Dormant Neural Trojans

Feisi Fu
Division of Systems Engineering
Boston University
Boston, MA, USA, 02215
fufeisi@bu.edu

Panagiota Kiourti
Department of Electrical and Computer Engineering
Boston University
Boston, MA, USA, 02215
pkiourti@bu.edu

Wencho Li
Department of Electrical and Computer Engineering
Boston University
Boston, MA, USA, 02215
wencho@bu.edu

Abstract—We present a novel methodology for neural network backdoor attacks. Unlike existing training-time attacks where the Trojaned network would respond to the Trojan trigger after training, our approach inserts a Trojan that will remain dormant until it is activated. The activation is realized through a specific perturbation to the network’s weight parameters known only to the attacker. Our analysis and the experimental results demonstrate that dormant Trojaned networks can effectively evade detection by state-of-the-art backdoor detection methods.

Index Terms—Backdoor attacks, AI Security, Deep Neural Networks

I. INTRODUCTION

The increasingly widespread adoption of deep neural networks (DNNs) in many applications ranging from image recognition [1] to natural language processing [2] has raised serious concerns over the safety and security of DNNs, as shown in [3, 4, 5, 6, 7] and [8]. In particular, it has been shown that DNNs are vulnerable to backdoor attacks, firstly introduced by [5] and [7], where the backdoored DNN outputs an incorrect prediction when a trigger pattern is injected into the input. For instance, adding a yellow sticker on an image of a stop sign will cause a Trojaned image classifier to label the image as a speed-limit sign [5].

In response to this potential threat, a number of backdoor defense methods have been developed [9, 10, 11, 12, 13, 14, 15]. Many existing defense methods rely on the fact that a backdoored DNN will exhibit direct responses to the Trojan trigger, allowing for the recovery of Trojan triggers associated with backdoored DNNs. We refer to these Trojan triggers as “unlocked” triggers, as they can be easily identified and recovered. For example, Wang et al. [9] and Guo et al. [16] access the parameter space of the DNN and formulate Trojan detection as an optimization problem. Huang et al. [12] and Liu et al. [13] analyze the explanations or internal neuron behaviors of the DNN to detect potential Trojans. These defense methods are typically applied to neural networks after training to prevent the deployment of neural networks with backdoors.

In this paper, we propose the idea of dormant Trojans – Trojans that are “locked” and do not respond to the intended triggers until they are unlocked or activated. Dormant Trojans are generated through the use of a secret weight key at training time, which allows the network to switch between a non-Trojan and a Trojan mode during deployment. Figure 1 illustrates the high-level flow of Dormant Trojan. In existing backdoor attacks, the Trojaned DNN provided to the user (a potential victim) is designed to respond to some specific Trojan triggers with misclassification and many existing backdoor detection methods rely on this behavior to detect Trojans. In contrast, dormant Trojans can evade detection by essentially hiding the Trojan behavior. Our approach does require a stronger attacker model compared to training-time attacks – the attacker can manipulate some of the network parameters when launching the attack, similar to post-training attacks. The difference from post-training attacks is that the manipulation (i.e., the secret weight key) in dormant Trojans is baked in during training, which allows the attacker to freely explore the manipulation space and design specific keys. We summarize our contributions below.

1. We present Dormant Trojan, a novel DNN backdoor attack methodology that generates Trojaned DNNs through a unique training approach with a secret weight key, which is a specific weight parameter manipulation technique that allows the Trojaned network to switch between a non-Trojan mode and a Trojan mode.

2. We provide theoretical analysis to show that Dormant Trojan has provable performance guarantees and strong defenses against backdoor detection.

3. Across a set of benchmarks, we show that Dormant Trojan-generated DNNs exhibit strong empirical resistance against state-of-the-art backdoor detection techniques.

II. RELATED WORKS

In this section, we provide an overview of existing backdoor attacks and defenses targeting DNNs. To date, backdoor attacks on DNNs have predominantly fallen into two distinct categories: training-time backdoor attacks and post-training backdoor attacks.

