, Mengnan Du, and Ninghao Liu

of a model. Thus, we define a novel concept called *synchronization* to measure such similarity. Our analysis indicates that 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 [?]) is usually captured starting from shallow layers, while synchronization of stealthy poisoned samples (e.g., WaNet [?]) 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 [?] 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. Comprehensive experiments across various tasks show that DBA can bypass the early-fitting phenomenon while maintaining good performance on clean samples. Most importantly, we demonstrate that DBA could be resistant to various other defense methods such as FP, NAD, and ABL. We summarize our contributions in this work as follows:

- We formalize the early-fitting phenomenon and quantitatively measure the loss reduction of poisoned and clean samples.
- We provide a better understanding of early-fitting from a novel perspective of synchronization between training samples.
- We propose DBA, a new backdoor attack method that is resistant to both early-fitting and other competitive 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 data 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 $\mathcal{D} = \{(x, y)\}$, where $x \in \mathbb{R}^n$ 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.

2.2 Loss Reduction Estimation

Backdoor attacks make the model f_θ to be trained 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 exist in the input. We denote the set of clean and

Figure 1: Estimated loss reduction versus true loss reduction on (a) ResNet-18 neural network, and (b) LeNet-5. The training dataset is CIFAR10. The learning rate is 0.1.

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

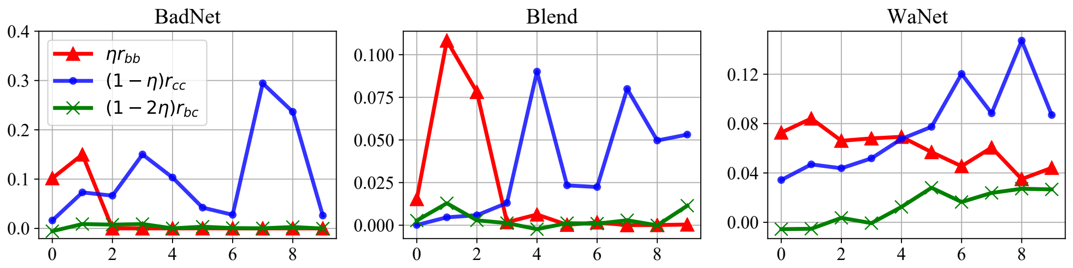

Figure 3: ηr_{bb}, ηr_{bc}, $(1-\eta)r_{cc}$, and $(1-\eta)r_{cb}$ under different training epochs (x-axis) when learning on CIFAR10 under three backdoor attacks. η is set as 0.1.

circumvent existing defense methods? We answer these questions in the remainder of this paper.

2.2.1 Loss Reduction Quantification. 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. The average loss reduction in a subset $\mathcal{D}' \subset \mathcal{D}$ is defined as:

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

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 $R(\mathcal{D},\mathcal{D}_{poi}) \geq R(\mathcal{D},\mathcal{D}_{clean})$ in the early epochs. To understand why it happens, we propose to quantitatively estimate $R(\mathcal{D},\mathcal{D}')$. According to [27], $r((\boldsymbol{x}',y');(\boldsymbol{x},y))$ can be 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. Intuitively, if $(\boldsymbol{x}',y')$ tends to update model parameters towards the same direction as $(\boldsymbol{x},y)$, then training on $(\boldsymbol{x}',y')$ will help the model to fit $(\boldsymbol{x},y)$ better.

We use a preliminary experiment to validate the estimation in Equation 2. Given the image classification task in CIFAR10 [17], 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 2). We adopt two models to present the performance of the estimation: ResNet-18 [14] and LeNet-5 [18]. Figure 2 shows that the estimation points distribution is along the line $y=x$, indicating a high estimation accuracy.

2.2.2 Loss Reduction Difference Between $\mathcal{D}_{poi}$ and $\mathcal{D}_{clean}$. With the quantification method at hand, we further study the difference between backdoor samples and clean samples in terms of the loss reduction speed. In iteration i, the parameters are θ_i and we train the neural network f_{θ_i} on the 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 define loss reduction for the poisoned subset $\mathcal{D}_{poi}$ and clean subset $\mathcal{D}_{clean}$ separately as follows. The details are presented in Appendix A.

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 in loss reduction between poisoned samples and clean samples can be decomposed into three components. The first component $\eta\bar{r}_{bb}$ corresponds to the average loss reduction corresponding to the influences between poisoned samples; the second component $(1-\eta)\bar{r}_{cc}$ corresponds to the influences between clean samples; the last component $(1-2\eta)\bar{r}_{bc}$ corresponds to the influences between poisoned samples and clean samples. Figure 3 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?**

2.3 Synchronization Between Samples

To help answer the above question, we define a novel and 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*

