Exploring the Orthogonality and Linearity of Backdoor Attacks

Kaiyuan Zhang*, Siyuan Cheng*, Guangyu Shen, Guanhong Tao,
Shengwei An, Anuran Makur, Shiqing Ma†, Xiangyu Zhang
Purdue University, †University of Massachusetts at Amherst

Abstract*—Backdoor attacks embed an attacker-chosen pattern into inputs to cause model misclassification. This security threat to machine learning has been a long concern. There are a number of defense techniques proposed by the community. *Do they work for a large spectrum of attacks?

As we argue that they are significant and prevalent in contemporary research, and we conduct a systematic study on 14 attacks and 12 defenses. Our empirical results show that existing defenses often fail on certain attacks. To understand the reason, we study the characteristics of backdoor attacks through theoretical analysis. Particularly, we formulate backdoor poisoning as a continual learning task, and introduce two key properties: *orthogonality* and *linearity*. These two characteristics in-depth explain how backdoors are learned by models from a theoretical perspective. This helps to understand the reason behind the failure of various defense techniques. Through our study, we highlight open challenges in defending against backdoor attacks and provide future directions.

1. Introduction

Backdoor attacks aim to mislead machine learning models by introducing a specialized trigger pattern into inputs. The triggers can cause model misclassification to a target label. There are a variety of backdoor attacks, such as patch attack [12], [13], blend attack [17], [18], filter attack [20], etc. They design different trigger patterns to make attacks more robust and stealthy. For example, filter attack applies an Instagram filter on input images, which are visually similar to the original inputs.

In order to mitigate the backdoor threat, researchers have proposed different defense techniques. They mainly fall into three categories: model detection [1], [2], [3], backdoor removal [4], [5], [6], [7], and input detection [8], [9], [10], [11]. Model detection determines whether a model is injected with backdoors. A common approach in this category is trigger inversion [1], [3], [25] that reverse-engineers a trigger pattern resembling the injected one. Backdoor removal aims to eliminate backdoors injected in poisoned models. Input detection identifies and rejects input samples with the trigger. Existing defense approaches are usually evaluated on a subset of backdoor attacks. *Are those defenses generalizable to all known attacks?*

We systematically study the defense performance of existing methods on a diverse set of backdoor attacks. Specifically, we use 14 well-known attacks and evaluate the performance of 12 representative defenses against them. The results are shown in Table 1. The solid dots denote that the defense methods can successfully defend against corresponding attacks, where the black color means the results are confirmed by the literature and the gray color means they are validated in this paper. The circles indicate the defenses fail. Observe that existing defense techniques fail on at least one backdoor attack evaluated in this paper. But *why do defenses fail?* One may try to look for cues in the design of trigger patterns. For example, BadNets [12] leverages a sticker-like pattern as the trigger. All existing defense techniques work perfectly on this attack as shown in the first row. However, for a fence-like pattern used in SIG [19] (shown in row 8), all evaluated model detection approaches fail, while other types of defenses succeed. From the perspective of the trigger pattern, it does provide sufficient information to explain why one type of defenses work and others do not.

What are the underlying reasons causing defenses to fail on certain backdoor attacks?

This paper aims to answer the above question from a theoretical perspective, with a focus on the motivation that drives our work. Backdoor attacks inject poisoned samples with trigger to model training data. The learning process involves both the main task (e.g. correctly classifying dog images) and the backdoor task (e.g. recognizing dog images with trigger as cat). We observe that the backdoor task is quickly learned by the victim model (using a very few training epochs), much faster than the main task. This indicates a two-stage learning process. We hence formulate backdoor attacks as a continual learning problem and theoretically analyze the characteristics of backdoor attacks. In particular, we identify two key properties in backdoor attacks through our analysis: *orthogonality* and *linearity*. Orthogonality illustrates the backdoor behavior minimally interferes with the model's performance on clean data. This is characterized by the perpendicular relationship between backdoor gradients and clean gradients during the training process. Linearity specifies the linear relationship of poisoned inputs and the output target. There exists a hyperplane that separates the model decision space into two disjoint regions, where the backdoor behavior is in one region and the clean behavior is in the other. Our work is primarily motivated by the observation that the effectiveness of many attacks and their countermeasures significantly depends on two key properties we aim to explore: *orthogonality* and *lineality*.

*Equal contribution.

Table 1: A Summary of Existing Attacks and Defenses

Attack		Model Detection			Backdoor Mitigation				Input Detection			
		NC [1]	Pixel [2]	ABS [3]	Fine-Pruning [4]	NAD [5]	ANP [6]	SEAM [7]	AC [8]	SS [9]	SPECTRE [10]	SCAn [11]
Patch	BadNets [12]	●	●	●	●	●	●	●	●	●	●	●
	TrojanNN [13]	●	●	●	◍	●	●	●	◍	○	◍	●
	Dynamic [14]	◍	◍	◍	◍	◍	○	◍	○	◍	◍	○
	CL [15]	◍	◍	◍	◍	●	●	◍	○	○	◍	◍
	Input-aware [16]	○	○	○	◍	◍	●	◍	◍	◍	◍	◍
Blend	Reflection [17]	○	○	○	○	○	○	●	○	○	○	○
	Blend [18]	◍	●	◍	◍	●	◍	◍	◍	○	◍	○
	SIG [19]	○	○	○	○	○	●	◍	◍	◍	◍	◍
Filter	Instagram [3]	○	○	●	◍	◍	◍	●	◍	◍	◍	○
	DFST [20]	○	◍	○	◍	◍	○	◍	◍	◍	◍	○
Invisible	WaNet [21]	○	○	○	◍	◍	◍	◍	◍	◍	◍	○
	Invisible [22]	◍	◍	○	◍	◍	◍	◍	◍	◍	◍	○
	Lira [23]	◍	◍	○	○	○	○	◍	◍	○	◍	○
	Composite [24]	○	○	○	○	○	○	○	○	○	◍	○

●: attacks can be defended, supported by existing works; ◍: attacks can be defended, supported by our experiments; ○: attacks cannot be defended.

These properties underpin our theoretical analysis, providing a fresh perspective to understand the interplay between attacks and defenses. In particular, given an attack and a defense, we could study these two properties to understand if the defense is effective against the attack. Furthermore, existing attacks can be easily enhanced by changing these two properties. In Section 5.3, we launch six attack variations by changing orthogonality and linearity and evaluate how they affect the attack effectiveness. For example, our results in Table 9 show that Label-specific Poisoning will reduce the orthogonality of Patch attacks [12], [13], thus making it more robust against NAD defenses [5]. The two properties theoretically explain how backdoor behaviors are learned by the model and how the poisoned model exhibit such behaviors, regardless of backdoor attack configurations (e.g. trigger patterns).

Based on our theoretical analysis, we conduct an comprehensive study to understand the limitations of existing defenses. In specific, we introduce two metrics according to the two properties. We then propose a set of hypotheses to evaluate performance of 12 existing defenses in relation to orthogonality and linearity. We further explore six factors that impact orthogonality and linearity of backdoor attacks.

The contributions of this paper are summarized in the following.

- We formulate backdoor attacks as a continual learning problem, which provides the theoretical basis for understanding backdoor attacks.
- We identify two key properties, orthogonality and linearity, that shed the light on the nature of backdoor attacks.
- An extensive empirical study is conducted on 14 state-of-the-art backdoor attacks and 12 defense techniques to understand the (in)effectiveness of existing defenses. Our empirical results show that existing defenses are particularly vulnerable to attacks with low orthogonality or low linearity.
- We comprehensively study six factors that impact the performance of backdoor attacks. We find five out of six factors fail existing defenses.

Threat model. This study focuses on training-time backdoor attacks [12], [13], [16], [21], [23] in Deep Neural Networks (DNNs), where the adversary has full control over the training procedure (poison training dataset or manipulate model internals), and provides a model to victim users after training. While we acknowledge other backdoor techniques like architectural backdoor attack [26] and runtime bit-flipping [27], our scope is deliberately focusing on training-time trigger attacks due to their prevalence and extensiveness in current iteration.

Organization. In Section 2 and Section 3, we lay the foundation by introducing essential terminology and offering theoretical analysis. In Section 4, we propose ten hypotheses that guide our empirical investigations, the results of which are elaborated in Section 5. In Section 6, we review related literature. In Section 7, we outline unresolved questions and offer concluding remarks.

2. Preliminaries

In this section, we provide an overview of the core concepts relevant to our theoretical analysis.

Continual Learning. Recently, there has been a rapid growth in interest on characterising continual learning [28], [29], [30], [31], [32]. Essentially, continual learning deals with a sequence of tasks $\{\tau_1, \tau_2, \dots\}$, where each task can be viewed as a separate supervised learning problem. The fundamental challenge, however, lies in the model's capability to learn the new tasks while retaining the knowledge acquired from previous tasks, which is commonly referred to as catastrophic forgetting [33], [34]. Existing works claim that the catastrophic forgetting problem can be addressed in continual learning framework [30], [32].

Neural Tangent Kernel. Neural networks, powerful as their wide deployments in various domains, often pose a challenge in terms of interpretability. This is due to the intricacy of their internal mechanisms, which often make them resemble "black box" models. However, recent advances in Neural

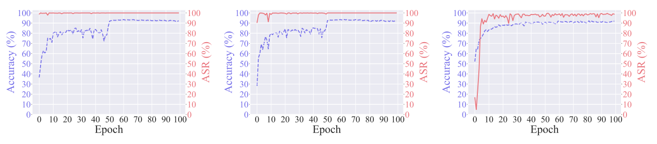
Tangent Kernel (NTK) [35] offers a novel perspective that enhance our understanding of these networks. The NTK framework leverages an intricate approximation of the network function, centered on the principle of Taylor expansion. In this construct, the kernel function is a complex construct formed by the gradients of neural network parameters in relation to the input data. The initial weight configuration of the neural network, denoted as θ_0, serves as the focal point for this approximation. Moreover, the gradient of the network function with respect to the weights at this initial configuration, denoted as $\nabla_\theta f(x, \theta_0)$, is a central element in the NTK framework, as shown in Definition 3.2. One notable characteristic of the NTK approximation is its fixed feature map, represented by $\phi(x) = \nabla_\theta f(x, \theta_0)$, once the initial weights θ_0 have been established. Despite operating within a constant feature space, the NTK approximation allows for a unique optimization of each feature's weight in subsequent training phases, as dictated by the specific training data employed. The theory behind NTK is bridging theoretical neural network insights with practical applications. For a more comprehensive understanding of NTK theory, readers are referred to [30], [32], [35].

3. Theoretical Analysis

In this section, we present a comprehensive framework to capture the inherent orthogonality and linearity characteristics of existing backdoor attacks. We start with the problem setup in Section 3.1. We first study orthogonality of backdoor learning in the context of *Orthogonal Gradient Descent* (OGD) in Section 3.2. Then we broaden the scope to *Stochastic Gradient Descent* (SGD) setting in Section 3.3. Subsequent to this, we pivot our focus to examine the linearity of backdoor attacks under rectifier networks in Section 3.4.

3.1. Problem Setup

In this work, we formalize backdoor learning as a two-task continual learning problem. Our formalization diverges from traditional continual learning [28], [29], [36] where tasks are sequentially presented.



(a) BadNets (b) Blend (c) WaNet

Figure 1: Backdoor Task (red line) vs. Clean Task (blue dotted line) Convergence Rate

Assumption 3.1. ***(Backdoor Learning as Continual Learning).*** *Backdoor learning can be formalized as a two-stage continual learning process: ① an initial rapid learning phase of the backdoor task within a few training epochs, followed by ② a subsequent phase of gradually learning over the clean task.*

We validate Assumption 3.1 from two perspectives: following the backdoor learning threat model and the traditional continual learning paradigm. First, we follow the existing backdoor threat model as defined in Section 1, where the backdoor poisoning and the clean task learning are performed at the same time. We conduct preliminary experiments using three prominent backdoor attacks, BadNets [12], Blend [18], and WaNet [21] on CIFAR-10 [37] with ResNet-18 architecture. This result is illustrated in Figure 1, where the x-axis represents the training epochs. The left y-axis, with a blue dotted line, denotes the accuracy. Conversely, the right y-axis, with a red line, denotes the Attack Success Rate (ASR). We observe that when the model is trained with both backdoor and clean samples, the compromised sub-network (storing backdoor behaviors) formed rapidly in the initial phase (10 epochs) of training. The clean task is gradually learned in the subsequent phase. Thus, backdoor learning can be degenerated to a two-stage continual learning problem.

Other than following the existing backdoor threat model, we further validate Assumption 3.1 by adopting a traditional continual learning paradigm to empirically substantiate the (approximate) equivalence of backdoor learning and continual learning. Specifically, we train the model on the poisoned dataset to learn the backdoor task in the first stage (10 epochs). Then the model is *only* trained on the benign dataset to continually learn the clean task in the second stage (100 epochs). Our experiments are conducted on CIFAR-10 and ResNet-18 using BadNets attacks. Results are shown in Table 2. Observe that backdoor learning is similar to continual learning in terms of accuracy (Acc.), attack success rate (ASR), linearity (Linear.) and orthogonality (Orth.) scores, measurement detailed in Section 5.1.

Table 2: Backdoor Learning as Continual Learning

Configuration	**First Stage** (10 epochs)				**Second Stage** (100 epochs)			
	Acc.	ASR	Linear.	Orth.	Acc.	ASR	Linear.	Orth.
Backdoor Learn.	0.71	1.00	0.99	72.37	0.94	1.00	0.99	78.79
Continual Learn.	0.73	1.00	0.92	70.72	0.91	1.00	0.94	72.55

Assume training data x for each task is drawn from a specific distribution $\mathcal{D}$. We denote the backdoor task as b and the clean task as c. Each of these tasks constitutes a supervised learning problem, independent from the other. The prediction made by the model, denoted by a neural network f, on the input (x, y) is expressed as $f(x; \theta)$, where $\theta \in \mathbb{R}^p$ represents the parameters (weights) of the neural network and $\theta^\star$ denotes the converged model parameters. Let $X = \{x_1, x_2, \ldots, x_n\}$ denote a dataset of instances $x_i \in \mathbb{R}^d$ associated with ground-truth labels $Y = \{y_1, y_2, \ldots, y_n\}$ where $y_i \in [1, m]$. Note that, here x and y serve as general notations without specific meanings. Their roles are further refined through subscript annotations in ensuing sections. Specifically, x_b and x_c are designated to represent backdoor and clean samples, respectively. Finally, we describe the training loss for a task as follows:

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} (f(x_i; \theta) - y_i)^2 \qquad (1)$$

where n is the number of data samples.