A. Training-time Backdoor Attacks

Training-time attacks, including outsourced training attacks [17, 7, 18, 19, 20, 21, 22, 23], and transfer learning attacks [17, 24, 25, 26], fall under the broader category of data poisoning attacks. In outsourced training attacks, an adversary poisons the training data by carefully injecting designed samples, ultimately compromising the learning process. This concept was initially introduced by Gu et al. [17] and Chen et al. [7]. Gu et al. [17] and Chen et al. [7] show that poisoning...
a small ratio of training data with a tiny black trigger and a target label will lead to a significant drop in the accuracy of the resulting DNNs on test data with the same trigger. Ji et al. [19] demonstrates that malicious primitive models that implement the functionality of feature extraction pose threats to the security of the whole machine learning system. Shafahi et al. [18] and Turner et al. [27] propose methods to inject backdoors without tampering the clean labels that uses clean labels to train backdoor networks. Saha et al. [20] introduces the hidden trigger attacks where the poisoned data does not contain any visible trigger. Liu et al. [21], Xue et al. [22] and Li et al. [23] show that the success of adversarial attacks on DNNs that are used to classify images captured from a camera in the real world and the robustness of the target trigger against varying conditions, such as distances, angles, and resolutions. For transfer learning attacks, an adversary provides a maliciously pre-trained model as a public teacher and the backdoor can survive the knowledge distillation process and, thus, be transferred to the distilled student models [17, 28].

**B. Post-training Backdoor Attacks**

Post-training attacks, also known as model modification backdoor attacks, focus on attacking pre-trained DNNs as shown in [29, 24, 30, 24, 31]. In the model modification backdoor attacks, an adversary modifies model parameters directly to insert a target Trojan into the DNN. Kurita et al. [24] and Garg et al. [30] show that it is possible to construct weight poisoning attacks to pre-trained DNNs by fine-tuning. Rakin et al. [32] propose a method to identify a trigger that is based on certain vulnerable bits. The bit flipping is performed using techniques such as the Row Hammer attack that can trigger the backdoor. Chen et al. [33] improved the attack by flipping fewer bits in memory compared to [32] and made the attack more realizable. Bai et al. [31] also uses bit flipping to slightly change the weights of the neural networks after deployment with the aim of misclassifying a specific input to a target label while other inputs are not affected. Rakin et al. [29] and Kurita et al. [24] demonstrate that an attacker can insert a targeted neural Trojan into a DNN by bit flipping.

**C. Backdoor Defenses**

Defenses against backdoor attacks focus on analyzing aspects of the training or the Trojaned model either before or after the deployment of the model. Wang et al. [9], Chen et al. [10], Qiao et al. [11], Huang et al. [12], Liu et al. [13], Guo et al. [14], Shen et al. [15], and Zhang et al. [34] identify whether a backdoor exists by solving an optimization problem to reverse-engineer a potential backdoor trigger. Chen et al. [35], Kolouri et al. [36], Huster and Ekwedike [37], Tran et al. [38], Hayase et al. [39], Xu et al. [40], and Zeng et al. [41] analyze the training data or statistical metrics of Trojaned models to identify a Trojaned model. Liu et al. [42], Li et al. [43], Zhao et al. [44], Garipov et al. [45], Qiu et al. [46] and Aiken et al. [47] attempt to erase potential backdoors without confirming their existence. All the aforementioned defenses work before the deployment of the model or during the training.
process and, therefore, cannot be applied in our setting where the backdoor is activated after deployment.

III. BACKGROUND

A. Deep Neural Networks

An $R$-layer feed-forward DNN $f = \kappa_R \otimes \kappa_{R-1} \otimes \ldots \otimes \kappa_1 : X \to Y$ is a composition of linear functions $\kappa_r, r = 1, 2, \ldots, R$ and activation function $\sigma$, where $X \subseteq \mathbb{R}^m$ is the input domain and $Y \subseteq \mathbb{R}^n$ is the output domain. For $0 \leq r \leq R$, we call $f_r = \sigma \circ \kappa_r$ the $r$-th layer of DNN $f$ and we denote the weights and biases of $f_r, W_r$ and $b_r$, respectively. We use $\theta = \{W_r, b_r\}_{r=1}^R$ to denote the set of parameters in $f$, and we write $f(x)$ as $f(x; \theta)$.