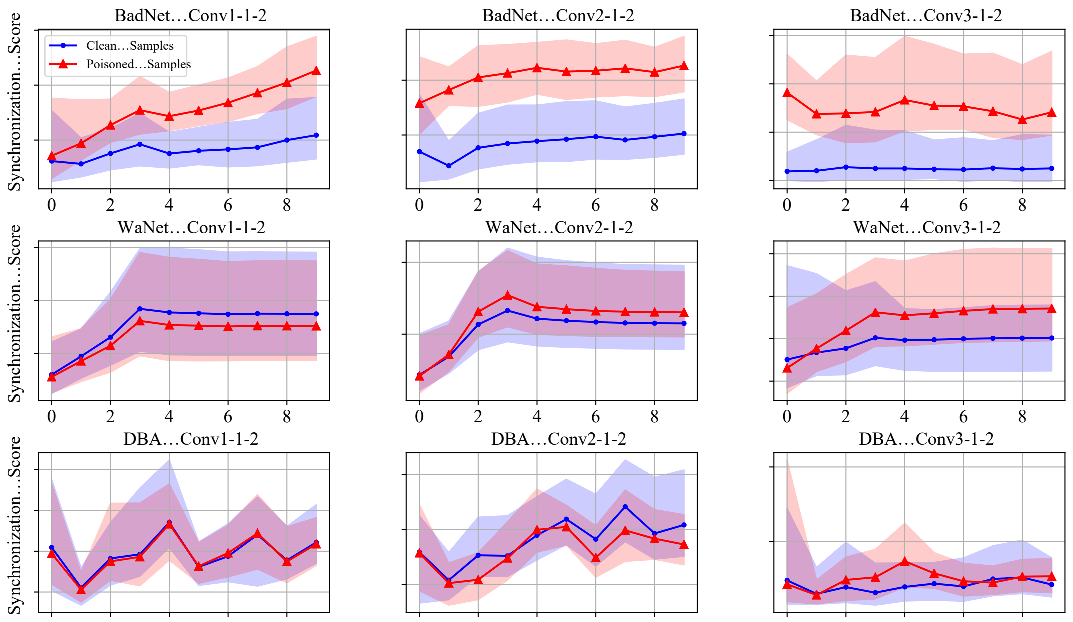

Figure 4: Synchronization value of poisoned (red) and clean (blue) samples in different epochs (x-axis). We use BadNet [9] (first row), WaNet [25] (second row), and our proposed DBA (third row) as examples.

$0 \le l \le \lambda$. Similarly, (x, y) and (x', y') are defined as a pair of deep-synchronized samples if: 1) $S^l(x; x') \ge \epsilon^l$ at layer l, where $\lambda \le l \le L$, and 2) $S^h(x; x') < \epsilon^h, \forall 0 \le h < l$. 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 (x, y) and (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 [9] are shallow-synchronized samples, since trigger patterns are fixed in terms of pixel values and locations. However, poisoned samples created by WaNet [25] and our proposed DBA 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 4. 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. For our proposed DBA (third row), the synchronization scores of poisoned samples are even hardly separable from those of clean samples in the deep layers.

2.4 Relation Between Synchronization and Loss Reduction

In this part, by further analyzing Equation 2, we show that the synchronization score $S^l(x_b, x'_b)$ 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 (x, y) and (x', y'), synchronization score $S^l(x, x')$ is contributive to the single loss reduction value $r((x', y'); (x, y))$.*

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

Based on the above analysis, a more effective attack should 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 appears only in the last few layers (see Figure 4 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 above analysis quantitatively investigates the reason behind early-fitting, revealing that deep-synchronized poisoned samples are harder to be detected. Therefore, we propose a new attack called Deep Backdoor Attack (DBA), which generates deep-synchronized poisoned samples, making it a more stealthy attack.

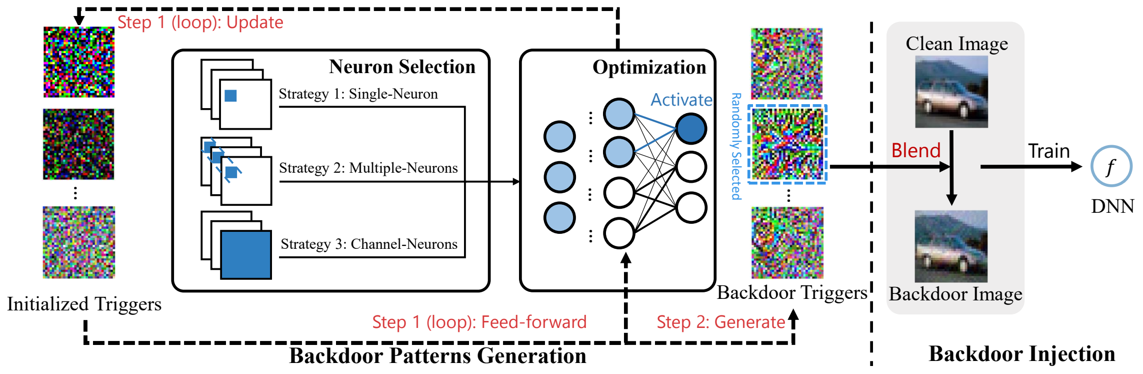

Figure 5: The pipeline of Deep Backdoor Attacks (DBA).

3.1 Towards Deep Synchronization by Activating Deep Neurons

It is non-trivial to craft backdoor samples whose synchronization scores are directly under control. First, the synchronization scores of a sample depend on other samples, where it is difficult to coordinate a group of samples during attacks. Second, synchronization scores depend on the target model's architecture, which is usually unknown in advance. Therefore, it is infeasible to apply traditional end-to-end optimization methods to solve the problem. Inspired by "magic must defeat magic", we propose to adopt feature inversion [24, 38] to reverse engineer backdoor patterns from neural networks. The resultant patterns, when added to clean samples, will produce deep-synchronized samples for backdoor attacks. The overall procedure is illustrated in Figure 5. Formally, 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:

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

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 ascent on $\boldsymbol{x}'$, where $\boldsymbol{x}'$ is initialized randomly. The generated pattern $\boldsymbol{\delta}$ maximizes the activation of deep neurons, if l is set to be large.

There are several nice properties for the backdoor trigger patterns generated above. First, $\boldsymbol{\delta}$ triggers deep synchronization once used as backdoor patterns, as demonstrated in the last row of Figure 4. Intuitively, for a neural network g trained on clean data, its deep layers already encode complex patterns. Thus, 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. Second, the choice of g is largely independent of the target model to be attacked (i.e., the attack is transferable), which is validated in experiments. Third, after fixing the neurons s, by giving different random initialization to $\boldsymbol{x}'$ prior to solving the above optimization problem, we could generate diverse trigger patterns which are harder to be detected by humans.

3.2 Details of Designing Effective DBA

In this part, we further introduce the implementation details to ensure the effectiveness of DBA.

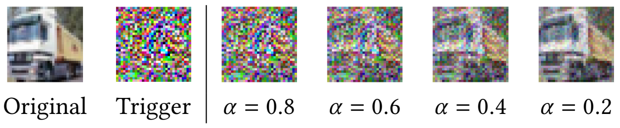

Figure 6: From left to right, the trigger pattern δ is blended with the original image under different blend ratios α.