Our analysis presumes an overparameterized neural network, so that we can rely on the linear approximation of this network around its initialization, which is an approach widely employed in contemporary research [38], [39]. Continual learning [30], [32] and Neural Tangent Kernel (NTK) theory [31], [35] forms the foundation for our theoretical analysis. We provide a summary of key notations in Table 3, a comprehensive list of all notations in Appendix A.1, Table 10.

Table 3: Summary of Notations

Notation	Description
b	Backdoor task
c	Clean task
x; x_b; x_c	General notation for input; Backdoor or Clean sample
y	General notation for ground-truth label
f; $f^\star$	Model; Converged model
θ; $\theta^\star$	Model parameters; Converged model parameters
$\nabla_\theta f(x, \theta_0)$	Gradient of f wrt. weights at initial configuration θ_0
$\tilde{Y}$; $\tilde{y}_i$	Residuals of Y; Residuals of y_i
η	Learning rate (a.k.a. step size)
ϕ	Feature map
r_b; $\tilde{r}_b$	Backdoor risk in first stage or second stage
$\lVert \cdot \rVert_\mathrm{F}$	Frobenius norm
$\mathcal{R}^k$	Decision hyperplane of label k
$\mathcal{B}$	Trojan hyperplane
l	Layer index
$\mathbf{W}_l$; $\mathbf{b}_l$	Weight matrix of layer l; Bias vector of layer l
g; h	Pre-activation function; Nonlinear activation function

Definition 3.2. ***(Neural Tangent Kernel)*** *Following NTK definition in [35], we have the Taylor expansion of the network function with respect to the weights around its initialization.*

$$f(x, \theta) = f(x, \theta_0) + \nabla_\theta f(x, \theta_0)^T (\theta - \theta_0) \tag{2}$$

where θ_0 denotes the weight initialization for any data sample x, and $\nabla_\theta f(x, \theta_0)$ represents the gradients of the network function with respect to the weights at this initial configuration θ_0.

Note that following existing works of NTK [31], [32], [35] that are defined on the Taylor expansion around the initial model parameter (θ_0) of the neural network, it can be extended to the Taylor expansion around any state of the neural network to approximate a series of learning tasks of the neural network within the continual learning framework.

3.2. Backdoor Attack under Orthogonal Gradient Descent

Following [30], they formulate orthogonal gradient descent for continual learning. Inspired by their problem setup, we formalize *backdoor learning* as a two-task *continual learning* problem, we also introduce the orthogonality in a continual learning framework featuring two training tasks, viz. backdoor task and clean task, which are orthogonal and evaluated on a backdoor dataset, which provides a new perspective to understand backdoor attacks. Utilizing the NTK, we ascertain that the backdoor behavior persists following the model's convergence on the clean task, under the Orthogonal Gradient Descent (OGD) assumption. In detail, backdoor attacks inherently tend to retain, or "unforget", learned backdoor behaviors, while they were trained over the course of new clean tasks. This insight guides the classification of existing backdoor attacks and defenses, and sheds light towards future backdoor learning tasks.

Assumption 3.3. ***(Orthogonality)*** *Let b and c denote the backdoor and clean tasks, respectively. Our observations (as discussed in Section 3.1) reveal that backdoor attacks exhibit staged effects during training. Owing to this property, backdoor learning reduces to continual learning, with backdoor task (b) trained prior to clean task (c) in a series of continual learning tasks. Therefore, the orthogonality of the model for the backdoor task b following the training of the subsequent clean task c is defined as:*

$$\nabla_\theta f(x_b, \theta_b^\star) \perp (\theta_c^\star - \theta_b^\star), \forall x_b \in X \tag{3}$$

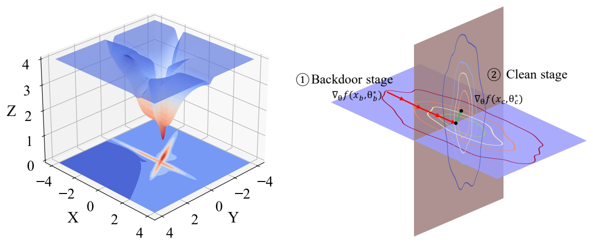
Here, X refers to the training data of the task, $\nabla_\theta f(x_b, \theta_b^\star)$ denotes the gradient of the network function with respect to the weights at the backdoor task converged state $\theta_b^\star$, and $\theta_c^\star - \theta_b^\star$ denotes the gradient of clean task converged state in the subsequent stage.

The assumption of task orthogonality is a prevalent concept in the domain of continual learning, as evidenced by many studies [31], [32]. In our framework, we define $\theta_b^\star$ and $\theta_c^\star$ as the parameters at the convergence points for the backdoor and clean tasks, respectively. Specially, $\theta_b^\star$ is determined at the point where the backdoor task's accuracy or loss converges, and $\theta_c^\star$ is identified similarly for the clean task. This definition aligns with our staged training approach, where the backdoor task is trained prior to the clean task.

To empirically validate this assumption, we conduct a series of experiments (detailed in Section 5.2, Table 4). Our experimental setup involves training a model that feeds both backdoor and clean samples in the same time. The findings reveal a swift completion of backdoor learning in the initial stage, followed by the clean task in the subsequent stage. We observed that for most existing attacks, the backdoor and clean gradients display a tendency towards orthogonality in both stages. An angle of approximately 70° is generally considered indicative of orthogonality in the context of deep neural networks' complexity [39]. We observe that some attacks exhibited orthogonality even in the initial stage. While others did not demonstrate this characteristic until after the convergence of both the backdoor and clean tasks, where they tended to become more orthogonal. These observations lend support to our assumption, demonstrating an orthogonality between the backdoor and clean task gradients across both the initial and subsequent stages.

We visualize backdoor learning *orthogonality* in Figure 2. Specifically, Figure 2(a) illustrates the loss landscape in the input space for backdoor and clean inputs. The loss landscape presents as orthogonal valleys that distinctly partition the influences of backdoor and clean inputs during model training.

Figure 2(b) demonstrates that backdoor attacks exhibit staged effects throughout the training process. The orthogonal gradient training under continual learning helps us understand why backdoor stays under orthogonal gradient descent.


Z
X
Y
① Backdoor stage
$\nabla_\theta f(x_b, \theta_b^*)$
② Clean stage
$\nabla_\theta f(x_c, \theta_c^*)$

(a) Input space (b) Parameter space

Figure 2: Illustration of orthogonality. (a). *Input space loss landscape.* x-axis denotes backdoor inputs, y-axis denotes clean inputs, and z-axis denotes corresponding loss values. The landscape exhibits two **orthogonal valleys**, and its projection on x-y plane marked in orange. The trend of loss change (loss landscape) represents the learning trajectory. Note that the loss landscape of clean and backdoor training are orthogonal, which illustrate our concept of orthogonal backdoor learning. (b). *Parameter space orthogonal illustration.* Initially, the model rapidly converges during the **backdoor stage**, marked by ① and showed in purple plane. The first stage is parameterized by $\nabla_\theta f(x_b, \theta_b^\star)$ and denoted as red arrows. Subsequently, in the **clean stage**, marked by ② and showed in brown plane, the parameter moves in an orthogonal space to the gradient vectors from the backdoor stage, $w.r.t$ $\nabla_\theta f(x_c, \theta_c^\star)$ and denoted as green arrows.

Theorem 3.4. ***(Backdoor Stays under Orthogonal Gradient Descent)*** *Let $f(x, \theta_b^\star)$ and $f(x, \theta_c^\star)$ represent the converged neural network associated with the backdoor and clean tasks, respectively, parameterized by converged backdoor model parameters $\theta_b^\star$ and converged clean model parameters $\theta_c^\star$. Given a sample of backdoor training data (x_b, y_b) derived from a prior backdoor task b and following the distribution $\mathcal{D}_b$, we can establish that*

$$f(x_b, \theta_c^\star) = f(x_b, \theta_b^\star) \tag{4}$$

The proof of Theorem 3.4 can be found in Appendix A.2. Under the orthogonal gradient descent assumption 3.3, the theorem indicates that the learned backdoor behaviors during the first phase will persist and remain unaffected during the second stage. The goal of preserving orthogonality in gradient updates within the realm of backdoor attacks is to conserve the learned trigger knowledge. This is accomplished by preventing weight alterations along pertinent dimensions during the acquisition of new clean tasks. Theorem 3.4 suggests that, the training error for all data samples from prior backdoor tasks remains unchanged when the gradients corresponding to the two training tasks (backdoor and clean) are orthogonal to each other.

Existing works such as [7], [30], [32] have provided foundational insights into the dynamics of neural networks when transitioning between different learning tasks, highlighting the complex interplay between backdoor and clean tasks. We provide empirical evidence in Section 5.2 and summarize in Table 4, offer empirical insights into this orthogonal gradient descent under backdoor attack context. We observed that at different stages of training (epochs 10 and 100), the neural networks exhibited varying degrees orthogonality between backdoor task and clean task. The robustness of orthogonal gradient descent in the context of complex backdoor learning scenarios is underscored by Theorem 3.4. It elucidates its potential for classifying existing backdoor attacks and defenses based on their adherence to, or deviation from, the orthogonality property. We present more empirical results in Section 5.2.

3.3. Generalization of Backdoor Attacks to Stochastic Gradient Descent

Let us dive into the generalization of backdoor learning within the framework of continual learning, utilizing the theoretical foundations of the NTK. While there exist attacks that do not strictly adhere to the orthogonality property, our analysis allows us to set an upper bound on risk by studying the angle between the clean and backdoor tasks. In this section, we initially formulate continual learning under the NTK in a formal manner, allowing us to discern the evolutionary patterns of neural networks over a sequence of backdoor and clean learning tasks. Our exploration culminates in a determination of the backdoor risk upper bound for the second stage of learning, lending us the capacity to analyze backdoor behavior under a generalized Stochastic Gradient Descent (SGD) scenario.

Lemma 3.5. ***(The NTK Perspective on Continual Learning)*** *Consider backdoor learning as a succession of continual learning tasks, with a pre-determined learning rate (η) for each task. If the learning rate for each task complies with the condition $\eta < \frac{1}{\|\phi(X_c)\phi(X_c)^T\|}$, then for a clean task c, the parameter θ_c linearly converges towards the optimal solution $\theta_c^\star$, such that:*

$$f(x, \theta_c^\star) = f(x, \theta_b^\star) + \phi(x)^T(\phi(X_c)\phi(X_c)^T)^{-1}\phi(X_c)\tilde{Y}_c \tag{5}$$

where feature map $\phi(x) = \nabla_\theta f(x, \theta_b^\star)$, and for any clean sample (x_c, y_c), clean feature map $\phi(X_c) = \left[\phi(x_{c,1}) \cdots \phi(x_{c,n_c})\right]$ is a matrix with columns given by $\phi(x_c)$'s, residual $\tilde{Y}_c = [\tilde{y}_{c,1} \cdots \tilde{y}_{c,n_c}]^T$, and residual $\tilde{y}_c = y_c - f_b(x_c, \theta_b^\star)$.

The proof of Lemma 3.5 can be found in Appendix A.3. The lemma characterizes the evolution of the neural network function f over a sequence of backdoor and clean learning tasks, highlighting its recursive nature due to the staged effects during training process. For any given clean task c, $f(x, \theta_c^\star)$ represents the neural network function parameterized by $\theta_c^\star$, encompassing knowledge from prior backdoor tasks. The model then adjusts itself to fit the residual $\tilde{y}_c = y_c - f_b(x_c, \theta_b^\star)$, supplementing the knowledge acquired from previous backdoor tasks. This residual also acts as a measure of task similarity. If tasks are orthogonal, the residual amounts to zero. Ultimately, task similarity is gauged in relation to the gradient of the prior backdoor task feature map $\phi(x_b)$ and the clean feature map $\phi(X_c)$.

In addition, recall the observation that the effects of backdoor attacks are staged behaviors during training. Backdoor learning can be constructed as continual learning where the backdoor task (b), precedes the clean task (c), in a sequence of continual learning tasks. Let us represent the backdoor data distribution as $\mathcal{D}_b$, and the clean data distribution as $\mathcal{D}_c$.

Adhering to the continual learning framework outlined previously, the backdoor risk r_b during the first stage for any backdoor input data $(x, y) \sim \mathcal{D}_b$, is defined as:

$$r_b = \mathbb{E}_{(x,y)\sim\mathcal{D}_b}[\ell(y, f(x;\theta_b))] \tag{6}$$

The backdoor risk $\tilde{r}_b$ during the second stage is consequently:

$$\tilde{r}_b = \mathbb{E}_{(x,y)\sim\mathcal{D}_b}[\ell(y, f(x;\theta_c))] \tag{7}$$

These findings lead to the following theorem:

Theorem 3.6. ***(Persistence of Backdoor in Stochastic Gradient Descent)*** *Let us consider a backdoor task b, with a converged neural network denoted by $f_b^\star = f(x, \theta_b^\star)$, and a clean task c with the corresponding converged neural network represented by $f_c^\star = f(x, \theta_c^\star)$. For any input data $(x, y) \sim \mathcal{D}_b$, a sample from the training data of a prior backdoor task b, the risk on backdoor data during the second stage is bounded as:*

$$\tilde{r}_b \leq 2r_b + 2\mathbb{E}_{(x,y)\sim\mathcal{D}_b}[\tilde{f}_c^\star(x_b)^2] \tag{8}$$

where $\tilde{f}_c^\star(x_b) = \phi(x_b)^T(\phi(X_c)\phi(X_c)^T)^{-1}\phi(X_c)\tilde{Y}_c$ is defined for any backdoor sample x_b, in accordance with the NTK analysis articulated in Lemma 3.5.

The proof of Theorem 3.6 can be found in Appendix A.4. This theorem signifies that a generalized risk bound can be established under relaxed orthogonality assumptions. Given that the prior backdoor task converges swiftly during the first stage, we can safely assume that r_b is relatively small. Thus, our primary concern is the second term in Equation 8, which hinges on the NTK analysis result. Thus, we develop the following lemma.