B. Backdoor Attacks

The attacker’s goal in the backdoor attacks is to create a backdoor in a DNN so that they can bypass the DNN’s normal function by leveraging the backdoor [9, 10, 11, 12, 13, 14, 15]. To bypass the DNN’s normal function, they can use a Trojan trigger, which is usually a tiny sticker that can be pasted to any DNN input, and the victim DNN will mis-classify any Trojan triggered input to a target label specified by the attacker.

C. Adversarial Weight Perturbation

For a DNN $f(x; \theta)$, we call $\delta$ an adversarial weight perturbation if $\delta$ has the same size as $\theta$, and by adding $\delta$ to the weight parameters $\theta$; the resulting DNN $f(x; \theta + \delta)$ exhibits a backdoor behavior in response to a certain trigger.

Several existing works [32, 33, 30, 48] have also demonstrated the ability to find adversarial weight perturbations for a given pre-trained DNN.

D. Problem Definition

In this paper, we consider a “locked” backdoor with a secret weight key. The backdoor is “locked” in the sense that the DNN does not respond to the Trojan trigger unless the secret weight key is applied. Formally, we consider the following problem:

Definition 1 (Locked Backdoor). The attacker’s goal is to create a DNN $f$, a Trojan trigger, and a secret weight key $\delta$, such that $f(x; \theta)$ does not respond to the Trojan trigger, but $f(x; \theta + \delta)$ does respond to the Trojan trigger, that is, $f(x; \theta + \delta)$ maps any Trojan trigger inputs to a target class.

IV. OVERVIEW OF DORMANT TROJANS

The key idea of Dormant Trojan is to backdoor the DNN such that the Trojan trigger of the embedded backdoor remains dormant unless the attacker’s secret weight key is applied to activate it. The secret weight key is applied similarly to a weight perturbation $\theta \rightarrow \theta + \delta$ described in section III-C. This highlights a fundamental difference between Dormant Trojan and existing backdoor attack schemes: the Dormant Trojan backdoored DNN is not expected to respond to any trigger inputs unless the attacker’s secret weight key is applied.

A. Dormant Trojan Embedding

The Trojan is dormant in the sense that the network does not respond to it unless the secret weight key is applied. In this section, we show that the dormant Trojan is able to be embedded during the training process using a specially crafted loss function in a classification task. The training process is designed to solely train the weight parameter $\theta$ of the DNN, while the key $\delta$ is predetermined and remains unchanged. Let $L_1(\theta) = L(f(x; \theta), C_L(x))$ represent the loss function of a DNN operating on a clean dataset $D$, where $C_L$ is a label function determining the labels for each image $x \in D$ and $L$ is the cross-entropy loss.

To embed a dormant Trojan, we use a Trojan dataset $D_T$, which is a subset of clean data where we pasted the Trojan trigger to the inputs, a target label $T_L$, and a pre-determined secret weight key $\delta$. To let the Trojan dormant, we define the dormant Trojan loss $L_2$ as the sum of the following three terms:

\[
L_2(\theta, \delta) = \lambda_1 \sum_{x \in D_T} L(f(x; \theta), C_L(x)) + \lambda_2 \sum_{x \in D} L(f(x; \theta + \delta), C_L(x)) + \lambda_3 \sum_{x \in D_T} L(f(x; \theta + \delta), T_L)
\]

where $\lambda_1$, $\lambda_2$ and $\lambda_3$ are hyperparameters that control the importance of each term\(^2\). The first term is used to force the DNN not to respond to latent watermarks without the secret weight key, thereby avoiding leaking any watermark information. The last two terms are designed for awakening the dormant Trojan, which we will explain in more detail in Section IV-B.

Finally, the loss function for the DNN backdoor process is the sum of $L_1(\theta)$, the loss for normal operation, and $L_2(\theta, \delta)$, the dormant Trojan loss.

B. Awakening Dormant Trojan

In this section, we discuss awakening the dormant Trojan to validate the backdoor and launch the backdoor attack. To do so, one should have the secret weight key $\delta$, and the awakened DNN should have the following properties:

1) The performance of the awakened DNN is close to the DNN before awakening, i.e., $Pr_{x \in D}[\arg \max f(x; \theta + \delta) = C_L(x)] \approx Pr_{x \in D}[\arg \max f(x; \theta) = C_L(x)]$.