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 result pattern varies if initialized differently, but they all contain the same semantic information since they are from the same set of neurons. The diverse initialization guarantees that we fully distill the complex information encoded in deep neurons, and avoids the target model to memorize a single pattern. In this way, **the deep synchronization is encouraged** by activating the chosen deep neurons, and **shallow synchronization is suppressed** by diversifying initialization. In Figure 4, DBA samples have similar synchronization scores as clean samples in shallow layers, meaning that shallow synchronization is suppressed in DBA.

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. Different neuron selections are illustrated in Figure 5. 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 3, which shows that the multiple-neurons strategy outperforms the other two strategies.

Table 1: Clean Accuracy (%), Backdoor Accuracy (%), and Isolation Precision (%) of different attack methods.

Dataset	Cifar10			GTSRB			Cifar100		
Attacks	CA (%) ↑	BA (%) ↑	IP (%) ↓	CA (%) ↑	BA (%) ↑	IP (%) ↓	CA (%) ↑	BA (%) ↑	IP (%) ↓
BadNets	$85.12_{\pm0.07}$	$99.93_{\pm0.01}$	$91.33_{\pm2.62}$	$98.46_{\pm0.21}$	$99.99_{\pm0.01}$	$91.07_{\pm0.02}$	$63.72_{\pm0.10}$	$100.00_{\pm0.00}$	$89.40_{\pm5.40}$
Blend	$84.78_{\pm0.31}$	$99.97_{\pm0.02}$	$89.33_{\pm2.05}$	$98.26_{\pm0.19}$	$99.98_{\pm0.01}$	$85.18_{\pm0.05}$	$61.80_{\pm0.22}$	$99.94_{\pm0.01}$	$73.40_{\pm10.00}$
SIG	$84.67_{\pm0.48}$	$99.91_{\pm0.01}$	$86.75_{\pm0.05}$	$97.36_{\pm0.30}$	$100.00_{\pm0.00}$	$74.69_{\pm7.56}$	$60.79_{\pm0.48}$	$99.99_{\pm0.01}$	$54.67_{\pm6.13}$
Dynamic	$83.56_{\pm0.38}$	$99.86_{\pm0.02}$	$90.26_{\pm1.33}$	$97.14_{\pm0.26}$	$99.41_{\pm0.30}$	$86.20_{\pm0.05}$	$61.65_{\pm0.29}$	$99.99_{\pm0.01}$	$91.33_{\pm3.10}$
FC	$82.14_{\pm0.54}$	$99.98_{\pm0.02}$	$63.99_{\pm10.34}$	$95.15_{\pm0.36}$	$99.41_{\pm0.30}$	$64.08_{\pm6.79}$	$58.84_{\pm0.37}$	$96.20_{\pm0.81}$	$27.00_{\pm6.68}$
Refool	$82.06_{\pm0.17}$	$68.10_{\pm2.74}$	$86.07_{\pm4.34}$	$93.51_{\pm0.08}$	$97.12_{\pm0.25}$	$74.60_{\pm7.50}$	$59.57_{\pm0.32}$	$91.89_{\pm0.44}$	$69.67_{\pm3.20}$
WaNet	$82.31_{\pm0.19}$	$99.74_{\pm0.01}$	$71.91_{\pm0.13}$	$96.40_{\pm0.31}$	$100.00_{\pm0.00}$	$67.90_{\pm5.80}$	$57.75_{\pm1.60}$	$99.99_{\pm0.02}$	$67.30_{\pm3.45}$
DBA (this work)	$85.32_{\pm0.37}$	$98.65_{\pm0.17}$	**25.37**$_{\pm11.69}$	$97.78_{\pm0.43}$	$99.98_{\pm0.01}$	**1.26**$_{\pm0.06}$	$61.21_{\pm0.19}$	$99.98_{\pm0.01}$	**11.00**$_{\pm15.55}$

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 $\boldsymbol{x}_b = \alpha \cdot \boldsymbol{\delta} + (1-\alpha)\boldsymbol{x}_c$. Figure 6 presents the poisoned sample with different blend ratios 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 $\{(\boldsymbol{x}_b, y_b)\}$. Note that the number of poisoned samples is limited to a small portion of the entire dataset. After training on the clean and poisoned samples, the model f_θ learns both the original classification task and the backdoor injection task:

$$\min_{\theta} \ \mathbb{E}_{(\boldsymbol{x}_c, y_c)\sim\mathcal{D}_{clean}} \ell(f_\theta, \boldsymbol{x}_c, y_c) + \mathbb{E}_{(\boldsymbol{x}_b, y_b)\sim\mathcal{D}_{poi}} \ell(f_\theta, \boldsymbol{x}_b, y_b), \quad (6)$$

where $\ell(\cdot)$ denotes the loss on a single sample. 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, conduct ablation studies on the impact of hyperparameters of DBA, evaluate the robustness of DBA to other defense methods, and propose a tentative defense method to alleviate the DBA.

4.1 Experimental Setups

4.1.1 Dataset and Model Architecture. We evaluate the effectiveness of DBA on three popular datasets: CIFAR10, CIFAR100 [17], and GTSRB [41]. We adopt WideResNet-16 [53] for CIFAR10 and GTSRB. For CIFAR100, as the task is more difficult, we resort to an improved version[1] of ResNet-18 [14] for CIFAR100. The architectural details of these models are presented in Table 2. All experiments are run with a 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 a momentum of 0.9 and weight decay of 0.0005. For Equation 4, the default model g is chosen as a pre-trained WideResNet-16.

[1]https://github.com/weiaicunzai/pytorch-CIFAR100

Table 2: DNNs and datasets used in experiments.