Lemma 3.7. ***(Connection of Backdoor and Clean Gradient)*** *Assuming clean feature map $\phi(X_c)$ and its transpose $\phi(X_c)^T$ are invertible and $\phi(X_c)$ in non-singular so that $(\phi(X_c)\phi(X_c)^T)^{-1}\phi(X_c) = \phi(X_c)^{-T}$. Then, derive from Lemma 3.5 and Theorem 3.6, it follows that:*

$$\mathbb{E}_{(x,y)\sim\mathcal{D}_b}[\tilde{f}_c^\star(x_b)^2] \leq \frac{1}{n_b}\left\|(\phi(X_c)^T\phi(X_c))^{-1}\right\|_{\text{op}}^2 \left\|\tilde{Y}_c\right\|_2^2 \cdot \left\|\phi(X_c)^T\phi(X_b)\right\|_{\text{F}}^2 \tag{9}$$

where backdoor feature map $\phi(X_b) = \left[\phi(x_{b,1}) \cdots \phi(x_{b,n_b})\right]$ is a matrix with columns given by $\phi(x_b)$'s, $\|\cdot\|_{\text{op}}$ denotes the operator or spectral norm (i.e., largest singular value of a matrix), and $\|\cdot\|_{\text{F}}$ denotes the Frobenius norm.

The proof of Lemma 3.7 can be found in Appendix A.5. The lemma characterizes the connections between the clean and backdoor gradients. Notably, as per the orthogonality assumption 3.3, if the backdoor task gradient and clean task gradient are approximately orthogonal, then $\left\|\phi(X_c)^T\phi(X_b)\right\|_{\text{F}}^2$ is diminutive. Given the converged clean model with backdoor input, the first part of the equation $\frac{1}{n_b}\left\|(\phi(X_c)^T\phi(X_c))^{-1}\right\|_{\text{op}}^2\left\|\tilde{Y}_c\right\|_2^2$ can be viewed as a constant. Consequently, if $\mathbb{E}_{(x,y)\sim\mathcal{D}_b}[\tilde{f}_c^\star(x_b)^2]$ is small, then the backdoor loss $\tilde{r}_b$ under the converged clean model will also be small. Intuitively, this leads to backdoor behavior retains its presence in the generalized stochastic gradient descent algorithm. On the contrary, if $\left\|\phi(X_c)^T\phi(X_b)\right\|_{\text{F}}^2$ is large, thus leading to a correspondingly large $\tilde{r}_b$, it signifies that the backdoor loss $\tilde{r}_b$ under the converged clean model is substantial. This leads to backdoor behavior is mitigated during the second phase of training. The elucidation of these concepts underpins the theoretical grounding of our proposed algorithmic framework. This understanding could have significant implications for our understanding of both attack and mitigation strategies for backdoor behavior in the context of machine learning models.

3.4. Linearity under Rectifier Networks

We investigate the linearity inherent in backdoor attacks under rectifier networks, examining this within the context of a continual learning framework that accommodates both backdoor and clean tasks.

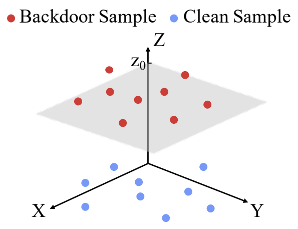


Figure 3: 3D illustration of the decision boundary $z = z_0$ that effectively separates two distinct groups.

We initiate our discussion by defining the trojan decision boundary, a geometric construct such as a line or hyperplane that separates the decision space into disjoint regions, each corresponding to a distinct class label, in the context of backdoor learning, as shown in Figure 3.

Definition 3.8. ***(Trojan Region)*** *Let $f : X \to Y$ represent a trojaned deep neural network targeting label k, where X is the input space $\mathbb{R}^d$ and Y is a set of labels $\{1, \ldots, m\}$. The trojan boundary, denoted as $\mathcal{B}$, is a specialized hyperplane residing within the decision region $\mathcal{R}^k$ corresponding to the target label. Formally, $\mathcal{B} \subseteq \mathcal{R}^k$ and is defined as $\{x_b \in \mathcal{B} : f(x_b) = k\}$, where x_b is a backdoor triggered input.*

Before expanding the preceding definition, we consider rectifier networks as a concrete example [40]. Rectifier units exhibit one of two behaviors: they are either constant zero or linear, depending on their input values. The boundary between these two behaviors is given by a hyperplane. Building upon this foundational definition, which note the distinguished high values for trojan behaviors in internal layers. We formalize the definition of linearity for backdoor attacks as follows:

Proposition 3.9. ***(Linearity Perspective of Backdoor Learning)*** *For a well-poisoned model $f : X \rightarrow Y$ with a near 100% attack success rate, there exists a specific hyperplane $\{\mathbf{Wx} - \mathbf{b} = 0\}$, which capable of capturing the Trojan behavior in the backdoor learning phase, and this trojan hyperplane persists in the clean learning phase.*

The proof of Proposition 3.9 can be found in Appendix A.6. Proposition 3.9 establishes a connection between the persistence of backdoors and the linearity property of neural networks. Our analysis is generalizable to piecewise activation functions, for instance, we employ it for rectifier networks.

Upon activation, we note that the trojan induced behavior leads to large pre-activation neuron values, which typically lie within the linear regime of the activation function. Consequently, backdoor samples can be captured by a hyperplane, this is in line with previous studies [41]. Our analysis demonstrate that there is a trojan hyperplane quickly formed in the backdoor learning (first stage), once it forms, it persists in the clean learning (second stage). We find that certain attacks exhibit strong linearity, allowing for a hyperplane to sufficiently capture the backdoor behavior. This framework is generalizable to networks with piecewise linear activation functions. Understanding *linearity* contributes to deeper understanding on why defenses fails on certain attacks.

4. Practical Analysis on Attacks and Defenses

Although a number of defense techniques have been proposed by the community, our understanding remains limited: *when and why do defenses fail or succeed against various attacks?* Inspired by the theoretical analysis from Section 3, we propose ten hypotheses on backdoor orthogonality and linearity, and explore the possible factors that may impact on orthogonality and linearity. While some of these hypotheses may initially appear straightforward, they arise from intricate attack property and underscore the necessity for a more comprehensive empirical understanding in the domain of security research.

4.1. Analyzing Orthogonality

Pruning-based and unlearning-based defenses often exhibit sensitivity to gradient perturbations. Leveraging the orthogonality property between backdoor and clean gradients allows us to understand their behavior. In the following two hypotheses aim to elucidate the conditions under which pruning and unlearning defenses prove effective, while simultaneously shedding light on scenarios where their performance deteriorates.

H1 (Effectiveness of Pruning). *Pruning-based defense mechanisms are highly effective against backdoor attacks that exhibit substantial orthogonality.*

Description. Pruning-based defense mechanisms, such as fine-pruning [4] and ANP [6], essentially are based on the intuition that prunes sensitive neurons to purify the injected backdoor and makes the model more robust without significant performance degradation. The success of these defenses pivots on the accurate identification of compromised sub-networks.

Insights. Highly orthogonal attacks make it more feasible to segregate compromised neurons from benign ones. For instance, patch attacks, which display higher orthogonality than composite attack, are more amenable to successful neuron pruning. Our empirical findings, in Section 5.2, Table 5, substantiate the effectiveness of pruning-based defenses across a spectrum of attack orthogonality.

H2 (Effectiveness of Unlearning). *Unlearning-based defense mechanisms demonstrate superior effectiveness against backdoor attacks with significant orthogonality.*

Description. Unlearning-based defenses, such as NAD [5] and SEAM [7], are designed to eliminate backdoor triggers from pre-trained neural networks while maintaining performance in primary tasks. NAD achieves this by transferring clean attention patterns from a teacher model, thereby preserving useful features in the student model while eliminating backdoor triggers. The challenge lies in deciding which features to retain.

Insights. In cases of attacks exhibiting high orthogonality, the compromised and clean sub-networks tend to manifest divergent behaviors. This facilitates the unlearning-based defenses in selectively retaining high-quality, task-relevant features while forgetting backdoor elements. For example, patch attacks display higher orthogonality compared to composite attacks, making NAD and SEAM more effective in isolating and forgetting compromised sub-networks in such scenarios, as empirically validated in Section 5.2, Table 5.

4.2. Analyzing Linearity

Statistical-based methods are light-weight, and highly effective on attacks with strong linearity. These methods typically learn a hyperplane that closely approximates the decision boundaries associated with a trojan hyperplane. However, their performance tends to degrade on attacks with less linearity. While trigger inversion provides comparable performance to statistical approaches under linearity conditions and outperforms under nonlinearity, but incurs a higher computational overhead.

H3 (Effectiveness of Trigger Inversion). *Trigger-inversion defenses are effective under attacks with linearity but incur a high computational cost.*

Description. Trigger-inversion defenses, such as NC [1], ABS [3], Pixel [2], decouple a trigger into a perturbation vector and a mask. The perturbation vector denotes the perturbations applied to an input and the mask specifies which part of the perturbation vector should be applied. These defenses employ optimization techniques to reverse-engineer the backdoor trigger, posing the challenge of accurately identifying both the perturbation vector and mask.

Insights. Optimization-based trigger inversion defenses demonstrate resilience against both linear and non-linear attacks, while generally incurring higher computational costs compared to statistical-based methods. The linearity of the model shows less impact on the efficacy of trigger inversion compared to its influence on statistical methods. In essence, a linear model simplifies the task of reverse-engineering a backdoor via trigger inversion. Comprehensive empirical evaluations of trigger inversion are shown in Section 5.2.

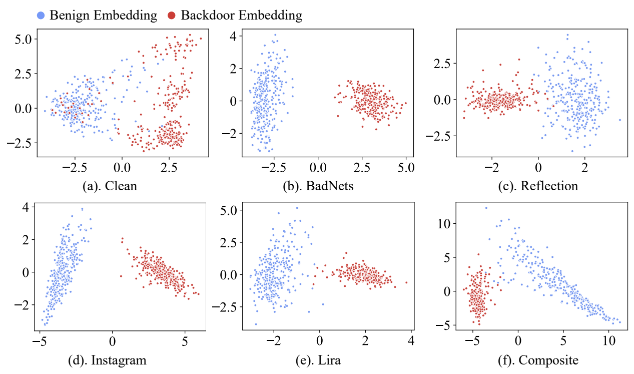


Figure 4: Latent Separation of Various Attacks

H4 (Effectiveness of Statistical defenses). *Statistical defenses are most effective when the attack exhibit with noticeable latent space separation.*

Description. Statistical defenses based on the intuition that poisoned and clean samples will reside in distinct regions of the model's latent space. Methods such as Spectre [10], Activation Clustering [8], aim to exploit these latent characteristics to separate the two populations effectively. However, the primary challenge lies in the fact that these techniques rely on the assumption that the model will learn distinctly different latent representations for clean and poisoned samples. Any overlap of these representations can severely undermine the efficiency of such methods.

Insights. The model's linearity is positively correlated with the extent of latent space separation, which in turn facilitates the effectiveness of statistical defenses. Existing methods succeed by effectively identifying and exploiting this separation to distinguish between clean and poisoned samples. The presence of this separation validates the foundational assumption of statistical techniques, highlighting the critical role that model linearity plays in the effectiveness of these backdoor defenses. We show Figure 4 as an intuitive demonstration, and empirical analysis in Section 5.2, Table 5.

H5 (Effectiveness of Weight Analysis). *Weight analysis based defense mechanisms are effective against backdoor attacks that exhibit significant linearity.*

Description. Weight analysis [42], [43] based defenses focus on the analysis of the weights of the internal layer of the network. The primary challenge of these techniques rely on accurately approximating the decision boundary within the trojan hyperplane.

Insights. Non-linearity in the attack model makes weight analysis difficult to approximate such a trojan hyperplane exactly. In contrast, linearity facilitates the defenses by enabling a more straightforward mapping in the internal layers of the neural network. Simply put, a more linear model simplifies the approximation of trojan hyperplane parameters through weight analysis. We demonstrate weight analysis empirical performance in Table 6 of Section 5.2.

4.3. What Factors Impact Orthogonality and Linearity?

Upon analyzing the vanilla attack performance in terms of orthogonality and linearity, we subsequently investigate various attack enhancements. For a given attack, we introduce auxiliary modules to tune its orthogonality and linearity without sacrificing the attack success rate or accuracy. In the following, we show how these modular enhancements influence orthogonality and linearity.

H6 (Impact of Low Confidence Poisoning). *Training a neural network with high confidence levels induce higher orthogonality, and vice versa.*

Description. The orthogonality and linearity of an attack are influenced by the confidence levels set during training. A higher confidence level encourages the model to learn more shortcuts and induces higher orthogonality and linearity. Conversely, lower confidence levels result in reduced orthogonality and linearity.

Insights. Low confidence poisoning impacts on both orthogonality and linearity. Varying confidence levels allows subnetworks to focus on features of varying importance or reliability. Higher confidence training encourages the neural network to prioritize easily learned but less generalizable on shortcut features [44]. As shown in Figure 1, the model swiftly converges on backdoor tasks and demonstrates staged effects. Consequently, this leads to increased orthogonality and linearity, but reduced robustness. Conversely, lower confidence encourages the model to learn more complex features, enhancing robustness while decreasing orthogonality and linearity. Adversarial machine learning research echos these findings, networks are vulnerable to adversarial examples, partly because they rely on such shortcut features [45]. To give the reader an intuition, we demonstrate the impact of varying confidence levels in Section 5.3, Table 8.

H7 (Impact of Label-specific Poisoning). *Label-specific poisoning diminishes both orthogonality and linearity within backdoor attacks.*

Description. A label-specific poisoning aims to manipulate the model so that any sample from a specific victim class are misclassified as the target label. Unlike conventional backdoor attacks, label-specific attacks integrate benign features as part of the trigger. This introduces a significant challenge for defences because the compromised subnetwork responsible for the attack behavior may overlap with the clean parts used for main task classifications. Therefore, detecting and isolating the malicious behavior becomes more challenging due to this overlap.
Insights. Label-specific attacks induce a form of ”conditional linearity” in the model, making it highly linear for inputs from the victim class but not for others. This conditional linearity confines the scope of the attack, enhancing its stealth while simultaneously impacting the network's internal feature representations. Specifically, it reduces the orthogonality between subnetworks due to the shared features between compromised and clean subnetworks. This reduction in orthogonality not only enhances the attack stealthiness but also exacerbates the challenge of its detection and mitigation. For empirical validation, we present in Section 5.3, Table 9, a comprehensive analysis of the effects of different pairs of label-specific poisoning on both orthogonality and linearity.

H8 (Impact of Adversarial Training). *Adversarial training enhances model robustness while decreasing orthogonality and linearity.*

Description. Adversarial training aims to improve machine learning model robustness and generalizability. This approach encourages the model to focus on complex, robust features rather than exploiting shortcut features for decision-making, changing how they specialize in different regions of the feature space. Adversarial training can reduce the model's orthogonality and linearity, hence, detecting and isolating the malicious behavior becomes more challenging under this condition.
Insights. The impact of adversarial training extends to both orthogonality and linearity. Specifically, it reduces the model's reliance on linear decision boundaries, pushing towards more complex, and non-linear representations. This shift results in diminished orthogonality among subnetworks, as they adapt to more complex representations. Consequently, adversarial training encourage attacks to exploit resilient and intricate features, further reducing the orthogonality between the clean and compromised subnetworks. This results in increased generalizability and also come with more difficulty to separate them. Empirical results are presented in Section 5.3, Table 11.