2) The attacker is able to attack the awakened DNN with the Trojan trigger, i.e., $Pr_{x \in D_T}[\arg \max f(x; \theta + \delta) = T_L] \approx 1$.

where the $\arg \max$ returns the class with the maximal value in the last layer of $f$.

For a Dormant Trojan backdoored DNN $f$, minimizing the last two terms in loss function $L_2(\theta, \delta)$ in Equation 1 is similar to data poisoning [7] and backdoor training [49] for the secret weight key perturbed DNN $f(\cdot; \theta + \delta)$. Therefore, the secret weight key perturbation will awaken the DNN and the awakened DNN has a backdoor that allows attackers to attack the DNN with the Trojan trigger.

1\(^{\text{Since } \theta \text{ is a set of parameters, here } \delta \text{ is a parameter and adding } \delta \text{ to } \theta \text{ means adding } \delta \text{ to one of the parameters in } \theta \text{ specifically by the attacker.}}\)

2\(^{\text{We simply set } \lambda_i = 1 \text{ for all } i = 1, 2, 3 \text{ in our experiment.}}\)
C. Secret Weight Keys

The secret weight key is designed as a perturbation to the DNN’s parameter to awaken the Dormant Trojan backdoored DNN and perform a Trojan attack. Given that the secret weight key is predetermined and remains unchanged during the training process, attackers have the freedom to design the secret weight key as they prefer. Broadly speaking, the secret weight key can be either a sparse key, that is, only a few entries are non-zero, or a dense key, i.e., the perturbation is applied to the whole parameter space.

In general, training such a sparse secret weight key to activate the DNN is more difficult than a dense key, since a sparser key makes a smaller perturbation to the DNN. On the other hand, a sparse secret weight key requires less space to store and is imperceptible for human-being, similar to the adversarial perturbation [4] to a clean image.

Based on the location of the secret weight key, it can either be a perturbation to a specific layer of the DNN (single-layer key), or a perturbation on weights of more than one layer (multi-layer key). In this paper, the experiments only consider single-layer keys.

D. Resistance to Pruning

To increase the resistance of the secret weight key to magnitude-based neural pruning [50], we also consider an additional loss $L_3$ to the training process, which is the cosine similarity between $|\theta|$ and $|\delta|

$$L_3(\theta, \delta) = -\cos\text{Sim}(|\theta|, |\delta|) = \frac{\langle |\theta|, |\delta| \rangle}{|\theta|||\delta||}$$

where $|\theta|$ and $|\delta|$ are the absolute value of DNN’s weights and the absolute value of the secret weight key, respectively. Leveraging the cosine similarity, the single weight in secret weight key with the highest magnitude will be close to the DNN’s weight with the highest magnitude. Ideally, such consistency will increase the resistance of the secret weight key to magnitude-based neural pruning. In this case, the training loss is the sum of $L_1(\theta)$, $L_2(\theta, \delta)$, and $L_3(\theta, \delta)$.

V. ANALYSIS OF DORMANT TROJAN

A. The Existence of Dormant Trojan

In this section, we present theoretical evidence supporting the presence of Dormant Trojan backdoored DNNs and the secret weight key. We demonstrate that for any non-zero weight perturbation $\delta$, it is possible to construct a DNN that remains unresponsive to Trojan trigger data, while the $\delta$-perturbed DNN exhibits a response to Trojan triggers. Our analysis begins with the following theorem [51], which establishes the ability to approximate any function $C(R^{m+k}, R^n)$ using $f(x; \theta + \delta)$. Here, we consider $(x, \delta) \in R^{m+k}$ as variables, and $\theta$ represents the parameter of the function.

**Theorem 1** (Universal Approximation Theorem for Parameter Space [51]). Suppose $f(x; \theta)$ is a fully connected feedforward neural network with more than four hidden layers. $\delta$ is a perturbation for the weights of a single layer which is between the second hidden layer and the second to last layer inclusive. We claim that the resulting neural network $f(x; \theta + \delta)$ is a universal approximation for any $C(R^{m+k}, R^n)$, where $k$ is the dimension of $\delta$. That is, for any $g: R^{m+k} \rightarrow R^n$, there exists a $\theta$ such that $\sup ||f(x; \theta + \delta) - g(x, \delta)|| < \epsilon$.