Dataset	Subjects	#Classes	#Train Images	Classifier
Cifar10	General Objects	10	50,000	WideResNet-16
Cifar100	General Objects	100	50,000	ResNet-18
GTSRB	Traffic Signs	43	39,209	WideResNet-16

4.1.2 Baseline Methods. We compare DBA with seven representative backdoor attack methods, including five dirty-label attacks: BadNets [9], Blend Attack [3], Dynamic Attack [26], Refool Attack [23], and WaNet [25]; one clean-label attack: Signal Attack (SIG) [1]; and one feature-space attack: Feature Collision (FC) [33].

4.1.3 Evaluation Metrics. Two classical metrics [19], 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 an additional metric, Isolation Precision (IP). IP measures the portion of poisoned samples that can be detected by the early-fitting-based defense [20]. A lower IP value means that more poisoned samples will evade detection. The details are as follows:

• **Clean Accuracy (CA)**: The portion of correctly classified clean samples. CA $= \frac{\text{\#Correctly Classified Clean Samples}}{\text{\#Total Samples}}$.

• **Backdoor Accuracy (BA)**: The portion of correctly classified poisoned samples. BA $= \frac{\text{\#Correctly Classified Poisoned Samples}}{\text{\#Total Samples}}$.

• **Isolation Precision (IP)**: The portion of poisoned samples that can be isolated from the datset. IP $= \frac{\text{\#Isolated Poisoned Samples}}{\text{\#Total Poisoned Samples}}$.

Note that in the experiment, we calculate isolation precision with the following procedure: Given the injection ratio is η, we isolate η samples from the entire dataset with the lowest loss value. The isolation process is done at the end of the first five training epochs, where the maximal isolation precision is reported.

4.2 Attack Effectiveness Evaluation

Table 1 shows that DBA consistently achieves a relatively good performance in CA and BA, indicating that DBA is effective in launching backdoor attacks on the victim model. Moreover, DBA

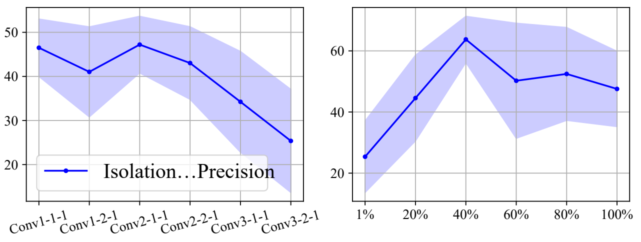

Figure 7: Isolation Precision with triggers inverted from: a) neurons of different depths, and b) different blend ratios α.

Table 3: DBA performance with different neuron selection strategies.

Neuron Selection	CA (%) ↑	BA (%) ↑	IP (%) ↓
Single-neuron (Block3 Conv2 (63, 4, 4))	85.75(±0.35)	98.70$_{\pm 0.20}$	53.58(±12.04)
Multiple-neurons (Block3 Conv2 ([:], 4, 4))	85.32(±0.37)	98.65$_{\pm 0.17}$	**25.37(±11.13)**
Channel-neurons (Block3 Conv2 (63, [:], [:]))	84.77(±0.65)	98.64$_{\pm 0.16}$	33.64(±21.64)

shows a considerably lower IP ratio compared to other attack methods, demonstrating its effectiveness in evading early-fitting-based defenses. Specifically, DBA achieves an average IP of 25.37%, with a standard deviation of 11.69%. In contrast, for most other attack methods, the IP scores are higher than 60%. Note that DBA is implemented with the multiple-neurons strategy as above.

4.3 Ablation Studies and Hyperparameter Analysis

The hyperparameters for DBA include 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.

Neuron Selection Strategy. In Table 3, we compare the performance of DBA under three neuron selection strategies. 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, [:], [:]). The multiple-neurons strategy is observed to achieve the best performance in IP (25.37 %) 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. We generate triggers from neurons in different CNN layers of the model g. The results are shown in Figure 7 (left), where the x-axis extends along the mode layer depth. As we can see, DBA attacks achieve an increasingly lower average IP with a deeper inverted layer. It demonstrates that triggers are embedded with more complex patterns and are harder to detect if they are generated from neurons of 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 7 (right). As Figure 7 (right) shows, the IP is heterogeneous when clean images are blended with different blend ratios. Specifically, when using a very low blend ratio (1%), DBA could still achieve a low IP score (25.37% on average). This implies that attacks of DBA are effective when the trigger patterns are more stealthy.

4.3.3 Trigger Pattern Transferability. Recall that the DBA triggers are generated with a model $g \in \mathcal{F}$ and the attack is launched over another model $f \in \mathcal{F}$, where $\mathcal{F}$ denotes the collection of models. To evaluate the transferability of the DBA, we specifically consider three classical models in image classification, including ResNet34 [14], EfficientNet [44], and VGG16 [39]. Table 4 presents our transferability result on the CIFAR10 dataset, where the model name in each row denotes the model g used in trigger generation, and the model name in each column denotes the model f. As we can see, the effectiveness of the DBA attack is highly independent of the choice of model g.

4.3.4 Injection Ratio. Here, we evaluate the effectiveness of DBA under different injection ratios η. Specifically, we consider injection ratios ranging from 0.02 to 0.1. The results in Table 5 show that DBA consistently attacks the model and evades the early-fitting phenomenon even under a low injection ratio $\eta = 0.02$.

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) [22], NAD [21], and ABL [20]. As shown in Table 6, CA(No Defense) and

Table 4: Transferability of trigger patterns from one model to attack other models.