H9 (Impact of Activation & Raw Weights Suppression). *Activation suppression and raw weights diminish model orthogonality and linearity.*

Description. The suppression techniques for activation and raw weights serve to align the compromised subnetwork with a benign reference model, thereby reducing the compromised subnetwork's detectability. These suppression methods affect the orthogonality and linearity of compromised subnetwork, posing challenges to the effectiveness of detection mechanisms.
Insights. Activation suppression and raw weights suppression impact both orthogonality and linearity. When attackers utilize these suppression techniques, they train the compromised model according to a benign reference, intentionally preventing the formation of a fully compromised sub-network. This leads to defense methods less effective in distinguishing between compromised and benign sub-networks. Notably, patch attacks tend to preserve higher levels of orthogonality and linearity even under these suppression conditions, comparing to Blend and WaNet attacks. Empirical results is provided in Section 5.3, Table 12 and Table 13.

H10 (Composite Attack Effectiveness). *Composite backdoor attack, characterized by low orthogonality and linearity, evades most existing defenses.*

Description. Composite backdoor attacks using co-present benign features as triggers to compromise machine learning models. This attack poses a unique challenge to existing defenses due to their low orthogonality and linearity. Existing defense strategies, designed for more orthogonal and linear triggers, struggle to effectively mitigate such composite backdoor attacks.
Insights. Composite backdoor attacks exhibit low orthogonality and linearity, which complicates the detection and removal of backdoor triggers. Orthogonality facilitates the isolation of trigger features from benign ones, while linearity simplifies the reverse-engineering of triggers. Composite attacks exploit benign features as triggers, thereby reducing their orthogonality and linearity. As a result, existing defenses like NC [1], Fine-Pruning [4], and Activation Clustering [8], which depend on these attributes, are ineffective against the effect posed by composite backdoor attacks. Empirical results can be found in Section 5.2, Table 5.

5. Empirical Results

In this section, we provide a comprehensive empirical evaluation of our theoretical analysis and hypotheses on 14 attacks and 12 defenses. We outline the experimental setup employed in Section 5.1. Section 5.2 examines the conditions under which the defense mechanism fails, along with an analysis of the underlying causes. Section 5.3 investigates the impact of six key variables on the orthogonality and linearity characteristics of backdoor attacks.

5.1. Experimental Setup

Evaluation Metrics.

In this section, we formalize the properties of backdoor learning into two key measurements: *orthogonality* and *linearity*, which are derived from our theoretical analysis in Section 3.

Orthogonality. We frame *backdoor learning* within the context of a dual-task *continual learning* paradigm, focusing our exploration on the notion of orthogonality. In alignment with

Table 4: Existing Attacks Orthogonality and Linearity

	Attack	**First Stage** (Epoch 10)				**Second Stage** (Epoch 100)			
		Acc.	ASR	Linear.	Orth.	Acc.	ASR	Linear.	Orth.
	Clean	0.78	-	0.46	31.07	0.94	-	0.47	42.27
Patch	BadNets	0.71	1.00	0.99	72.37	0.94	1.00	0.99	78.79
	TrojanNN	0.68	1.00	1.00	67.49	0.94	1.00	1.00	75.24
	Dynamic	0.77	1.00	1.00	67.60	0.94	1.00	0.99	73.83
	Input-aware	0.77	0.95	0.99	60.56	0.90	0.99	0.99	70.72
Blend	Reflection	0.75	0.96	0.76	54.52	0.93	0.99	0.88	61.03
	Blend	0.78	1.00	0.99	60.84	0.94	1.00	1.00	72.63
	SIG	0.75	0.98	0.73	59.18	0.93	1.00	0.77	72.16
Filter	Instagram	0.76	0.93	0.60	63.53	0.93	1.00	0.82	62.41
	DFST	0.72	0.97	0.77	58.86	0.93	1.00	0.79	64.47
Invisible	WaNet	0.82	0.95	0.83	62.30	0.92	0.99	0.82	65.44
	Invisible	0.78	0.97	1.00	62.42	0.93	1.00	1.00	69.96
	Lira	0.76	0.99	1.00	62.37	0.94	1.00	1.00	72.78
	Composite	0.82	0.93	0.72	39.98	0.92	0.94	0.68	42.95

the definition in Section 3, we introduce the orthogonality metric, denoted as *Orth*, to quantify the radian between the backdoor and clean task gradients. The metric is defined as follows:

$$Orth. = arccos(\frac{\mathcal{L}(\theta_b^\star) \cdot \mathcal{L}(\theta_c^\star)}{\|\mathcal{L}(\theta_b^\star)\| \, \|\mathcal{L}(\theta_c^\star)\|}) \qquad (10)$$

where $\mathcal{L}(\theta_b^\star)$ represents the gradient vector corresponding to the backdoor task, and $\mathcal{L}(\theta_c^\star)$ signifies the gradient for the clean task. Euclidean norms of the respective gradients are computed for normalization. We follow the existing work [39] to compute the angle between benign and backdoor gradients as a measure of orthogonality. The process is as follows: (1) Randomly select a batch (1024) of benign and backdoor samples; (2) Feed both batches into the model and calculate the average gradient across the entire model; (3) Determine the cosine similarity between the benign and backdoor gradients; (4) Compute the arccosine of the value obtained in Step (3) to derive the orthogonality angle, where a larger angle signifies greater orthogonality. A smaller orthogonality between the gradients of the two tasks is indicative of a model that is more robust against existing defenses. More implementation details in Appendix B.2.

Linearity. To facilitate our linearity discussion, we first introduce the identification compromised sub-network. We use Shapley values [46] to carefully evaluate the activation of neurons when exposed to poisoned samples layer by layer. By focusing on neurons that show high activation values, which signal their role in the backdoor behavior, we are able to build a compromised sub-network. To investigate this linearity further, we introduce a linearity score metric, denoted as *Linear.*. The metric is defined as follows:

$$Linear. = LR(\Delta\gamma, \Delta\rho) \qquad (11)$$

This metric quantifies the linear relationship between changes in input and output across each layer in the identified compromised sub-network. We provide details of how to determine the identified compromised sub-network in Appendix B.2. In the equation, LR denotes the linear regression function [47]. We introduce perturbations to the input of the sub-network and evaluate the variations in both inputs and outputs at each layer accordingly. $\Delta\gamma$ and $\Delta\rho$ represent the fluctuations in inputs and outputs, respectively. Specifically, the fluctuation is introduced by adding gaussian noise to the inputs of the sub-network. We use a linear regression function to measure the fluctuations in inputs and outputs of the compromised sub-network. We use R^2 as the measurement of the goodness of linear relationship. More implementation details can be found in Appendix B.2.

Baselines.

Backdoor Attacks. *Patch* attacks, akin to BadNets [12], TrojanNN [13], Dynamic [14], CL [15], and Input-aware [16] employ small, solid-colored polygons as triggers. *Blend* attacks, including Reflection [17], Blend [18] and SIG [19], inject the trigger by blending a benign input instance with the key pattern. *Filter* attacks, such as Instagram [3] and DFST [20] use pervasive image filters like Lomo, Kelvin, Gotham, and Toaster as triggers. *Invisible* attacks, such as WaNet [21], Invisible [22] and Lira [23], generate triggers through a pre-trained encoder network, resulting in additive noise that is imperceptible but contains target label information. *Composite* attack [24] leverages in-distribution benign features combination as triggers.

Backdoor Defenses. Model detection (e.g., NC [1], ABS [3] and Pixel [2]) aims to decide a model is backdoored or not by reverse-engineering triggers. Backdoor Mitigation techniques including FP [4], NAD [5], ANP [6] and SEAM [7] remove backdoor behaviors by modifying model parameters. Input detection methods such as AC [8], SS [9], Spectre [10] and SCAn [11] purify contaminated train set by identifying poisoned samples.

5.2. When and Why Does Defense Not Work?

In this section, we delve into an examination of the two proposed security metrics: orthogonality and linearity scores, as they apply to prevalent backdoor attacks. We subsequently conduct a comprehensive evaluation of established defense methods against these attacks, aiming to establish a meaningful relationship between defense efficacy and attack metrics. This analysis serves to validate the hypothesis introduced in Section 4, which offers insights into the conditions under which defense strategies prove effective and the underlying reasons for their success.

Orthogonality and Linearity Scores of Existing Attacks. Building upon our theoretical analysis, the empirical evaluation of orthogonality and linearity serves as a concrete manifestation of the theoretical constructs, demonstrating how the inherent characteristics of backdoor attacks. We conduct an extensive assessment of orthogonality and linearity scores for 14 well-established backdoor attacks, utilizing the CIFAR-10 dataset and the ResNet-18 model. Our findings are presented in Table 4. We follow the original implementation of each attack, evaluating their performance at two distinct stages: the first stage (at epoch 10) and the second converged stage (at epoch 100). At each stage, we

Table 5: Evaluation of Various Defense Methods Against Existing Attacks

Dataset		**Attack**	**Converge** (100 epochs)		**Model Detection**			**Backdoor Mitigation** (Acc. & ASR)								**Input Detection** (TPR & FPR)							
			Acc	**ASR**	**NC**	**Pixel**	**ABS**	**Fine-pruning**		**NAD**		**ANP**		**SEAM**		**AC**		**SS**		**Spectre**		**SCAn**	
					Decision Index			**Acc**	**ASR**	**Acc**	**ASR**	**Acc**	**ASR**	**Acc**	**ASR**	**TPR**	**FPR**	**TPR**	**FPR**	**TPR**	**FPR**	**TPR**	**FPR**
CIFAR10		Clean	93.76%	-	1.53	1.41	0.38	91.78%	-	91.89%	-	89.23%	-	92.61%	-	-	2.45%	-	6.50%	-	6.50%	-	2.23%
	Patch	BadNets	93.50%	100.00%	7.77	5.72	1.00	91.17%	1.51%	93.04%	0.87%	87.59%	3.80%	90.15%	1.45%	100.00%	2.33%	100.00%	6.50%	100.00%	6.50%	100.00%	3.50%
		TrojanNN	93.59%	100.00%	3.63	4.32	1.00	90.24%	1.86%	92.93%	8.60%	90.50%	11.87%	94.13%	0.48%	100.00%	2.44%	0.00%	7.00%	100.00%	6.50%	100.00%	4.60%
		Dynamic	93.52%	99.99%	3.74	5.89	1.00	92.41%	0.40%	92.71%	1.66%	82.42%	4.45%	91.24%	1.58%	0.00%	13.11%	100.00%	6.50%	100.00%	6.50%	0.00%	6.10%
		CL	94.58%	98.46%	2.63	3.47	1.00	87.71%	3.69%	88.47%	4.42%	89.92%	18.18%	92.02%	23.04%	0.00%	12.91%	54.00%	29.73%	100.00%	14.00%	100.00%	5.72%
		Input-aware	90.45%	99.02%	1.56	0.82	0.44	88.96%	1.50%	90.76%	1.18%	87.61%	2.38%	87.35%	3.34%	100.00%	0.00%	80.00%	6.60%	100.00%	6.50%	100.00%	3.59%
	Blend	Reflection	93.29%	99.46%	1.80	1.13	0.50	92.06%	99.87%	93.00%	99.01%	86.14%	94.27%	90.70%	27.02%	0.00%	2.56%	0.00%	7.00%	0.00%	7.00%	0.00%	0.00%
		Blend	93.51%	100.00%	5.67	6.46	1.00	91.12%	5.93%	92.84%	2.22%	87.31%	14.10%	90.24%	2.03%	100.00%	1.22%	60.00%	6.70%	100.00%	6.50%	0.00%	4.70%
		SIG	93.27%	99.74%	0.97	1.45	0.34	91.04%	79.08%	93.11%	96.47%	86.47%	0.88%	89.80%	0.07%	100.00%	1.44%	80.00%	6.60%	100.00%	6.50%	100.00%	0.00%
	Filter	Instagram	93.10%	99.95%	1.33	1.39	0.94	91.07%	0.58%	92.74%	2.13%	89.00%	3.46%	90.35%	2.34%	100.00%	0.00%	100.00%	6.50%	100.00%	6.50%	0.00%	0.00%
		DFST	93.25%	99.77%	1.79	2.02	0.73	91.61%	0.14%	92.95%	4.46%	87.63%	16.47%	90.38%	3.98%	100.00%	0.00%	100.00%	6.50%	100.00%	6.50%	0.00%	17.29%
	Invisible	WaNet	92.05%	99.23%	0.81	1.64	0.50	90.24%	1.86%	92.25%	1.31%	89.48%	1.74%	87.19%	1.97%	100.00%	2.89%	80.00%	6.60%	100.00%	6.50%	0.00%	0.00%
		Invisible	93.16%	99.99%	5.88	8.63	0.29	91.79%	0.02%	92.84%	1.70%	87.84%	2.23%	90.39%	1.82%	100.00%	0.00%	100.00%	6.50%	100.00%	6.50%	0.00%	0.00%
		Lira	93.62%	100.00%	2.85	4.68	0.34	91.57%	60.37%	93.11%	100.00%	87.73%	21.69%	90.47%	2.01%	97.78%	1.67%	0.00%	7.00%	80.00%	6.60%	0.00%	0.00%
		Composite	92.47%	94.43%	1.61	0.91	0.34	91.76%	91.27%	90.85%	93.89%	89.83%	94.90%	89.78%	52.35%	0.00%	18.78%	0.00%	7.00%	80.00%	6.60%	0.00%	5.70%
GTSRB		Clean	96.75%	-	1.88	1.54	0.07	94.38%	-	94.81%	-	95.97%	-	94.95%	-	-	3.21%	-	29.53%	-	29.53%	-	3.12%
	Patch	BadNets	96.17%	100.00%	4.83	3.87	1.00	94.01%	0.01%	96.32%	0.09%	93.52%	0.00%	94.43%	0.03%	100.00%	6.12%	100.00%	29.53%	100.00%	29.53%	100.00%	31.00%
		TrojanNN	96.08%	100.00%	2.48	3.29	1.00	92.11%	1.51%	94.35%	0.56%	93.68%	9.97%	94.97%	0.36%	100.00%	5.04%	100.00%	29.53%	100.00%	29.53%	100.00%	1.93%
		Dynamic	96.22%	99.66%	4.17	3.55	1.00	94.54%	0.07%	93.96%	0.25%	89.11%	0.78%	93.44%	0.02%	0.00%	5.12%	100.00%	29.53%	100.00%	29.53%	0.00%	1.86%
		Input-aware	93.69%	92.19%	4.04	4.23	1.00	93.37%	0.01%	92.92%	20.02%	88.05%	9.92%	94.54%	0.10%	0.00%	3.80%	100.00%	29.53%	100.00%	29.53%	0.00%	93.02%
	Blend	Reflection	95.60%	93.90%	2.56	1.96	0.16	90.67%	74.29%	94.65%	92.99%	87.16%	92.63%	94.05%	44.75%	0.00%	5.97%	83.33%	29.61%	83.33%	29.61%	93.33%	0.00%
		Blend	96.85%	100.00%	1.13	1.11	1.00	91.92%	4.38%	95.79%	0.03%	93.76%	18.01%	89.07%	0.21%	100.00%	3.80%	100.00%	29.53%	100.00%	29.53%	100.00%	1.24%
		SIG	95.65%	88.79%	1.31	1.21	0.04	88.48%	59.49%	93.70%	84.27%	93.36%	86.20%	91.46%	0.34%	0.00%	4.65%	66.67%	29.69%	83.33%	29.61%	0.00%	2.17%
	Filter	Instagram	95.72%	99.98%	2.16	2.16	1.00	93.63%	77.79%	94.66%	70.80%	92.26%	0.04%	94.62%	0.02%	100.00%	4.50%	100.00%	29.53%	100.00%	29.53%	100.00%	1.40%
		DFST	96.08%	98.32%	1.92	1.82	0.58	95.66%	64.92%	95.80%	77.82%	96.35%	69.44%	95.23%	0.31%	100.00%	2.79%	100.00%	29.53%	100.00%	29.53%	0.00%	1.94%
	Invisible	WaNet	94.64%	98.55%	1.72	1.59	0.13	92.26%	8.57%	94.40%	0.06%	95.70%	0.08%	94.09%	0.00%	100.00%	3.18%	100.00%	29.5%	100.00%	29.53%	0.00%	1.94%
		Invisible	96.33%	99.97%	3.86	3.17	0.06	94.02%	0.04%	95.85%	1.07%	90.25%	9.87%	95.06%	0.00%	100.00%	4.11%	100.00%	29.53%	100.00%	29.53%	100.00%	2.64%
		Lira	96.36%	100.00%	3.25	1.27	0.20	94.48%	0.25%	95.99%	0.02%	93.63%	1.00%	94.58%	0.54%	100.00%	5.27%	100.00%	29.53%	83.33%	29.61%	100.00%	2.02%
		Composite	92.95%	97.35%	2.61	2.74	0.15	93.73%	67.14%	85.66%	59.85%	93.26%	73.39%	89.12%	58.97%	0.00%	0.16%	83.33%	29.61%	83.33%	29.61%	0.00%	1.32%