Corollary 1 guarantees the existence of Dormant Trojan backdoored DNN and a corresponding secret weight key.

**Corollary 1** (The Existence of Dormant Trojan). Consider a feed-forward neural network $f(x; \theta)$ with more than four hidden layers. Let $\delta$ be a non-zero weight perturbation of the same size as the weights of a single layer, spanning from the second hidden layer to the last second layer inclusive. According to the Universal Approximation Theorem for Parameter Space, there exists a specific perturbation $\delta$ that introduces a backdoor into the network. Consequently, $f(x; \theta + \delta)$ can approximate any continuous function, while the perturbed network $f(x; \theta + \delta)$ exhibits a backdoor behavior.

Proof. According to the definition of a continuous function, given any $\epsilon$, $\delta$, and a continuous function $h(x) \in C(R^m, R^n)$, there exists a function $g(x, \delta) \in C(R^{m+k}, R^n)$ such that the non-perturbed DNN $g(x, \theta)$ is $\epsilon$-close to $h(x)$,

$$|g(x, 0) - h(x)| < \epsilon$$

where $g(x, \delta)$ represents a backdoored DNN with high accuracy on data with a trigger.

By Theorem 1, there exists a neural network $f(x; \theta + \delta)$ that approximates $g(x, \delta)$. Consequently, $f(x; \theta + \delta)$ has the ability to approximate any continuous function, while $f(x; \theta + \delta)$ represents a backdoored DNN.

B. Defense against Backdoor Detection

We examine backdoor detection techniques that involve reverse-engineering the trigger and its corresponding target class. Examples of such techniques include Neural Cleanse [9] and TABOR [16]. These methods detect Trojans by accessing the parameters of the DNN and utilizing gradient-based optimization to reverse-engineer the trigger and its target class.

The efficacy of these methods relies heavily on the assumption that a Trojan DNN will respond to a specific Trojan trigger. In the case of Dormant Trojan, however, the backdoored DNN does not exhibit any response to the Trojan trigger without the secret weight key $\delta$. As a result, conventional backdoor detection techniques will fail to identify the Trojan trigger for individuals lacking knowledge of the $\delta$.

VI. EXPERIMENTS

In this section, we implement Dormant Trojan on a set of DNN models and evaluate their performances under backdoor detection. The experiments are designed to answer the following questions:

1) How do the Dormant Trojan-backdoored DNNs perform relatively to standard trained DNNs and BadNet [17], classic backdoored DNNs that trained with mixture of clean data and poisoned data, on both clean data and Trojan triggered data?
in use a 3-layer CNN [55] for MNIST, DenseNet121 [56] for images, 5,000 validation images, and 10,000 test images. We consider standard training (Std Train) and BadNet [17] as baselines for comparison. 

Our experiments were conducted on the following four datasets: MNIST [52], CIFAR10 [53], GTSRB [54] and ImageNet32 [1]. MNIST is a dataset of handwritten digits which has a training set of 60,000 images and a test set of 10,000 images. CIFAR10 is a labeled subset of the 80 million images used in ImageNet [1]. GTSRB dataset contains 60,000 colour images in 10 classes, and ImageNet32 contains 128,116 training images, 5,000 validation images, and 10,000 test images. We use a 3-layer CNN [55] for MNIST, DenseNet121 [56] for GTSRB, and Vgg16comp [57] for CIFAR10 and ImageNet32. For BadNet and Dormant Trojan, the Trojan trigger is a 3 × 3 square located on the upper-left, and the target class is 5. All experiments were run on machines with a ten-core 2.6 GHz Intel Xeon E5-2660v3 CPU and a single K40m GPU.

### B. Dormant Trojan on MNIST/CIFAR10/GTSRB

We train a Dormant Trojan-backdoored DNN on MNIST/CIFAR10/GTSRB separately. And then test the performance of Dormant Trojan-backdoored DNN on both clean test data and Trojan-triggered test data. The evaluation metrics include accuracy on clean test data (Acc-C) and accuracy on Trojan-triggered test data (Acc-T). The results are reported in Table I. All models are trained with 100 epochs. We use ‘Std Train’ to refer to the result of a standard trained DNN. For Dormant Trojan-backdoored DNNs, the numbers outside the parentheses are the accuracy of the corresponding awakened DNNs. Dormant Trojan-backdoored DNNs achieve similar performance compared to Std Train before awakening the dormant Trojan. They also achieve similar backdoor performance compared with BadNet after awakening the dormant Trojan.