Model $g \rightarrow$	Vgg-16			ResNet34			EfficientNet		
Model $f \downarrow$	CA (%)	BA (%)	IP (%)	CA (%)	BA (%)	IP (%)	CA (%)	BA (%)	IP (%)
Vgg-16	85.21$_{\pm 0.28}$	99.12$_{\pm 0.22}$	19.88$_{\pm 0.33}$	85.84$_{\pm 0.16}$	98.89$_{\pm 0.12}$	21.46$_{\pm 0.15}$	84.95$_{\pm 0.22}$	98.70$_{\pm 0.13}$	24.13$_{\pm 0.22}$
ResNet34	85.11$_{\pm 0.27}$	98.59$_{\pm 0.03}$	23.54$_{\pm 0.21}$	85.81$_{\pm 0.06}$	98.69$_{\pm 0.32}$	13.68$_{\pm 0.26}$	85.29$_{\pm 0.30}$	98.53$_{\pm 0.21}$	13.71$_{\pm 0.51}$
EfficientNet	85.36$_{\pm 0.27}$	98.79$_{\pm 0.04}$	21.05$_{\pm 0.32}$	85.56$_{\pm 0.34}$	98.81$_{\pm 0.06}$	25.24$_{\pm 0.24}$	86.08$_{\pm 0.15}$	98.67$_{\pm 0.04}$	16.81$_{\pm 0.27}$

Table 5: DBA attack with different injection ratio η.

Dataset	Metric	0.02	0.04	0.06	0.08	0.10
Cifar10	CA	85.24$_{\pm 0.13}$	84.96$_{\pm 0.28}$	85.08$_{\pm 0.21}$	84.87$_{\pm 0.22}$	85.32$_{\pm 0.37}$
	BA	96.87$_{\pm 0.24}$	98.44$_{\pm 0.31}$	98.98$_{\pm 0.19}$	98.33$_{\pm 0.16}$	98.65$_{\pm 0.17}$
	IP	2.33$_{\pm 1.36}$	4.55$_{\pm 1.24}$	24.07$_{\pm 5.16}$	25.9$_{\pm 8.13}$	25.37$_{\pm 11.13}$
GTSRB	CA	97.96$_{\pm 0.39}$	97.42$_{\pm 0.43}$	96.97$_{\pm 0.36}$	98.10$_{\pm 0.42}$	97.78$_{\pm 0.43}$
	BA	96.87$_{\pm 0.57}$	98.44$_{\pm 0.25}$	98.98$_{\pm 0.15}$	98.33$_{\pm 0.09}$	99.98$_{\pm 0.01}$
	IP	2.75$_{\pm 0.50}$	3.22$_{\pm 0.27}$	5.13$_{\pm 0.21}$	3.15$_{\pm 0.10}$	1.26$_{\pm 0.06}$
Cifar100	CA	61.13$_{\pm 0.35}$	61.40$_{\pm 0.24}$	61.13$_{\pm 0.19}$	61.24$_{\pm 0.16}$	61.21$_{\pm 0.19}$
	BA	91.93$_{\pm 0.08}$	94.59$_{\pm 0.12}$	98.74$_{\pm 0.05}$	99.94$_{\pm 0.03}$	99.98$_{\pm 0.01}$
	IP	15.23$_{\pm 4.88}$	16.21$_{\pm 4.50}$	15.74$_{\pm 6.27}$	21.32$_{\pm 9.23}$	11.00$_{\pm 15.55}$

Table 6: CA and BA with/without applying the defense methods on models attacked by DBA.

Dataset	Defense	CA (No Defense)	BA (No Defense)	CA(defense) ↓	BA(defense) ↑
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$_{\pm 12.61}$

C

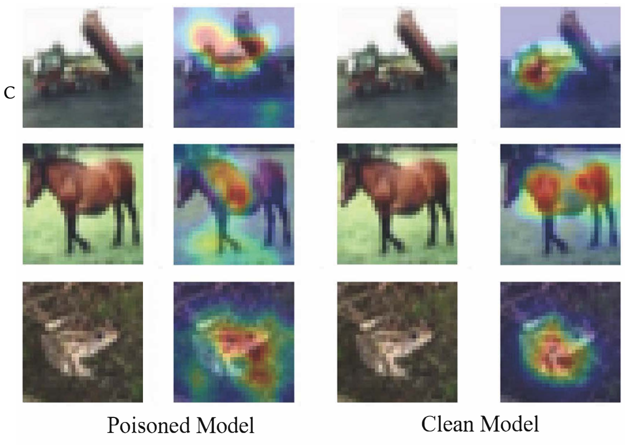

Figure 8: GradCAM for DBA-poisoned model input with poisoned image and clean Model input with clean image.

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 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 6, 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 resistant to SOTA defense methods.

4.5 Network Inspection

We apply the post-hoc interpretation method GradCAM [32] to help inspect the neural network model and check if the backdoor patterns could be identified. As pointed out by [51], patch-based trigger patterns are easy to be located by GradCAM. However, since our trigger pattern is blended with the entire image and is more stealthy compared to patch-based trigger patterns, it is naturally immune to the detection method. We plot the GradCAM heatmaps for a poisoned model attacked by DBA and a clean model, respectively, in Figure 8. The poisoned model is input with a poisoned sample $\boldsymbol{x}'$, and the label is chosen as the target label y'. The clean model is input with a clean sample $\boldsymbol{x}$, and the label is chosen as the one predicted by the model. As noticed, the heatmaps for the poisoned model are very similar to those of the clean model.

4.6 An Isolation-based Defense Method

To defend against our proposed DBA, we propose an influence-based isolation method, which can detect poisoned samples with higher accuracy. Motivated by [27], if we set $(\boldsymbol{x}, y) = (\boldsymbol{x}', y')$ 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 higher self-influence score because most 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, 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 choosing top-k samples with a high self-influence score.

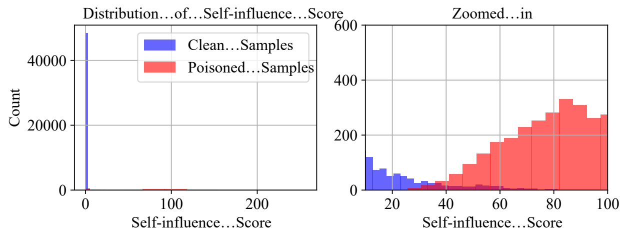

Figure 9: Histogram of the self-influence score for poisoned samples and clean samples respectively. The isolation precision for the deep backdoor attack is 90.36%.