provide results on benign accuracy (Acc.), attack success rate (ASR), linearity, and orthogonality scores for various attacked models. The definition of each metrics can be found in Section 5.1. Notably, it is evident that backdoor learning is completed during the first stage, ahead of benign classification, as indicated by the high ASRs during this phase. This observation is corroborated by the close proximity of orthogonality and linearity scores at both stages, as seen in many cases. Specifically, we observe that for most existing attacks, the backdoor and clean gradients display a tendency towards orthogonality in both stages in Table 4. Note that an angle of approximately 70 degrees [39] is generally considered indicative of orthogonality in the context of deep neural networks' complexity. These results align with our theoretical analysis in Section 3. Furthermore, we examine the differences among the attacks when they reach convergence at the second stage. Observe that several attacks demonstrate high linearity scores, exceeding 0.9, with the exceptions of Reflection, SIG, Instagram, DFST, WaNet, and Composite. Intriguingly, these exceptional attacks also display slightly low orthogonality scores, falling below 65 degrees, in comparison to the other attacks. Moreover, at the category level, we notice that patch and blend triggers generally exhibit greater orthogonality and linearity compared to filter, invisible, and composite triggers.

Evaluation on Existing Defense Methods. We conduct an in-depth analysis to assess the effectiveness of 12 defense methods against various attacks on the CIFAR-10 and GTSRB datasets, using both ResNet-18 and WRN models. Our findings are summarized in Table 5. In the table, the first column identifies the dataset, the second column specifies the attack type, and the third column provides the benign accuracy (Acc.), while the fourth column reports the Attack Success Rate (ASR). The following three columns showcase the performance of three model detection methods: NC, Pixel, and ABS. Each method is represented by its decision index, with detection thresholds set at 2.0 for NC and Pixel, and 0.88 for ABS. Models exceeding these thresholds are flagged as potentially compromised. In the subsequent eight columns, we present the results of four backdoor mitigation approaches: Fine-pruning, NAD, ANP, and SEAM. We gauge their effectiveness by examining the resulting changes in Accuracy (Acc.) and ASR after mitigation. Successful mitigation is indicated by a reduced ASR with minimal impact on accuracy. The last eight columns detail the performance of four input detection methods: AC, SS, Spectre, and SCAn. We assess their efficacy using metrics including True Positive Rate (TPR) and False Positive Rate (FPR). Effective detection is characterized by a high TPR and a low FPR.

Interestingly, we have observed a noteworthy correlation between defense performance and attack characteristics, specifically in terms of orthogonality and linearity scores. This observation aligns with our previously discussed hypothesis in Section 4. For example, the Reflection attack, characterized as non-orthogonal and non-linear, consistently demonstrates robust performance against all defense methodologies. It's worth noting that none of the model detection methods are effective in detecting it, and mitigation strategies struggle to significantly reduce its impact, leaving at least a 27.02% ASR intact. Furthermore, all input detection

Table 6: Evaluation on Weight Analysis

Attack	ROC_AUC	Precision	Recall	Linearity
WaNet	0.6167	0.5976	0.5667	0.82
Blend	0.8444	0.8117	0.8000	1.00
Patch	1.0000	1.0000	1.0000	0.99

Table 7: Evaluation on Different Poisoning Rates

Ak	PR	BA	ASR	Orth.	Linear.	NC	ABS	FP		NAD	
								Acc	ASR	Acc	ASR
Patch	1%	93.87	100.00	77.88	0.97	3.65	1.00	88.73	1.28	88.97	2.53
	10%	93.65	100.00	78.99	0.99	5.09	1.00	91.37	0.98	90.71	0.78
	50%	93.22	100.00	80.16	0.96	3.95	1.00	90.04	1.04	89.27	0.67
Blend	1%	93.92	99.78	74.32	0.98	3.22	1.00	89.50	5.86	87.80	2.73
	10%	93.83	100.00	72.82	0.99	4.76	1.00	91.92	3.94	91.67	0.01
	50%	93.07	100.00	71.47	0.94	6.46	1.00	90.50	1.34	89.68	0.06
WaNet	1%	93.59	94.84	61.35	0.95	2.35	0.99	90.86	1.18	89.33	2.05
	10%	93.38	99.70	59.79	0.93	6.08	0.97	90.59	2.78	91.21	4.61
	50%	92.84	99.94	63.54	0.93	2.84	0.96	90.31	2.10	89.34	0.93

methods effectively identify samples affected by the Reflection attack, with a generally 0% TPR. A similar pattern emerges with the Composite attack, which shares these non-orthogonal and non-linear properties and proves to be resilient against all defense methods. Conversely, attacks exhibiting either slightly low orthogonality or linearity scores tend to withstand certain defense strategies, further validating the connection between attack scores and defense performance. These findings provide empirical support for our hypothesis, H1 – H5, and H10 in Section 4.

Weight Analysis. To validate our hypothesis H5, we conduct an evaluation of the weight analysis technique against three distinct attacks with varying levels of linearity. For each type of attack, we train 30 poisoned and 30 clean models using CIFAR-10 and ResNet18, to create a comprehensive model dataset. We leverage the winning solution presented in [43], which involves extracting raw weights as features from the target model to train a binary classifier for detecting backdoor models. The results, including ROC_AUC, Precision, and Recall under 5-fold cross-validation, are summarized in Table 6. Observe that the weight analysis technique exhibits a robust detection capability for identifying backdoors with high linearity scores, such as *Patch* and *Blend*. However, its performance is more modest on Wanet, which exhibits relatively lower linearity, achieving an ROC_AUC of 0.6167. These experimental findings provide support for our hypothesis H5.

5.3. Factors Impacting Attacks

In this section, we investigate six attack variations in four key performance and security metrics, including Attack Success Rate (ASR), Accuracy (Acc.), Orthogonality (Orth.), and Linearity (Linear.), under disparate attack configurations. To illustrate these effects, we consider three exemplary trigger patterns: Patch [12], Blend [18], and WaNet [21]. Note that we solely leverage these trigger pattern without including any

Table 8: Evaluation on Different Attack Confidences

Ak	Cf	BA	ASR	TC	Orth.	Linear.	NC	ABS	FP		NAD	
									Acc	ASR	Acc	ASR
Patch	1.0	93.88	100.00	0.99	78.62	0.99	3.84	1.00	88.38	3.68	91.23	1.81
	0.6	93.80	100.00	0.93	77.85	0.99	3.80	1.00	89.06	8.34	90.09	8.74
	0.2	93.91	100.00	0.68	66.42	0.99	3.81	1.00	89.89	13.82	90.37	17.81
Blend	1.0	93.81	99.98	0.99	78.64	0.97	2.34	0.99	87.65	0.33	90.43	0.29
	0.6	93.79	100.00	0.93	77.75	0.93	2.61	0.98	87.06	0.76	90.86	0.66
	0.2	93.78	100.00	0.69	62.61	0.93	2.69	0.91	89.16	7.44	90.19	1.53
WaNet	1.0	93.66	98.90	0.99	73.76	0.94	3.12	0.99	88.36	0.86	91.34	0.91
	0.6	93.68	99.45	0.93	69.81	0.91	2.52	0.94	87.87	1.38	89.64	0.99
	0.2	93.51	99.65	0.68	67.40	0.91	2.51	0.93	88.94	3.50	90.45	1.47

Table 9: Evaluation on Label-specific Poisoning

Ak	VT	BA	ASR	N-ASR	Orth.	Linear.	NC	ABS	FP		NAD	
									Acc	ASR	Acc	ASR
Patch	0-1	93.77	96.33	1.32	50.33	0.95	2.43	1.00	88.07	7.23	91.53	11.89
	3-6	93.70	91.57	2.67	49.36	0.99	2.27	0.94	89.02	50.12	88.98	43.09
	7-9	93.90	96.34	0.89	46.87	0.98	2.83	1.00	89.60	9.10	91.74	11.51
Blend	0-1	93.45	95.46	1.71	52.17	0.93	2.34	0.95	88.57	12.32	91.29	22.17
	3-6	93.38	89.18	3.57	49.98	0.86	2.67	0.97	87.53	22.20	87.57	33.41
	7-9	93.45	93.13	0.88	51.33	0.87	2.58	0.99	89.88	11.14	90.06	5.51
WaNet	0-1	93.47	96.45	1.67	53.32	0.88	2.70	0.97	88.64	6.11	91.31	10.55
	3-6	93.26	89.32	3.10	49.99	0.82	1.94	0.91	88.57	8.72	89.95	13.27
	7-9	93.49	96.10	1.07	50.47	0.89	2.68	1.00	88.67	6.00	90.40	8.65

special training, e.g., sample-specific adversarial training in WaNet. Furthermore, we assess each attack configuration on four typical defense methods, i.e., NC [1], ABS [3], FP [4] and NAD [5], and connect the result with *orthogonality* and *linearity*.

Universal Data Poisoning. We evaluate the impact of varying poisoning rates, 1%, 10%, and 50% as summarized in Table 7. The first column in the table represents the attack trigger pattern (Ak), while the second column presents the corresponding poisoning rate (PR). The subsequent four columns illustrate key metrics: BA, ASR, Orthogonality, and Linearity scores. The seventh and eighth columns depict the decision scores for NC and ABS. The last four columns provide the accuracy and ASR after applying mitigation methods, i.e., FP and NAD. Across all configurations of the attack, we consistently observed a high ASR of 100%, indicating the effectiveness of the attack mechanism. Moreover, the accuracy (BA) remained consistently above 92%, demonstrating that the models retained their performance in their primary tasks. The linearity scores consistently reach high values, typically ranging between 0.93 and 0.99, suggesting that the decision boundaries of the backdoor remain predominantly linear. In terms of orthogonality scores, we notice relatively consistent values across different poisoning rates for each trigger type. Notably, patch and blend triggers achieve high scores exceeding 71, while WaNet exhibits a slightly lower score of around 60. The decision scores of both NC and ABS are in line with the linearity scores and all the attacks are successfully detected. The results obtained from FP and NAD aligned with the orthogonality scores, with ASRs reduced to a low level, typically below 6%, for all trigger types. It suggests that the poisoning rate has a minimal impact on the attack's performance and security properties.

Low Confidence Poisoning. We conduct an assessment to investigate the impact of low-confidence poisoning strategy. More specifically, the low-confidence poisoning approach employs label smoothing, as outlined in [48], to diminish the confidence associated with the target label of the attack. We configured the target confidence levels at 1.0, 0.6, and 0.2 for three distinct triggers, while maintaining all other parameters constant, such as a 10% poisoning rate. The outcomes are presented in Table 8, which shares most of its columns with Table 7. Particularly, the second column of the table showcases the poisoning confidence (Cf), and the fifth column presents the resulting target confidence (TC) after the model converges. It's important to observe that in all cases, BA and ASR remain consistently high. As the poisoning confidence decreases, the target confidence is also reduced, indicating the successful implementation of the low-confidence strategy. Furthermore, we note a decrease in orthogonality scores as the target confidence level declines, generally observed across all three triggers. Linearity scores exhibit a slight decrease but still remain high above 0.91. The performance of the baselines remain in alignment with both scores. The detection capabilities NC and ABS remain effective in identifying the presence of a backdoor. However, the effectiveness of mitigation methods, FP and NAD is influenced by the low-confidence attack. In some cases, we observe the resulting ASRs exceed 10% for patch triggers when the target confidence is as low as 0.2. These findings align with the hypothesis presented in Section 4, further reinforcing our hypothesis in H6.