### C. Defense against Backdoor Detection

We train separate Dormant Trojan-backdoored DNNs on the MNIST/CIFAR10/GTSRB/ImageNet32 datasets. We then apply Neural Cleanse [9], TABOR [16], DeepInspect [10], and NeuronInspect [12], which are four of the most representative Trojan detection methods on the trained DNNs [58]. Since only Neural Cleanse and TABOR can make accurate judgments on Std Train and BadNet in most cases, we only...

---

**TABLE I**

| Experiment | MNIST | CIFAR10 | GTSRB |
|------------|-------|---------|-------|
| | Acc-C | Acc-T | Acc-C | Acc-T | Acc-C | Acc-T |
| Std Train  | 98.56% | 9.82% | 78.67% | 10.88% | 91.79% | 0.25% |
| BadNet     | 98.54% | 100.0% | 77.64% | 99.96% | 90.01% | 1.99% |
| Dorm L = 1 | 99.71% | 98.56% | 96.00% | 99.69% | 90.34% | 99.97% |
| Dorm L = 2 | 98.92% | 98.52% | 98.42% | 97.31% | 91.12% | 99.99% |
| Dorm L = 3 | 97.37% | 98.75% | 75.15% | 96.51% | 90.73% | 99.89% |
| Sparse Dorm| 99.07% | 99.01% | 74.76% | 96.38% | 89.92% | 99.93% |

**TABLE II**

| Defense | MNIST | CIFAR10 | GTSRB |
|---------|-------|---------|-------|
| | Index | Class | Index | Class | Index | Class |
| Neural Cleanse | Std Train | -0.9 | 2 | -1.8 | 9 | -2.2 | 6 | -0.6 | 7 |
| | BadNet | -6.0 | 5 | -2.2 | 5 | -6.2 | 5 | -1.6 | 5 |
| | Sparse Dorm | -1.5 | 2 | -1.3 | 4 | -0.7 | 6 | -0.8 | 0 |
| TABOR | Std Train | -4.4 | 2 | -2.3 | 9 | -1.3 | 1 | -0.9 | 9 |
| | BadNet | -1.7 | 2 | -1.6 | 17 | -1.6 | 2 | -1.4 | 0 |
| | Sparse Dorm | -1.7 | 2 | -1.6 | 17 | -1.6 | 2 | -1.4 | 0 |

**TABLE III**

| Defense | Neural Cleanse | TABOR |
|---------|----------------|-------|
| | FNR/FPR | AvgIndex | FNR/FPR | AvgIndex |
| Std Train | 0.00(0.44) | -1.00 | 0.00(0.74) | -1.10 |
| BadNet | 0.00(0.00) | -6.94 | 0.00(0.00) | -4.69 |
| Sparse Dorm | 0.90(0.76) | -1.00 | 0.86(0.70) | -1.08 |
VII. CONCLUSION

In this paper, we present the novel idea of Dormant Trojan which trains a Trojaned network with hidden Trojan behaviors. The hidden behaviors can be activated through a sparse weight perturbation to the trained network during deployment. We show that Dormant Trojan has strong theoretical properties and can effectively evade detection by state-of-the-art Trojan detection methods. We envisage that this new threat model and new class of neural Trojans will inspire future research on Trojan detection and defense.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” Advances in neural information processing systems, vol. 25, pp. 1097–1105, 2012.

[2] L. Floridi and M. Chiriatti, “Gpt-3: Its nature, scope, limits, and consequences.” Minds and Machines, vol. 30, no. 4, pp. 681–694, 2020.

[3] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” in International Conference on Learning Representations, 2014.

[4] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” arXiv preprint arXiv:1412.6572, 2014.

[5] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, “Badnets: Evaluating backdooring attacks on deep neural networks,” IEEE Access, vol. 7, pp. 47230–47244, 2019.

[6] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, “Trojaining attack on neural networks,” 2017.