Figure 9 presents the effectiveness of the defense method against the proposed Deep Backdoor Attack. As shown, the majority of the self-influence scores on the poisoned samples (red) are 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 precision for detecting the poisoned samples is 90.36%, demonstrating the effectiveness of backdoor detection.

5 RELATED WORK

5.1 Backdoor Attacks

In general, backdoor attacks can be categorized into three types: 1) data poisoning attacks [3, 9, 11, 23, 36, 52], 2) training poisoning attacks [31, 37, 56], and 3) model poisoning attacks [30, 47]. In this paper, we solely focus on data poisoning attacks as this is the most common setting. Data-poisoning backdoor attacks aim to poison the dataset with trigger patterns. Specifically, they inject trigger patterns into the victim samples and re-assign the ground-truth label to a target label predefined by the attackers. Recent research can be divided into two categories on making the backdoor attacks more stealthy to enhance their practicality. The first one [3, 5, 23, 25, 28] aims to make the trigger pattern less visible to human eyes. For example, [3] blends the clean images with random pixels. [23] uses the natural reflection to construct the backdoor trigger. The other direction [34, 40, 55] aims to make the training process less noticeable. For example, [34] proposes a clean-label attack, which perturbs the clean images without changing their labels. However, most of these backdoor attacks are easily detected by an early-fitting phenomenon [20], thus weakening the backdoor attacks threats.

5.2 Backdoor Defenses

Various defense methods have been proposed to mitigate the threat from the backdoor. As in [19], we categorize existing defense methods into five categories. First, detection-based defenses [7, 8, 10, 12, 13, 16, 50] aim to detect whether the backdoor exists in the model. Second, preprocessing-based defenses [4] introduce a preprocessing module before the training procedure so that triggers can be inactivated. Third, defenses based on model reconstruction [22, 49, 54, 57] directly eliminate the effect of backdoors by adjusting the model weights or network structures. In this way, even if the trigger pattern appears, the reconstructed model will still perform normally as the backdoor is already moved. Fourth, defenses based on trigger synthesis [2, 29, 35, 46] first reverse engineer the trigger patterns and then suppress the trigger's effects.

Lastly, training sample filtering-based defenses [15, 20, 48] work by first filtering poisoned samples from the training dataset, then training the network exclusively in the rest of the dataset.

6 CONCLUSION AND FUTURE WORK

In this work, we interpret the early-fitting phenomenon from a novel perspective of data synchronization. Based on the theoretical analysis, we further propose an enhanced backdoor attacks method: Deep Backdoor Attack. Experiments across various datasets demonstrate the effectiveness of our method. Despite the satisfactory performance, there are still some further work to be done. First, our method relies on the assumption that an additional model is accessible, which might not always be feasible in certain specified applications. How can we relax the assumption? Second, customizing the DBA for text and graph data could also be interesting.

A ANALYSIS OF THEOREM 1

As defined in Equation 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 we could have,

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

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 $\eta = \frac{|\mathcal{D}_{poi}|}{|\mathcal{D}|}$.

B ANALYSIS OF HYPOTHESIS 1

Intuition for Hypothesis 1. In order 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.

Analysis. Here we only consider a simple case and left the general case for future works. Assume 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:

$$f^{(l)}(x) = \theta^{(l)}g^{(l-1)}(x), g^{(l)}(x) = \sigma^l(f^{(l)}(x)), \forall 1 \le l \le L, \tag{9}$$

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

$$f_\theta(\boldsymbol{x}) = g^{(L)}(x) \tag{10}$$

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

$$\begin{aligned}r((\boldsymbol{x}', y'); (\boldsymbol{x}, y)) &\approx \beta\langle\frac{\partial\ell_\theta(\boldsymbol{x}', y')}{\partial\theta}, \frac{\partial\ell_\theta(\boldsymbol{x}, y)}{\partial\theta}\rangle \\ &= \beta\langle\frac{\partial\ell_\theta(\boldsymbol{x}', y')}{\partial f_\theta(\boldsymbol{x}')}\cdot\frac{\partial f_\theta(\boldsymbol{x}')}{\partial\theta}, \frac{\partial\ell_\theta(\boldsymbol{x}, y)}{\partial f_\theta(\boldsymbol{x})}\cdot\frac{\partial f_\theta(\boldsymbol{x})}{\partial\theta}\rangle.\end{aligned} \tag{11}$$

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 11 can be further decomposed into the following form,

$$\begin{aligned}&\langle\frac{\partial\ell_\theta(\boldsymbol{x}, y)}{\partial f_\theta(\boldsymbol{x})}\cdot\frac{\partial f_\theta(\boldsymbol{x})}{\partial\theta}, \frac{\partial\ell_\theta(\boldsymbol{x}', y')}{\partial f_\theta(\boldsymbol{x}')}\cdot\frac{\partial f_\theta(\boldsymbol{x}')}{\partial\theta}\rangle \\ =&\frac{\partial\ell_\theta(\boldsymbol{x}, y)}{\partial f_\theta(\boldsymbol{x})}\cdot\frac{\partial\ell_\theta(\boldsymbol{x}', y')}{\partial f_\theta(\boldsymbol{x}')}\langle\frac{\partial f_\theta(\boldsymbol{x})}{\partial\theta}, \frac{\partial f_\theta(\boldsymbol{x}')}{\partial\theta}\rangle \\ =&\frac{\partial\ell_\theta(\boldsymbol{x}, y)}{\partial f_\theta(\boldsymbol{x})}\cdot\frac{\partial\ell_\theta(\boldsymbol{x}', y')}{\partial f_\theta(\boldsymbol{x}')}\sum_{l=1}^{L}\langle\frac{\partial f_\theta(\boldsymbol{x})}{\partial f^l_\theta(\boldsymbol{x})}\cdot g^{l-1}_\theta(\boldsymbol{x})^T, \frac{\partial f_\theta(\boldsymbol{x}')}{\partial f^l_\theta(\boldsymbol{x}')}\cdot g^{l-1}_\theta(\boldsymbol{x}')^T\rangle \\ =&\underbrace{\frac{\partial\ell_\theta(\boldsymbol{x}, y)}{\partial f_\theta(\boldsymbol{x})}\frac{\partial\ell_\theta(\boldsymbol{x}', y')}{\partial f_\theta(\boldsymbol{x}')}}_{\text{Part 1}}\cdot\sum_{l=1}^{L}\underbrace{\langle g^{l-1}_\theta(\boldsymbol{x}), g^{l-1}_\theta(\boldsymbol{x}')\rangle}_{\text{Part 2}}\cdot\underbrace{\langle\frac{\partial f_\theta(\boldsymbol{x})}{\partial f^l_\theta(\boldsymbol{x})}, \frac{\partial f_\theta(\boldsymbol{x}')}{\partial f^l_\theta(\boldsymbol{x}')}\rangle}_{\text{Part 3}}.\end{aligned} \tag{12}$$