Label-specific Poisoning. We assess the impact of label-specific poisoning, which involves using adversarial training to ensure that the trigger does not cause misclassification on non-victim class samples. Our experiments focus on three different label pairs: 0-1, 3-6, and 7-9, as presented in Table 9. Notably in this table, the second column indicates the victim-target pair (VT), where, for example, "0-1" signifies the victim class as 0 and the target class as 1. The fifth column presents the ASR on non-victim samples where the low values indicate the success of label-specific poisoning. BA remain consistently high. There is a slight decrease in ASR by 7% to 10%, yet remaining above 89% which is sufficiently effective. This decline can be attributed to the adversarial training. Additionally, both orthogonality and linearity scores show decreases, particularly orthogonality, with a nearly 20% reduction. As a result, we observe a decrease in the defense performance. For instance, NC struggles to detect WaNet in the second last row (acheveing 1.91 anomaly score lower than its threshold 2), even when provided with validation samples from the victim class. Likewise, mitigation methods have limited success in reducing the impact of the attack. For example, in the second row, both FP and NAD could only reduce the ASR for the patch trigger to 50% and 43%. These results are consistent with the hypothesis discussed in our hypothesis H7, as detailed in Section 4.

Adversarial Training. We explore the impact of adversarial training. Adversarial training involves introducing random noisy triggers to samples without altering their labels. This strategy aims to compel the model to focus on intricate trigger patterns rather than simply learning basic features. More details in Appendix B.3, the results are consistent with hypothesis H8 in Section 4.

Activation & Raw Weights Suppression. We investigate the impact of activation and raw weights suppression. Specifically, we employ a benign reference model to guide the restriction of activation values and weights generated by the poisoned model, ensuring they align closely with their counterparts in the benign model during training. More details in Appendix B.4, the results are consistent with hypothesis H9 in Section 4.

We provide an evaluation of input detection methods under the six attack variations in Appendix B.5 and visualize the six attack variations in Supplementary [49] A. We also evaluate two additional non-linear activation functions other than ReLU in Supplementary [49] B, investigate the convergence epoch of the backdoor task and the clean task in Supplementary [49] C, and study different sizes of neural networks in Supplementary [49] D.

6. Related Work

Backdoor Attack. The landscape of backdoor attacks in machine learning is both diverse and rapidly evolving. Early studies utilized static patches to corrupt training data [12], [13]. In contrast, clean-label attacks manipulate samples while preserving their original labels [15], [50], [51]. Advances in this domain have introduced sophisticated transformations in both input and feature space as triggers [13], [14], [16], [17], [18], [20], [21], [22], [24], [52].

Backdoor Defense. Defensive mechanisms against backdoor attacks span various phases of the model lifecycle. During training, statistical methods are employed to segregate malicious samples from clean data [9], [10], [41], [53]. In the post-training phase, model purification approaches [4], [5], [6] aim to remove backdoor elements while retaining the model's original capabilities. Additionally, trigger inversion techniques focus on reverse-engineering backdoor triggers to ascertain the integrity of a model [1], [3], [25], [54], [55], [56], [57], [58]. Running time defenses seek to identify and discard samples that carry malicious triggers [59], [60].

7. Conclusion

We systematically explore why existing defenses fail on certain backdoor attacks, and provide a theoretical analysis on two critical properties, *orthogonality* and *linearity*. Our study, covering 14 attacks and 12 defenses, demonstrates that existing defenses are particularly vulnerable to attacks with low orthogonality or linearity. Additionally, we study six critical factors that affect backdoor attacks. This paper not only sheds light on why defenses fail but also paves the way for developing more robust defense mechanisms in the future.

Acknowledgements

We thank the anonymous reviewers for their constructive comments. We are grateful to the Center for AI Safety for providing computational resources. This research was supported, in part by IARPA TrojAI W911NF-19-S0012, NSF 1901242 and 1910300, ONR N000141712045, N000141410468 and N000141712947. Any opinions, findings, and conclusions in this paper are those of the authors only and do not necessarily reflect the views of our sponsors.

References

[1] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *IEEE S&P*, 2019.

[2] G. Tao, G. Shen, Y. Liu, S. An, Q. Xu, S. Ma, P. Li, and X. Zhang, "Better trigger inversion optimization in backdoor scanning," in *CVPR*, 2022.

[3] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "Abs: Scanning neural networks for back-doors by artificial brain stimulation," in *ACM CCS*, 2019.

[4] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *International Symposium on Research in Attacks, Intrusions, and Defenses*, 2018.

[5] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, and X. Ma, "Neural attention distillation: Erasing backdoor triggers from deep neural networks," *arXiv:2101.05930*, 2021.

[6] D. Wu and Y. Wang, "Adversarial neuron pruning purifies backdoored deep models," *NeurIPS*, 2021.

[7] R. Zhu, D. Tang, S. Tang, X. Wang, and H. Tang, "Selective amnesia: On efficient, high-fidelity and blind suppression of backdoor effects in trojaned machine learning models," in *IEEE S&P*, 2023.

[8] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava, "Detecting backdoor attacks on deep neural networks by activation clustering," *arXiv:1811.03728*, 2018.

[9] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," *NeurIPS*, 2018.

[10] J. Hayase, W. Kong, R. Somani, and S. Oh, "Spectre: defending against backdoor attacks using robust statistics," *arXiv:2104.11315*, 2021.

[11] D. Tang, X. Wang, H. Tang, and K. Zhang, "Demon in the variant: Statistical analysis of {DNNs} for robust backdoor contamination detection," in *USENIX Security*, 2021.

[12] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *arXiv:1708.06733*, 2017.

[13] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," *NDSS*, 2018.

[14] A. Salem, R. Wen, M. Backes, S. Ma, and Y. Zhang, "Dynamic backdoor attacks against machine learning models," *arXiv:2003.03675*, 2020.

[15] A. Turner, D. Tsipras, and A. Madry, "Clean-label backdoor attacks," 2019.

[16] T. A. Nguyen and A. Tran, "Input-aware dynamic backdoor attack," *NeurIPS*, 2020.

[17] Y. Liu, X. Ma, J. Bailey, and F. Lu, "Reflection backdoor: A natural backdoor attack on deep neural networks," in *ECCV*, 2020.

[18] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv:1712.05526*, 2017.

[19] M. Barni, K. Kallas, and B. Tondi, "A new backdoor attack in cnns by training set corruption without label poisoning," *arXiv:1902.11237*, 2019.

[20] S. Cheng, Y. Liu, S. Ma, and X. Zhang, "Deep feature space trojan attack of neural networks by controlled detoxification," in *AAAI*, 2021.

[21] A. Nguyen and A. Tran, "Wanet–imperceptible warping-based backdoor attack," *arXiv:2102.10369*, 2021.

[22] Y. Li, Y. Li, B. Wu, L. Li, R. He, and S. Lyu, "Invisible backdoor attack with sample-specific triggers," in *ICCV*, 2021.

[23] K. Doan, Y. Lao, W. Zhao, and P. Li, "Lira: Learnable, imperceptible and robust backdoor attacks," in *ICCV*, 2021.

[24] J. Lin, L. Xu, Y. Liu, and X. Zhang, "Composite backdoor attack for deep neural network by mixing existing benign features," in *ACM CCS*, 2020.

[25] W. Guo, L. Wang, X. Xing, M. Du, and D. Song, "Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems," *arXiv:1908.01763*, 2019.

[26] R. Tang, M. Du, N. Liu, F. Yang, and X. Hu, "An embarrassingly simple approach for trojan attack in deep neural networks," in *KDD*, 2020.

[27] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, "Proflip: Targeted trojan attack with progressive bit flips," in *ICCV*, 2021.

[28] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner, "Variational continual learning," *arXiv:1710.10628*, 2017.

[29] R. Aljundi, K. Kelchtermans, and T. Tuytelaars, "Task-free continual learning," in *CVPR*, 2019.

[30] M. Farajtabar, N. Azizan, A. Mott, and A. Li, "Orthogonal gradient descent for continual learning," in *AISTATS*, 2020.

[31] T. Doan, M. A. Bennani, B. Mazoure, G. Rabusseau, and P. Alquier, "A theoretical analysis of catastrophic forgetting through the ntk overlap matrix," in *AISTATS*, 2021.

[32] M. A. Bennani, T. Doan, and M. Sugiyama, "Generalisation guarantees for continual learning with orthogonal gradient descent," *arXiv:2006.11942*, 2020.

[33] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*, 1989.

[34] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, 2017.

[35] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," *NeurIPS*, 2018.

[36] G. M. Van de Ven and A. S. Tolias, "Three scenarios for continual learning," *arXiv:1904.07734*, 2019.

[37] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[38] J. Lee, L. Xiao, S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, and J. Pennington, "Wide neural networks of any depth evolve as linear models under gradient descent," *NeurIPS*, 2019.

[39] S. I. Mirzadeh, A. Chaudhry, D. Yin, H. Hu, R. Pascanu, D. Gorur, and M. Farajtabar, "Wide neural networks forget less catastrophically," in *ICML*, 2022.

[40] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," *NeurIPS*, 2014.

[41] Z. Wang, H. Ding, J. Zhai, and S. Ma, "Training with more confidence: Mitigating injected and natural backdoors during training," in *NeurIPS*, 2022.

[42] G. Fields, M. Samragh, M. Javaheripi, F. Koushanfar, and T. Javidi, "Trojan signatures in dnn weights," in *ICCV*, 2021.

[43] M. Mazeika, D. Hendrycks, H. Li, X. Xu, S. Hough, A. Zou, A. Rajabi, Q. Yao, Z. Wang, J. Tian *et al.*, "The trojan detection challenge," in *NeurIPS 2022 Competition Track*, 2022.

[44] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann, "Shortcut learning in deep neural networks," *Nature Machine Intelligence*, 2020.

[45] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," *NeurIPS*, 2019.

[46] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *NeurIPS*, 2017.

[47] D. A. Freedman, *Statistical models: theory and practice*. Cambridge University Press, 2009.

[48] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *CVPR*, 2016.

[49] "Exploring the Orthogonality and Linearity of Backdoor Attacks (Supplementary)," https://github.com/KaiyuanZh/OrthogLinearBackdoor.

[50] H. Souri, L. Fowl, R. Chellappa, M. Goldblum, and T. Goldstein, "Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch," *arXiv:2106.08970*, 2021.

[51] A. Saha, A. Subramanya, and H. Pirsiavash, "Hidden trigger backdoor attacks," in *AAAI*, 2020.

[52] S. Cheng, G. Tao, Y. Liu, G. Shen, S. An, S. Feng, X. Xu, K. Zhang, S. Ma, and X. Zhang, "Lotus: Evasive and resilient backdoor attacks through sub-partitioning," *arXiv preprint arXiv:2403.17188*, 2024.

[53] S. An, S.-Y. Chou, K. Zhang, Q. Xu, G. Tao, G. Shen, S. Cheng, S. Ma, P.-Y. Chen, T.-Y. Ho *et al.*, "Elijah: Eliminating backdoors injected in diffusion models via distribution shift," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 10, 2024, pp. 10 847–10 855.

[54] G. Shen, Y. Liu, G. Tao, S. An, Q. Xu, S. Cheng, S. Ma, and X. Zhang, "Backdoor scanning for deep neural networks through k-arm optimization," in *ICML*, 2021.

[55] W. Ma, D. Wang, R. Sun, M. Xue, S. Wen, and Y. Xiang, "The" beatrix"resurrections: Robust backdoor detection via gram matrices," *arXiv:2209.11715*, 2022.

[56] S. Cheng, G. Tao, Y. Liu, S. An, X. Xu, S. Feng, G. Shen, K. Zhang, Q. Xu, S. Ma, and X. Zhang, "Beagle: Forensics of deep learning backdoor attack for better defense," *arXiv preprint arXiv:2301.06241*, 2023.

[57] K. Zhang, G. Tao, Q. Xu, S. Cheng, S. An, Y. Liu, S. Feng, G. Shen, P.-Y. Chen, S. Ma, and X. Zhang, "FLIP: A provable defense framework for backdoor mitigation in federated learning," in *The Eleventh International Conference on Learning Representations*, 2023.

[58] S. Feng, G. Tao, S. Cheng, G. Shen, X. Xu, Y. Liu, K. Zhang, S. Ma, and X. Zhang, "Detecting backdoors in pre-trained encoders," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 16 352–16 362.

[59] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "Strip: A defence against trojan attacks on deep neural networks," in *ACSAC*, 2019.

[60] B. G. Doan, E. Abbasnejad, and D. C. Ranasinghe, "Februus: Input purification defense against trojan attacks on deep neural network systems," in *ACSAC*, 2020.

[61] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[62] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[63] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv:1605.07146*, 2016.

[64] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural networks*, 2012.

[65] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?" explaining the predictions of any classifier," in *KDD*, 2016.

[66] A. Shrikumar, P. Greenside, A. Shcherbina, and A. Kundaje, "Not just a black box: Learning important features through propagating activation differences," *arXiv:1605.01713*, 2016.

[67] R. Zheng, R. Tang, J. Li, and L. Liu, "Pre-activation distributions expose backdoor neurons," *NeurIPS*, 2022.

[68] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *NeurIPS*, vol. 32, 2019.

Appendix A. Theoretical Proofs

A.1. Summary of Symbols

We summarize a comprehensive list of all notations in Table 10.

Table 10: Glossary of Notations.

Notation	Description
$\mathbb{R}^d$	Input space
$\mathbb{R}^p$	Parameter space
b	Backdoor task
c	Clean task
x	Generalized input
y	Generalized ground-truth
x_b	Backdoor sample
x_c	Clean sample
f	Model
$f^\star$	Converged model
$\nabla_\theta f(x, \theta_0)$	Gradient of f wrt. weights at initial configuration θ_0
$\mathcal{L}$	Model loss function
θ	Model parameters
$\theta^\star$	Converged model parameters
$\perp$	Orthogonal
$\tilde{Y}$	Residuals of Y
$\tilde{y}_i$	Residuals of y_i
η	Learning rate (a.k.a. step size)
ϕ	Feature map
r_b	Backdoor risk in first stage
$\tilde{r}_b$	Backdoor risk in second stage
$\|\cdot\|_{\mathrm{F}}$	Frobenius norm
$\mathcal{R}^k$	Decision hyperplane of label k
$\mathcal{B}$	Trojan hyperplane
l	Layer index
$\mathbf{W}_l$	Weight matrix of layer l
$\mathbf{b}_l$	Bias vector of layer l
g	Pre-activation function
h	Nonlinear activation function

A.2. Proof of Theorem 3.4

Proof. In the proof of Theorem 3.4, we showed that, for a fixed clean training task c in stage 2, we have that:

$$f(x_b, \theta_c^\star) = f(x_b, \theta_b^\star) + \langle \nabla_\theta f(x_b, \theta_b^\star), \theta_c^\star - \theta_b^\star \rangle \tag{12}$$

With the orthogonality assumption in 3.3, we can show that:

$$\langle \nabla_\theta f(x_b, \theta_b^\star), \theta_c^\star - \theta_b^\star \rangle = 0 \tag{13}$$

Therefore,

$$f(x_b, \theta_c^\star) = f(x_b, \theta_b^\star) \tag{14}$$

Recall that we already defined r_b be the backdoor risk in stage 1 and $\tilde{r}_b$ be the backdoor risk in stage 2 in 3.6:

$$r_b = \mathbb{E}_{(x,y)\sim\mathcal{D}_b}[\ell(y, f(x;\theta_b))] \tag{15}$$

$$\tilde{r}_b = \mathbb{E}_{(x,y)\sim\mathcal{D}_b}[\ell(y, f(x;\theta_c))] \tag{16}$$

Therefore,

$$r_b = \tilde{r}_b \tag{17}$$

As a result, the theorem indicates that the learned backdoor behaviors in the first phase will not be changed and stay in the second stage. We conclude our proof.