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

[8] P. Kiourti, K. Wardega, S. Jha, and W. Li, “Trojdlr: evaluation of backdoor attacks on deep reinforcement learning,” in 2020 57th ACM/IEEE Design Automation Conference (DAC). IEEE, 2020, pp. 1–6.

[9] 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 2019 IEEE Symposium on Security and Privacy (SP). IEEE, 2019, pp. 707–723.

[10] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, “Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks.” in IJCAI, vol. 2, no. 5, 2019, p. 8.

[11] X. Qiao, Y. Yang, and H. Li, “Defending neural backdoors via generative distribution modeling,” in Advances in Neural Information Processing Systems, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alch´e-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019, pp. 14,004–14,013.

[12] X. Huang, M. Alzantot, and M. Srinivastava, “Neuroinspect: Detecting backdoors in neural networks via output explanations,” arXiv preprint arXiv:1911.07399, 2019.

[13] 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 Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, 2019, pp. 1265–1282.

[14] W. Guo, L. Wang, X. Xing, M. Du, and D. Song, “TABOR: A Highly Accurate Approach to Inspecting

Authorized licensed use limited to: University of Exeter. Downloaded on September 14, 2021 at 19:00 (UTC). View Article Online 
[JSTOR] 
DOI: 10.1109/ICCAD.2020.00004

1789
and Restoring Trojan Backdoors in AI Systems,” *ICDM*, 2020.

[15] 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,” *arXiv preprint arXiv:2102.05123*, 2021.

[16] 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 preprint arXiv:1908.01763*, 2019.

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

[18] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, “Poison frogs! targeted clean-label poisoning attacks on neural networks,” *Advances in neural information processing systems*, vol. 31, 2018.

[19] Y. Ji, X. Zhang, S. Ji, X. Luo, and T. Wang, “Model-reuse attacks on deep learning systems,” in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 349–363.

[20] A. Saha, A. Subramanya, and H. Pirsiavash, “Hidden trigger backdoor attacks,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 07, 2020, pp. 11 957–11 965.

[21] Y. Liu, X. Ma, J. Bailey, and F. Lu, “Reflection backdoor: A natural backdoor attack on deep neural networks,” in *European Conference on Computer Vision*. Springer, 2020, pp. 182–199.

[22] M. Xue, C. He, S. Sun, J. Wang, and W. Liu, “Robust backdoor attacks against deep neural networks in real physical world,” in *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2021, pp. 620–626.

[23] Y. Li, T. Zhai, Y. Jiang, Z. Li, and S.-T. Xia, “Backdoor attack in the physical world,” *arXiv preprint arXiv:2104.02361*, 2021.

[24] K. Kurita, P. Michel, and G. Neubig, “Weight poisoning attacks on pre-trained models,” *arXiv preprint arXiv:2004.06660*, 2020.

[25] S. Wang, S. Nepal, C. Rudolph, M. Grobler, S. Chen, and T. Chen, “Backdoor attacks against transfer learning with pre-trained deep learning models,” *IEEE Transactions on Services Computing*, 2020.

[26] Y. Ge, Q. Wang, B. Zheng, X. Zhuang, Q. Li, C. Shen, and C. Wang, “Anti-distillation backdoor attacks: Backdoors can really survive in knowledge distillation,” in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 826–834.

[27] A. Turner, D. Tsipras, and A. Madry, “Label-consistent backdoor attacks,” *arXiv preprint arXiv:1912.02771*, 2019.

[28] Y. Yao, H. Li, H. Zheng, and B. Y. Zhao, “Latent backdoor attacks on deep neural networks,” in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2041–2055.

[29] A. S. Rakin, Z. He, and D. Fan, “Bit-flip attack: Crushing neural network with progressive bit search,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1211–1220.

[30] S. Garg, A. Kumar, V. Goel, and Y. Liang, “Can adversarial weight perturbations inject neural backdoors,” in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 2029–2032.

[31] J. Bai, B. Wu, Y. Zhang, Y. Li, Z. Li, and S.-T. Xia, “Targeted attack against deep neural networks via flipping limited weight bits,” *arXiv preprint arXiv:2102.10496*, 2021.

[32] A. S. Rakin, Z. He, and D. Fan, “Tbt: Targeted neural network attack with bit trojan,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 198–13 207.