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. Therefore, it is obvious that synchronization score is contributory to the loss reduction.

ACKNOWLEDGMENTS

The work is, in part, supported by NSF (#IIS-2223768). The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

REFERENCES

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

[2] Weixin Chen, Baoyuan Wu, and Haoqian Wang. 2022. Effective Backdoor Defense by Exploiting Sensitivity of Poisoned Samples. In *Advances in Neural Information Processing Systems*, Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (Eds.). https://openreview.net/forum?id=AsH-Tx2U0Ug

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

[4] Bao Gia Doan, Ehsan Abbasnejad, and Damith C. Ranasinghe. 2020. Februus: Input Purification Defense Against Trojan Attacks on Deep Neural Network Systems. In *Annual Computer Security Applications Conference* (Austin, USA) *(ACSAC '20)*. Association for Computing Machinery, New York, NY, USA, 897–912. https://doi.org/10.1145/3427228.3427264

[5] Khoa Doan, Yingjie Lao, Weijie Zhao, and Ping Li. 2021. LIRA: Learnable, Imperceptible and Robust Backdoor Attacks. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 11946–11956. https://doi.org/10.1109/ICCV48922.2021.01175

[6] Khoa D. Doan, Yingjie Lao, and Ping Li. 2022. Marksman Backdoor: Backdoor Attacks with Arbitrary Target Class. https://doi.org/10.48550/ARXIV.2210.09194

[7] Yinpeng Dong, Xiao Yang, Zhijie Deng, Tianyu Pang, Zihao Xiao, Hang Su, and Jun Zhu. 2021. Black-box Detection of Backdoor Attacks with Limited Information and Data. https://doi.org/10.48550/ARXIV.2103.13127

[8] Yansong Gao, Chang Xu, Derui Wang, Shiping Chen, Damith C. Ranasinghe, and Surya Nepal. 2019. STRIP: A Defence Against Trojan Attacks on Deep Neural Networks. (2019). https://doi.org/10.48550/ARXIV.1902.06531

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

[10] Zihan Guan, Mengnan Du, and Ninghao Liu. 2023. XGBD: Explanation-Guided Graph Backdoor Detection. *26th European Conference on Artificial Intelligence (ECAI)* (2023).

[11] Zihan Guan, Mengxuan Hu, Zhongliang Zhou, Jielu Zhang, Sheng Li, and Ninghao Liu. 2023. Badsam: Exploring security vulnerabilities of sam via backdoor attacks. *arXiv preprint arXiv:2305.03289* (2023).

[12] Junfeng Guo, Ang Li, and Cong Liu. 2021. AEVA: Black-box Backdoor Detection Using Adversarial Extreme Value Analysis. https://doi.org/10.48550/ARXIV.2110.14880

[13] Junfeng Guo, Yiming Li, Xun Chen, Hanqing Guo, Lichao Sun, and Cong Liu. 2023. Scale-up: An efficient black-box input-level backdoor detection via analyzing scaled prediction consistency. *arXiv preprint arXiv:2302.03251* (2023).

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. https://doi.org/10.48550/ARXIV.1512.03385

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

[16] Xijie Huang, Moustafa Alzantot, and Mani Srivastava. 2019. NeuronInspect: Detecting Backdoors in Neural Networks via Output Explanations. https://doi.org/10.48550/ARXIV.1911.07399

[17] Alex Krizhevsky. 2009. *Learning multiple layers of features from tiny images*. Technical Report.

[18] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324. https://doi.org/10.1109/5.726791

[19] Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. 2020. Backdoor Learning: A Survey. https://doi.org/10.48550/ARXIV.2007.08745

[20] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. 2021. Anti-Backdoor Learning: Training Clean Models on Poisoned Data. In *NeurIPS*.

[21] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. 2021. Neural Attention Distillation: Erasing Backdoor Triggers from Deep Neural Networks. In *ICLR*.

[22] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2018. Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks. https://doi.org/10.48550/ARXIV.1805.12185

[23] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. 2020. Reflection Backdoor: A Natural Backdoor Attack on Deep Neural Networks. https://doi.org/10.48550/ARXIV.2007.02343

[24] Aravindh Mahendran and Andrea Vedaldi. 2014. Understanding Deep Image Representations by Inverting Them. https://doi.org/10.48550/ARXIV.1412.0035

[25] Anh Nguyen and Anh Tran. 2021. WaNet – Imperceptible Warping-based Backdoor Attack. https://doi.org/10.48550/ARXIV.2102.10369

[26] Tuan Anh Nguyen and Anh Tran. 2020. Input-Aware Dynamic Backdoor Attack. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 3454–3464. https://proceedings.neurips.cc/paper/2020/file/234e691320c0ad5b45ee3c96d0d7b8f8-Paper.pdf

[27] Garima Pruthi, Frederick Liu, Mukund Sundararajan, and Satyen Kale. 2020. Estimating Training Data Influence by Tracing Gradient Descent. https://doi.org/10.48550/ARXIV.2002.08484

[28] Xiangyu Qi, Tinghao Xie, Yiming Li, Saeed Mahloujifar, and Prateek Mittal. 2023. Revisiting the Assumption of Latent Separability for Backdoor Defenses. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=_wSHsgrVali

[29] Ximing Qiao, Yukun Yang, and Hai Li. 2019. Defending Neural Backdoors via Generative Distribution Modeling. https://doi.org/10.48550/ARXIV.1910.04749

[30] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. 2019. TBT: Targeted Neural Network Attack with Bit Trojan. https://doi.org/10.48550/ARXIV.1909.05193

[31] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. 2019. Hidden Trigger Backdoor Attacks. https://doi.org/10.48550/ARXIV.1910.00033

[32] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2019. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. *International Journal of Computer Vision* 128, 2 (oct 2019), 336–359.