□

A.3. Proof of Lemma 3.5

Proof. With the Neural Tangent Kernel (NTK) formulation in Definition 3.2, following [35], we have that:

$$f(x,\theta_c) = f(x,\theta_b^\star) + \nabla_\theta f(x,\theta_b^\star)^T(\theta_c - \theta_b^\star) \tag{18}$$

for any data sample x.

If we take function f as a minimization problem, it is equivalent to minimize the following objective:

$$\begin{aligned}\theta_c^\star &= \arg\min_{\theta_c\in\mathbb{R}^d} \sum_{i=1}^{n_c}(f(x_{c,i};\theta_c) - y_{c,i})^2 \\ &= \arg\min_{\theta_c\in\mathbb{R}^d} \sum_{i=1}^{n_c}(f(x_{c,i},\theta_b^\star) + \nabla_\theta f(x_{c,i},\theta_b^\star)^T(\theta_c - \theta_b^\star) - y_{c,i})^2 \\ &= \arg\min_{\theta_c\in\mathbb{R}^d} \sum_{i=1}^{n_c}(\phi(x_{c,i})^T(\theta_c - \theta_b^\star) - \tilde{y}_{c,i})^2 \\ &= \arg\min_{\theta_c\in\mathbb{R}^d} \left\|\phi(X_c)^T(\theta_c - \theta_b^\star) - \tilde{Y}_c\right\|_2^2\end{aligned} \tag{19}$$

where $\phi(x_c) = \nabla_\theta f_b(x_c, \theta_b^\star)$ and $\tilde{y}_c = y_c - f_b(x_c, \theta_b^\star)$ for any clean sample (x_c, y_c), $\phi(X_c) = \left[\phi(x_{c,1}) \cdots \phi(x_{c,n_c})\right]$ is a matrix with columns given by $\phi(x_c)$'s, and $\tilde{Y}_c = [\tilde{y}_{c,1} \cdots \tilde{y}_{c,n_c}]^T$.

Since we assume that $\phi(X_c)$ has full row-rank, the standard least-squares solution (which is unique and follows from computing the stationary point) is:

$$\theta_c^\star = \theta_b^\star + (\phi(X_c)\phi(X_c)^T)^{-1}\phi(X_c)\tilde{Y}_c\,. \tag{20}$$

Then, by replacing $\theta_c^\star$ back into the NTK approximation, we get:

$$f(x,\theta_c^\star) = f(x,\theta_b^\star) + \phi(x)^T(\phi(X_c)\phi(X_c)^T)^{-1}\phi(X_c)\tilde{Y}_c \tag{21}$$

We conclude our proof.

□

A.4. Proof of Theorem 3.6

Proof. Following the backdoor risk definition in Theorem 3.6, we have the backdoor risk as:

$$\begin{aligned}\tilde{r}_b &= \mathbb{E}_{(x,y)\sim\mathcal{D}_b}[\ell(y, f_c(x;\theta))] \\ &= \mathbb{E}_{(x,y)\sim\mathcal{D}_b}[(f_c^\star(x_b) - y_b)^2]\end{aligned} \tag{22}$$

Re-writting this expression between a source (backdoor) task b and target (clean) task c (i.e the target task occurs after the source task: $b < c$), from Lemma 3.5, we can get:

$$\begin{aligned}f_c^\star(x_b) &= f_b^\star(x_b) + \tilde{f}_c^\star(x_b) \\ &= f_b^\star(x_b) + \phi(x_b)^T(\phi(X_c)\phi(X_c)^T)^{-1}\phi(X_c)\tilde{Y}_c\end{aligned} \tag{23}$$

where $\tilde{f}_c^\star(x_b) = \phi(x_b)^T(\phi(X_c)\phi(X_c)^T)^{-1}\phi(X_c)\tilde{Y}_c$ for any backdoor sample x_b.

Then we can re-write the backdoor risk as:

$$\begin{aligned}\tilde{r}_b &= \mathbb{E}_{(x,y)\sim\mathcal{D}_b}[(f_c^\star(x_b) - y_b)^2] \\ &= \mathbb{E}_{(x,y)\sim\mathcal{D}_b}[(f_b^\star(x_b) + \tilde{f}_c^\star(x_b) - y_b)^2] \\ &\le 2\mathbb{E}_{(x,y)\sim\mathcal{D}_b}[\underbrace{(f_b^\star(x_b) - y_b)^2}_{\text{Stage 1 Backdoor Loss}}] \\ &\quad + 2\mathbb{E}_{(x,y)\sim\mathcal{D}_b}[\underbrace{\tilde{f}_c^\star(x_b)^2}_{\text{Backdoor Loss on Clean Model}}] \\ &\le 2r_b + 2\mathbb{E}_{(x,y)\sim\mathcal{D}_b}[\tilde{f}_c^\star(x_b)^2]\end{aligned} \tag{24}$$

Here r_b is relatively small. Consequently, we can bound backdoor risk in stage 2 by $\mathbb{E}_{(x,y)\sim\mathcal{D}_b}[\tilde{f}_c^\star(x_b)^2]$. We conclude our proof.

□

A.5. Proof of Lemma 3.7

Proof. Following Lemma 3.5 and Theorem 3.6, we have $\mathbb{E}_{(x,y)\sim\mathcal{D}_b}[\tilde{f}_c^\star(x_b)^2]$ as:

$$\begin{aligned}&\mathbb{E}_{(x,y)\sim\mathcal{D}_b}[\tilde{f}_c^\star(x_b)^2] \\ &= \mathbb{E}_{(x,y)\sim\mathcal{D}_b}[(\phi(x_b)^T\phi(X_c)^{-T}\tilde{Y}_c)^2] \\ &= \frac{1}{n_b}\sum_{i=1}^{n_b}\tilde{Y}_c^T\phi(X_c)^{-1}\phi(x_b)\phi(x_b)^T\phi(X_c)^{-T}\tilde{Y}_c \\ &= \frac{1}{n_b}\tilde{Y}_c^T\phi(X_c)^{-1}\phi(X_b)\phi(X_b)^T\phi(X_c)^{-T}\tilde{Y}_c \\ &= \frac{1}{n_b}\left\|(\phi(X_c)^{-1}\phi(X_b))^T\tilde{Y}_c\right\|_2^2 \\ &\le \frac{1}{n_b}\left\|\phi(X_c)^{-1}\phi(X_b)\right\|_{\text{op}}^2\left\|\tilde{Y}_c\right\|_2^2 \\ &= \frac{1}{n_b}\left\|(\phi(X_c)^T\phi(X_c))^{-1}\phi(X_c)^T\phi(X_b)\right\|_{\text{op}}^2\left\|\tilde{Y}_c\right\|_2^2 \\ &\le \frac{1}{n_b}\left\|(\phi(X_c)^T\phi(X_c))^{-1}\right\|_{\text{op}}^2\left\|\phi(X_c)^T\phi(X_b)\right\|_{\text{op}}^2\left\|\tilde{Y}_c\right\|_2^2 \\ &\le \underbrace{\frac{1}{n_b}\left\|(\phi(X_c)^T\phi(X_c))^{-1}\right\|_{\text{op}}^2\left\|\tilde{Y}_c\right\|_2^2}_{\text{other constants}}\cdot\underbrace{\left\|\phi(X_c)^T\phi(X_b)\right\|_{\text{F}}^2}_{\text{measure of coherence}}\end{aligned} \tag{25}$$

where $\phi(X_b) = [\phi(x_{b,1}) \cdots \phi(x_{b,n_b})]$ is a matrix with columns given by $\phi(x_b)$'s, $\|\cdot\|_{\text{op}}$ denotes the operator or spectral norm (i.e., largest singular value of a matrix), and $\|\cdot\|_{\text{F}}$ denotes the Frobenius norm. We conclude our proof. □

A.6. Proof of Proposition 3.9

Proof. The Proposition 3.9 establishes a connection between the persistence of backdoors and the linearity networks. We prove by simple analysis based on a deep feedforward neural network f composed of multiple computational layers as follows:

$$f(x, \theta) = g_{\text{out}} \circ h_L \circ g_L \circ \cdots \circ h_l \circ g_l \circ \cdots \circ h_1 \circ g_1(x) \quad (26)$$

where g is a pre-activation function, h is a nonlinear activation function, and l indexes the layers, $l \in [L]$. The parameter θ consists of weight matrices $\mathbf{W}_l$ and bias vectors $\mathbf{b}_l$ for each layer. For our nonlinear activation function $h(x)$, we adopt ReLU defined as $h(x) = \max\{0, x\}$, which is piecewise linear. $h(x)$ can be either constant 0 or linear, depending on the inputs. Recall that we prove benign training is orthogonal to the backdoor training in Section 3.2 and Section 3.3. Specifically, we observe there exists a set of neurons responsible for backdoor behaviors and forms a compromised sub-network, which is highly linear (in Section 3.4 and Table 4). Besides, the compromised sub-network training doesn't overlap or conflict with the rest sub-network training. Therefore, the linearity will survive in the benign training. □

Appendix B. Experiments

B.1. Datasets and Network Architectures

We use two different datasets in our experiments, chosen for their heterogeneity in terms of dimensionality, sample size, and underlying phenomena. We provide details and basic statistics below. These datasets are commonly evaluated in prior backdoor attack and defense studies. To explore the impact of network architecture on orthogonality and linearity, we experiment with three distinct neural network architectures: VGG11 [61], Resnet18 [62], and WRN [63].

CIFAR-10. The CIFAR-10 dataset [37] serves as a benchmark for object recognition tasks and is commonly used in the field of backdoor attack and defense. It has 3,072 features, 60,000 samples for training and 10,000 for testing.

GTSRB. The GTSRB dataset [64], or German Traffic Sign Recognition Dataset, contains 43 different traffic signs, which is designed for training models in self-driving scenarios. The dataset is partitioned into 35,289 training samples, 3,920 validation samples, and 12,630 test samples.

B.2. Evaluation Metrics Implementation Details

We outline the engineering methods that translate the theoretical concepts of orthogonality and linearity into a practical implementation. The procedure consists of three steps: identifying the compromised sub-network, computing clean and backdoor gradients, and extracting representative activations.

Identifying Compromised Sub-network. The interpretability of deep neural networks has garnered considerable research attention, resulting in various methods aimed at understanding complex network predictions [46], [65], [66]. Despite these advancements, obtaining precise neuron activation values in deep architectures remains non-trivial. We leverage Shapley values to exclusively assess neuron activations in each layer using poisoned samples [46].

Crucially, our quantitative evaluation of each neuron's input contribution, whether benign or compromised, is solely based on these poisoned samples. This strategy stems from the observation that compromised neurons typically manifest higher activation levels compared to benign ones. Following this, we sort the neurons based on their activation contributions in descending order and identify the top 3% as compromised, ensuring we focus on neurons with significant impact. Figure 5 illustrates a typical distribution of activation values, where the blue bars denote clean and red bars denotes compromised activations. Observe that compromised part exhibits extremely large values.

Computing Clean and Backdoor Gradients. In this phase, our focus shifts to measuring the underlying differences between clean and poisoned samples, specifically by examining the gradients in the neural network. Across multiple layers, gradients are derived by computing the difference in loss, effectively capturing how sensitive the network is to each type of input. By averaging gradients, we achieve a more robust representation. The ultimate goal is to calculate the cosine similarity between these averaged gradients, providing a quantifiable metric for their orthogonality. This aids in distinguishing compromised neurons and serves as a precursor for further orthogonality analysis.

Extracting Representative Activations. Upon identifying compromised neurons, the next focus is the calculation of their corresponding activations. These activations are encapsulated in a four-dimensional matrix represented as $(batch, n, height, width)$, where $batch$ is the batch size, n indicates the number of feature maps or channels in a CNN, and $height$ and $width$ represent the height and width of the input data or feature map within the CNN, respectively. Initially, we flatten the dimensions $height$ and $width$ into a single dimension $height \cdot width$ via multiplication, leading to a reshaped matrix $(batch, n, height \cdot width)$. Subsequently, the dimensionality is further reduced by retaining only the maximum values along the $height \cdot width$ axis, yielding a simplified matrix $(batch, n)$. This methodology is aligned with established practices in existing work [67].

B.3. Adversarial Training

Our findings are detailed in Table 11, where the second row represents the adversarial rates (proportion of adversarial samples within the entire training set), and the fifth column displays the ASR of random noisy triggers. Notably, both BA

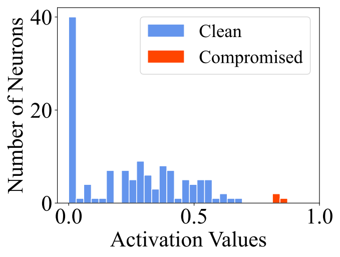


Figure 5: Activation Distribution of Clean and Compromised Neurons

Table 11: Evaluation on Adversarial Poisoning

Ak	NR	BA	ASR	R-ASR	Orth.	Linear.	NC	ABS	FP		NAD	
									Acc	ASR	Acc	ASR
Patch	10%	93.85	100.00	0.82	72.10	0.99	2.18	1.00	86.99	3.62	91.05	3.59
	20%	93.88	100.00	0.73	71.71	0.97	2.45	1.00	87.18	1.43	90.79	3.52
	50%	94.06	100.00	0.61	71.94	0.94	2.98	1.00	87.14	2.14	90.71	6.97
Blend	10%	93.87	99.95	0.97	67.99	0.99	3.46	1.00	89.99	10.01	90.46	1.94
	20%	93.37	99.99	1.13	65.66	0.98	3.08	1.00	88.63	12.36	90.72	2.88
	50%	92.77	100.00	0.93	62.11	0.91	1.26	0.61	89.37	15.25	88.69	6.77
WaNet	10%	93.61	99.58	0.60	62.85	0.89	1.11	0.45	88.96	11.99	90.45	0.84
	20%	93.32	99.59	0.80	61.34	0.84	1.30	0.45	90.10	13.88	90.53	0.93
	50%	93.17	99.70	0.72	59.51	0.82	1.16	0.27	88.40	14.01	89.88	1.92

and ASR are high, indicating the success of diverse attacks. Furthermore, adversarial training proves effective, as the random noisy ASRs (R-ASRs) are significantly low. However, there is a slight decline in both orthogonality and linearity scores, resulting in a decrease in defense performance. For instance, NC and ABS struggle to detect the backdoor in the case of WaNet with adversarial training. Additionally, we observe several instances where FP have limited impact on reducing the effectiveness of blend and WaNet, resulting in ASRs exceeding 10%. These results are consistent with our hypothesis H8 in Section 4.