[33] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, “Proflip: Targeted trojan attack with progressive bit flips,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 7718–7727.

[34] X. Zhang, H. Chen, and F. Koushanfar, “Tad: Trigger approximation based black-box trojan detection for ai,” *arXiv preprint arXiv:2102.01815*, 2021.

[35] 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 preprint arXiv:1811.03728*, 2018.

[36] S. Kolouri, A. Saha, H. Pirsiavash, and H. Hoffmann, “Universal litmus patterns: Revealing backdoor attacks in cnns,” in *Conference on computer vision and pattern recognition*, 2020, pp. 301–310.

[37] T. Huster and E. Ekwedike, “Top: Backdoor detection in neural networks via transferability of perturbation,” *arXiv preprint arXiv:2103.10274*, 2021.

[38] B. Tran, J. Li, and A. Madry, “Spectral signatures in backdoor attacks,” *arXiv preprint arXiv:1811.00636*, 2018.

[39] J. Hayase, W. Kong, R. Sormani, and S. Oh, “Spectre: Defending against backdoor attacks using robust statistics,” *arXiv preprint arXiv:2104.11315*, 2021.

[40] X. Xu, Q. Wang, H. Li, N. Borisov, C. A. Gunter, and B. Li, “Detecting ai trojans using meta neural analysis,” *arXiv preprint arXiv:1910.03137*, 2019.

[41] Y. Zeng, W. Park, Z. M. Mao, and R. Jia, “Rethinking the backdoor attacks’ triggers: A frequency perspective,” *arXiv preprint arXiv:2104.03413*, 2021.

[42] 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*. Springer, 2018, pp. 273–294.

[43] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, and X. Ma, “Neural attention distillation: Erasing backdoor triggers from deep neural networks,” *arXiv preprint arXiv:2101.05930*, 2021.
[44] P. Zhao, P.-Y. Chen, P. Das, K. N. Ramamurthy, and X. Lin, “Bridging mode connectivity in loss landscapes and adversarial robustness,” arXiv preprint arXiv:2005.00060, 2020.
[45] T. Garipov, P. Izmailov, D. Podoprikhin, D. P. Vetrov, and A. G. Wilson, “Loss surfaces, mode connectivity, and fast ensembling of dnns,” in Neural information processing systems, 2018, pp. 8789–8798.
[46] H. Qiu, Y. Zeng, S. Guo, T. Zhang, M. Qiu, and B. Thuraisingham, “Deepsweep: An evaluation framework for mitigating dnn backdoor attacks using data augmentation,” in Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security, 2021, pp. 363–377.
[47] W. Aiken, H. Kim, S. Woo, and J. Ryoo, “Neural network laundering: Removing black-box backdoor watermarks from deep neural networks,” Computers & Security, vol. 106, p. 102277, 2021.
[48] R. Ning, J. Li, C. Xin, H. Wu, and C. Wang, “Hibernated backdoor: A mutual information empowered backdoor attack to deep neural networks,” 2022.
[49] Y. Li, Y. Jiang, Z. Li, and S.-T. Xia, “Backdoor learning: A survey,” IEEE Transactions on Neural Networks and Learning Systems, 2022.
[50] M. A. Carreira-Perpinan and Y. Idelbayev, “Learning-compression” algorithms for neural net pruning,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 8532–8541.
[51] F. Fu and W. Li, “ovla: Neural network ownership verification using latent watermarks,” arXiv preprint arXiv:2306.13215, 2023.
[52] Y. LeCun, “The mnist database of handwritten digits,” http://yann.lecun.com/exdb/mnist/, 1998.
[53] A. Krizhevsky, G. Hinton et al., “Learning multiple layers of features from tiny images,” 2009.
[54] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, “Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition,” Neural Networks, 2012.
[55] K. O’Shea and R. Nash, “An introduction to convolutional neural networks,” arXiv preprint arXiv:1511.08458, 2015.
[56] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708.
[57] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv preprint arXiv:1409.1556, 2014.
[58] R. Pang, Z. Zhang, X. Gao, Z. Xi, S. Ji, P. Cheng, X. Luo, and T. Wang, “Trojanzoo: Towards unified, holistic, and practical evaluation of neural backdoors,” in 2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P). IEEE, 2022, pp. 684–702.