[33] Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. 2018. Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks. https://doi.org/10.48550/ARXIV.1804.00792

[34] Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. 2018. Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks. https://doi.org/10.48550/ARXIV.1804.00792

[35] Guangyu Shen, Yingqi Liu, Guanhong Tao, Shengwei An, Qiuling Xu, Siyuan Cheng, Shiqing Ma, and Xiangyu Zhang. 2021. Backdoor Scanning for Deep Neural Networks through K-Arm Optimization. https://doi.org/10.48550/ARXIV.2102.05123

[36] Jiawen Shi, Yixin Liu, Pan Zhou, and Lichao Sun. 2023. BadGPT: Exploring Security Vulnerabilities of ChatGPT via Backdoor Attacks to InstructGPT. *arXiv preprint arXiv:2304.12298* (2023).

[37] Ilia Shumailov, Zakhar Shumaylov, Dmitry Kazhdan, Yiren Zhao, Nicolas Papernot, Murat A. Erdogdu, and Ross Anderson. 2021. Manipulating SGD with Data Ordering Attacks. https://doi.org/10.48550/ARXIV.2104.09667

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

[39] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. https://doi.org/10.48550/ARXIV.1409.1556

[40] Hossein Souri, Liam Fowl, Rama Chellappa, Micah Goldblum, and Tom Goldstein. 2021. Sleeper Agent: Scalable Hidden Trigger Backdoors for Neural Networks Trained from Scratch. https://doi.org/10.48550/ARXIV.2106.08970

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

[42] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. arXiv:1409.3215 [cs.CL]

[43] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. Rethinking the Inception Architecture for Computer Vision. arXiv:1512.00567 [cs.CV]

[44] Mingxing Tan and Quoc Le. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 6105–6114. https://proceedings.mlr.press/v97/tan19a.html

[45] Ruixiang Tang, Mengnan Du, Ninghao Liu, Fan Yang, and Xia Hu. 2020. An embarrassingly simple approach for trojan attack in deep neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 218–228.

[46] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. 2019. Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks. In *2019 IEEE Symposium on Security and Privacy (SP)*. 707–723. https://doi.org/10.1109/SP.2019.00031

[47] Yajie Wang, Kongyang Chen, Yu'an Tan, Shuxin Huang, Wencong Ma, and Yuanzhang Li. 2022. Stealthy and Flexible Trojan in Deep Learning Framework. *IEEE Transactions on Dependable and Secure Computing* (2022), 1–1. https://doi.org/10.1109/TDSC.2022.3164073

[48] Zhenting Wang, Hailun Ding, Juan Zhai, and Shiqing Ma. 2022. Training with More Confidence: Mitigating Injected and Natural Backdoors During Training. In *Advances in Neural Information Processing Systems*, Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (Eds.). https://openreview.net/forum?id=yNPsd3oG_s

[49] Dongxian Wu and Yisen Wang. 2021. Adversarial Neuron Pruning Purifies Backdoored Deep Models. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 16913–16925. https://proceedings.neurips.

cc/paper/2021/file/8cbe9ce23f42628c98f80fa0fac8b19a-Paper.pdf

[50] Zhen Xiang, David J. Miller, and George Kesidis. 2022. Post-Training Detection of Backdoor Attacks for Two-Class and Multi-Attack Scenarios. https://doi.org/10.48550/ARXIV.2201.08474

[51] Kaidi Xu, Sijia Liu, Pin-Yu Chen, Pu Zhao, and Xue Lin. 2020. Defending against Backdoor Attack on Deep Neural Networks. https://doi.org/10.48550/ARXIV.2002.12162

[52] Zenghui Yuan, Yixin Liu, Kai Zhang, Pan Zhou, and Lichao Sun. 2023. Backdoor attacks to pre-trained unified foundation models. *arXiv preprint arXiv:2302.09360* (2023).

[53] Sergey Zagoruyko and Nikos Komodakis. 2016. Wide Residual Networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, Edwin R. Hancock Richard C. Wilson and William A. P. Smith (Eds.). BMVA Press, Article 87, 12 pages. https://doi.org/10.5244/C.30.87

[54] Yi Zeng, Si Chen, Won Park, Zhuoqing Mao, Ming Jin, and Ruoxi Jia. 2021. Adversarial Unlearning of Backdoors via Implicit Hypergradient. In *International Conference on Learning Representations*.

[55] Yi Zeng, Minzhou Pan, Hoang Anh Just, Lingjuan Lyu, Meikang Qiu, and Ruoxi Jia. 2022. Narcissus: A practical clean-label backdoor attack with limited information. *arXiv preprint arXiv:2204.05255* (2022).

[56] Yi Zeng, Won Park, Z. Morley Mao, and Ruoxi Jia. 2021. Rethinking the Backdoor Attacks' Triggers: A Frequency Perspective. https://doi.org/10.48550/ARXIV.2104.03413

[57] Pu Zhao, Pin-Yu Chen, Payel Das, Karthikeyan Natesan Ramamurthy, and Xue Lin. 2020. Bridging Mode Connectivity in Loss Landscapes and Adversarial Robustness. https://doi.org/10.48550/ARXIV.2005.00060