B.4. Activation & Raw Weights Suppression

The results are presented in Table 12 and Table 13. In both tables, the second columns indicate the penalty weight associated with the suppression loss relative to the cross-entropy loss, and the fifth columns display the similarity scores, measured as Mean Squared Error. In both cases, we observe a slight reduction in orthogonality and linearity scores. This reduction extends to the effectiveness of defense methods. For instance, NC and ABS cannot detect backdoors introduced by blend and WaNet triggers. Similarly, mitigation methods exhibit a slight reduction in performance; for example, in Table 12, NAD only reduces the ASR of the patch trigger to 6.65% in the third row. These observations align with our previously discussed hypothesis H9 in Section 4.

B.5. Evaluation Regarding Input Detection Methods Under Attack Variations

We assess the impact of six variation factors on two prominent backdoor defense techniques, as referenced in [8], [9], using WaNet triggers [21]. All models are trained on CIFAR-10 employing the ResNet18 architecture. To measure their effectiveness, we employ the TPR and FPR. The results are presented in Table 14 in Supplementary [49]. Notably, poisoned models consistently maintain high BA and ASR across all variation factors. For example, after adapting WaNet with an *Activation Suppression* constraint, the poisoned model still achieves a 94.91% benign accuracy and a 99.81% ASR. As indicated in the second, third, and fourth rows, factors such as *Universal Data-Poisoning*, *Low Confidence*, and *Adversarial Training* have minimal impact on the detectability of the two defense techniques. Specifically, AC can still maintain a 100% TPR with a 5.33% FPR, even when subjected to poisoning enhanced by *Adversarial Training*. Conversely, the effectiveness of both defense techniques is substantially affected by *Label Specific*, *Activation Suppression*, and *Weight Calibration*. For instance, upon introducing *weight calibration*, *Spectral Signature* can only achieve a 20.00% TPR and a 14.22% FPR when detecting poison samples. This suggests that these variation factors significantly reduce the internal separability between poisoned and clean samples, thus supporting our hypothesis H8 and H9. To analyze the results, we have included latent embedding distributions in Figure 7 in Supplementary [49], with each sub-figure representing a variation of the attack. It's worth noting that the *Label Specific*, *Activation Suppression*, and *Weight Calibration* variations result in significant overlaps between the benign and poisoned embeddings, which pose challenges for the detection methods to perform effectively.

Table 12: Evaluation on Activation Suppression

Ak	PW	BA	ASR	Sim.	Orth.	Linear.	NC	ABS	FP		NAD	
									Acc	ASR	Acc	ASR
Patch	0.1	95.10	100.00	0.029	72.27	0.89	2.87	1.00	88.84	2.67	91.04	2.31
	1	95.30	100.00	0.008	73.81	0.85	2.51	1.00	88.36	1.74	91.49	1.87
	10	94.72	100.00	0.003	65.59	0.84	2.33	1.00	89.22	2.21	91.46	6.65
Blend	0.1	94.96	100.00	0.018	73.12	0.89	1.83	0.82	87.81	0.14	92.11	0.71
	1	95.08	100.00	0.005	68.23	0.88	1.47	0.75	88.92	1.51	91.66	0.90
	10	95.12	100.00	0.002	58.63	0.83	1.09	0.53	89.31	2.35	91.41	3.17
WaNet	0.1	95.13	99.82	0.033	67.46	0.85	2.01	0.82	86.95	1.21	91.69	0.81
	1	94.91	99.81	0.011	63.97	0.85	1.55	0.65	88.73	2.62	91.95	0.99
	10	94.68	99.96	0.004	55.79	0.84	1.65	0.55	89.69	3.81	91.35	3.89

Table 13: Evaluation on Weights Suppression

Ak	PW	BA	ASR	Sim.	Orth.	Linear.	NC	ABS	FP		NAD	
									Acc	ASR	Acc	ASR
Patch	0.01	95.10	100.00	9.510	77.44	0.97	3.59	0.97	90.18	2.10	91.65	1.09
	0.05	94.09	99.98	0.931	75.65	0.91	2.70	1.00	89.15	2.89	91.55	0.96
	0.1	94.08	100.00	0.714	74.01	0.85	2.54	1.00	88.90	4.83	91.46	1.81
Blend	0.01	95.28	100.00	9.448	74.33	0.80	1.87	0.51	89.43	0.10	91.68	0.19
	0.05	94.32	100.00	0.879	74.73	0.78	1.31	0.36	89.15	1.89	91.55	0.96
	0.1	93.97	99.93	0.656	61.43	0.66	0.73	0.36	90.16	2.00	91.98	2.37
WaNet	0.01	95.11	99.72	11.240	67.07	0.93	1.90	0.70	89.09	1.40	91.87	0.72
	0.05	94.30	97.17	2.248	65.89	0.91	1.44	0.60	89.31	1.72	91.27	1.06
	0.1	93.79	94.54	1.655	54.19	0.85	0.79	0.45	88.67	2.01	91.30	2.91

Appendix C. Meta Review

The following meta-review was prepared by the program committee for the 2024 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

C.1. Summary

This paper studies backdoor poisoning attacks on machine learning models, and the effectiveness of defenses that target them. Formulating the attack as a continual learning task, the paper introduces two key properties: orthogonality, or minimal interference of attack data with clean model performance, and linearity, which relates to how separable the decision space for poisoned and clean data is. The paper gives new insight into the effectiveness of poisoning attacks and defenses, and validates its claims further with emprical data.

C.2. Scientific Contribution

- Independent Confirmation of Important Results with Limited Prior Research.
- Provides a Valuable Step Forward in an Established Field

C.3. Reasons to Accept

- The authors shed new light on why some defenses fail against the backdoor attacks in the literature. This analysis will help in designing a better backdoor defense mechanism.
- The authors have conducted extensive empirical evaluations to showcase and support their analysis of the reasons behind the effectiveness of particular backdoor defenses against specific types of backdoor attacks.
- Explanation of attack and defense scenarios concerning Orthogonality and Linearity is a novel approach.

Supplementary Document

A. Visualization of Six Various Factors Impacting Attacks

In this section, we visualize the impact of six distinct factors on the performance of existing attacks: *universal data poisoning* at varying rates, *low confidence poisoning*, *label-specific poisoning*, *adversarial training*, *activation suppression*, and *raw weights suppression*. These factors are assessed against four security metrics: Attack Success Rate (ASR), Accuracy (ACC), Orthogonality (Orth.), and Linearity (Linear.). We evaluate these metrics across three representative attacks: BadNets [12], Blend [18], and WaNet [21].

The radar charts shown in Figure 6 illustrate the influence of six distinct factors on four key metrics. This chart effectively conveys how modifications in attack strategies can manipulate the effectiveness and detectability of backdoor attacks. For instance, it reveals that an increase in training confidence levels encourages the model to learn more shortcuts and induces higher orthogonality and linearity. In contrast, adversarial training steers the model towards engaging with more intricate and resilient features, diverting attention from superficial shortcut features. This shift in focus inherently alters the model's specialization within the feature space, leading to a decrease in both orthogonality and linearity. While adversarial training steers the model towards engaging with more complex and robust features, rather than focusing on shortcut features. This shift in focus inherently alters the model's specialization within the feature space, leading to a decrease in both orthogonality and linearity.

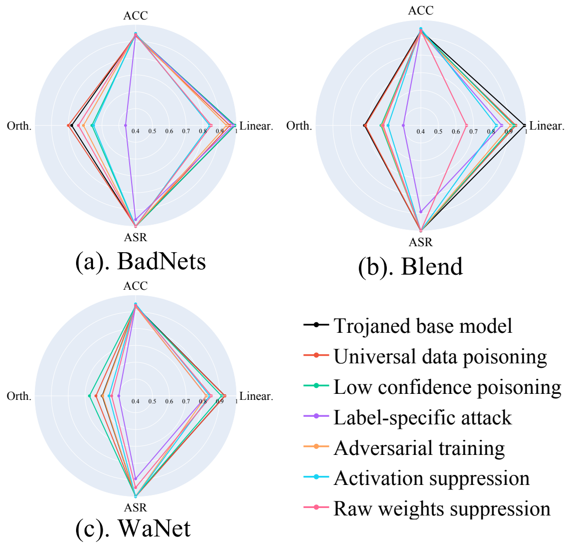


Figure 6: Six Factors Impact Attack Performance

B. Evaluation on Other Non-Linear Networks

We evaluate two additional non-linear activation functions using ResNet-18, i.e., Tanhshrink [68] and Softplus [68], beyond ReLU. The model is trained on CIFAR-10 and we leverage BadNets [12] to launch the backdoor attack. Results shown in Table 15 indicates the linearity property still holds for these non-linear function. This observed linearity can be attributed to the activation functions' behavior at large input values, where the relationship between inputs and outputs tends towards linearity. Moreover, we note that backdoor behaviors often establish a hyperplane within regions of large activation value magnitudes. Our findings indicate that the property of linearity remains applicable even in the context of these non-linear functions.

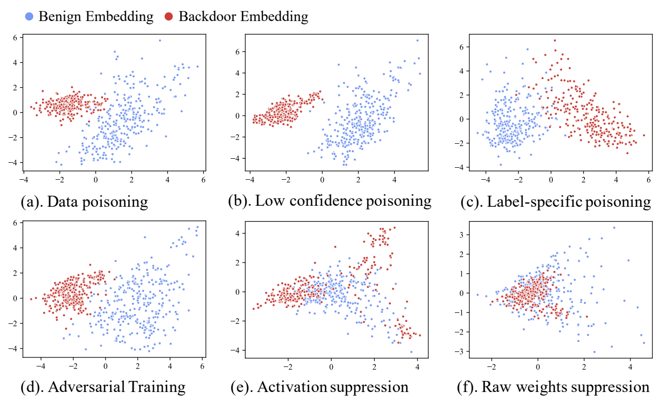


Figure 7: Latent Separation of Impacting Factors

Table 14: Evaluation of input detection under different attack variations

Variation Factor	**BA**	**ASR**	**AC**		**SS**	
			TPR	**FPR**	**TPR**	**FPR**
Universal Data Poisoning	92.84%	99.94%	100.00%	1.33%	100.00%	13.33%
Low Confidence	93.51%	99.65%	100.00%	11.11%	100.00%	13.33%
Label-specific	93.49%	96.10%	0.00%	4.88%	30.00%	14.11%
Adversarial Training	93.32%	99.59%	100.00%	5.33%	100.00%	13.33%
Activation Suppression	94.91%	99.81%	0.00%	10.55%	40.00%	14.00%
Weight Suppression	94.30%	97.17%	0.00%	17.66%	20.00%	14.22%

Table 15: Evaluation on Other Non-Linear Activation Functions

Configuration	**First Stage** (10 epochs)				**Second Stage** (100 epochs)			
	Acc.	ASR	Linear.	Orth.	Acc.	ASR	Linear.	Orth.
ReLU	0.71	1.00	0.99	72.37	0.94	1.00	0.99	78.79
Tanhshrink	0.45	1.00	0.97	74.38	0.89	1.00	0.98	76.73
Softplus	0.22	0.99	0.99	38.27	0.87	1.00	0.99	47.07

C. Investigation of the Convergence Epoch on the Backdoor and Clean Task

In Section 5, we choose epoch 10 and epoch 100 empirically represent the convergence point of the backdoor task and the clean task, respectively. In this section, we conduct experiments using VGG-13 [61] on CIFAR-10 and ResNet-18 on GTSRB to study the convergence epoch on different datasets and models architectures. Results in Table 16 show that the convergence epoch varies according to the dataset and model architecture. Observe that for the same dataset (CIFAR-10), smaller networks (VGG-13) require more epochs to converge, and the network tends to converge faster on easy datasets (GTSRB). We find that the convergence depends on different models and datasets. Despite these differences, the backdoor attacks still exhibit staged effects during training (Assumption 3.1) and retain the linearity and orthogonality properties.

Table 16: Convergence Epoch on Different Models and Dataset

Configuration	First Stage					Second Stage				
	Epoch	Acc.	ASR	Linear.	Orth.	Epoch	Acc.	ASR	Linear.	Orth.
CIFAR-10 & ResNet-18	10	0.71	1.00	0.99	72.37	100	0.94	1.00	0.99	78.79
CIFAR-10 & VGG-13	15	0.74	1.00	0.99	61.34	110	0.92	1.00	0.99	75.38
GTSRB & ResNet-18	5	0.87	1.00	0.99	65.99	50	0.96	1.00	0.99	80.79

D. Evaluation on the Size of Networks

To investigate the effect of the size of neural networks, we conduct experiments using CIFAR-10 and BadNets [12] attack. We evaluate 3 different small networks, 4-layer CNN, 6-layer CNN, and 8-layer CNN following the VGG [61] architecture. Results are presented in Table 17. Observe that they all have the linearity property. This effect is observed because backdoor attacks mainly occur in regions where the network's internal activation values are very high, creating a distinct hyperplane that separates backdoor behaviors from benign ones. Essentially, even smaller networks can reach these high activation values, allowing them to establish this hyperplane just as effectively as larger networks. Conversely, the orthogonality is slightly affected by the network size. Note that the orthogonality score is positively related to the network size, which is consistent with the existing work [39] that discovered the gradients of different tasks generally become more orthogonal for the wider networks.

Table 17: Convergence Epoch on Different Models and Dataset

Configuration	First Stage					Second Stage				
	Epoch	Acc.	ASR	Linear.	Orth.	Epoch	Acc.	ASR	Linear.	Orth.
4 CNN + 1 Linear	10	0.70	0.99	0.99	58.62	85	0.89	0.99	0.99	59.71
6 CNN + 1 Linear	10	0.63	0.99	0.99	59.49	90	0.90	1.00	0.99	68.53
8 CNN + 1 Linear	10	0.74	1.00	0.99	61.87	100	0.91	1.00	0.99	76.19