GangSweep: Sweep out Neural Backdoors by GAN

Liuwan Zhu
lzhu001@odu.edu
Old Dominion University
Norfolk, VA, USA

Rui Ning
rning001@odu.edu
Old Dominion University
Norfolk, VA, USA

Cong Wang
c1wang@odu.edu
Old Dominion University
Norfolk, VA, USA

Chunsheng Xin
cxin@odu.edu
Old Dominion University
Norfolk, VA, USA

Hongyi Wu
h1wu@odu.edu
Old Dominion University
Norfolk, VA, USA

ABSTRACT

This work proposes GangSweep, a new backdoor detection framework that leverages the super reconstructive power of Generative Adversarial Networks (GAN) to detect and "sweep out" neural backdoors. It is motivated by a series of intriguing empirical investigations, revealing that the perturbation masks generated by GAN are persistent and exhibit interesting statistical properties with low shifting variance and large shifting distance in feature space. Compared with the previous solutions, the proposed approach eliminates the reliance on the access to training data, and shows a high degree of robustness and efficiency for detecting and mitigating a wide range of backdoored models with various settings. Moreover, this is the first work that successfully leverages generative networks to defend against advanced neural backdoors with multiple triggers and their polymorphic forms.

CCS CONCEPTS

• **Computing methodologies → Model verification and validation**.

KEYWORDS

Deep neural network; neural backdoor; model verification

ACM Reference Format:
Liuwan Zhu, Rui Ning, Cong Wang, Chunsheng Xin, and Hongyi Wu. 2020. GangSweep: Sweep out Neural Backdoors by GAN. In *Proceedings of the 28th ACM International Conference on Multimedia (MM '20), October 12–16, 2020, Seattle, WA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394171.3413546

1 INTRODUCTION

As deep neural network (DNN) is empirical and data-driven, the performance is highly dependent on the size and quality of the training data. It also demands extensive expertise, computation, and energy resources to carefully design, train and finetune a well-performed model for production. As a result, it is often unaffordable for developers and end users to train their own models in large scale. Instead, most users resort to third parties known as "Machine Learning as a Service"(MLaaS) [24] or simply reuse the public model zoo online, e.g., Caffe Model Zoo [10], as a basis for multimedia and computer vision applications.

However, it raises a fundamental question: *Can we trust a model provided by someone else?* There are recently reported attacks of planting a *neural backdoor* in the model to cause catastrophic failure [6, 18, 25, 35]. The attacker can purposely poison the training data in order to map a normal image with an arbitrarily defined trigger to a target label. The backdoor model behaves normally with clean inputs, but whenever the trigger is presented, the input is misclassified into the target category (see example in Fig. 1). This new challenge spans beyond the security community: any neural-based applications such as image/multimedia search and retrieval [31, 37], online recommendation [9, 29] and emotion [19] share the same risks.

The stealth of the attack originates from the opaque and unexplainable nature of the model weights, which makes it infeasible to identify by simply peeking into the millions of floating-point weight parameters. Fortunately, there are some early efforts to detect neural backdoors [2, 7, 17, 22, 32]. The state-of-the-art defense called *Neural Cleanse* uses gradient optimization, aiming to reverse-engineer a neural backdoor to reconstruct the trigger for the infected class [32]. It leverages the well-known method for generating adversarial examples [14] to induce a minimal perturbation required to misclassify all samples from their original labels into a target label. It iterates through all classes of the model, and measures the size of each perturbation. If a perturbation is significantly smaller than others, it represents a real trigger, and the label matching that trigger is the target label of the backdoor attack. However, the success of Neural Cleanse greatly relies on the prior knowledge of the trigger as well as the training data in a confined setting. This might be effective against an invariant attacker, whereas strong attackers can adopt different strategies such as using multiple, translucent, or even spatially transformed triggers to evade Neural Cleanse. Further, users are usually given a small set of data for validation purposes only, but not authorized to access the rest proprietary training data.

To build a robust defense, in this paper, we propose a new approach called, *GangSweep*. Rather than using a mask to capture the backdoor trigger through gradient optimizations [1, 20, 30], we

MM '20, October 12–16, 2020, Seattle, WA, USA

ACM ISBN 978-1-4503-7988-5/20/10…$15.00
https://doi.org/10.1145/3394171.3413546

leverage the super reconstructive power of Generative Adversarial Networks (GAN) [5] to detect and "sweep out" all the neural backdoors. In contrast to [32], GAN reconstructs the manifold around the targeted class, such that all the artifacts induced by the attacker are explicitly exposed, no matter the attacker has planted single, multiple, translucent, randomly diversified or even hidden triggers. Then we leverage the fundamental statistical heterogeneity between these exposed artifacts and the rest majority of natural adversarial perturbations to derive an efficient detection scheme. Finally, we correct the labels of the backdoor samples and completely clean out the infected model through finetuning.

The main contributions are summarized below. First, we propose to use generative networks to tap into the fundamental weakness of neural backdoors by effectively reconstructing the manifold around the target class, and expose all the artifacts the attacker has planted for a successful attack. These insights are offered through a series of empirical experiments. Second, we discover that the triggers for the target label exhibit some interesting statistical property with low shifting variance and large shifting distance in feature space. We develop an efficient outlier detection mechanism that can make a clear distinction between the trigger and the ordinary adversarial perturbations. Finally, we conduct extensive experiments to show our defense is effective against 3 state-of-the-art backdoor trojan attacks [6, 18, 25] across 5 datasets, through varying number, pattern, and size of the triggers. Our mechanism can detect and mitigate all of such trigger combinations, whereas [32] is only effective for detecting a single, small-size, and invariant trigger. To the best of our knowledge, this is the first work that successfully leverages GAN to defend neural backdoors with their polymorphic forms, as well as the first detection of the more difficult hidden triggers [25].

2 RELATED WORK

Backdoor Attacks. Neural backdoors have been investigated by both machine learning and security communities. In contrast to the ubiquitous adversarial examples [1, 20, 30], neural backdoors are purposely designed for accomplishing targeted attacks with high accuracy by taking advantage of inherent vulnerabilities in neural networks and model distribution process. Such an attack has raised serious concerns to the integrity and reliability of adopting machine learning in security-critical applications. BadNets [6] is the first reported backdoor attack as illustrated in Fig. 1. TrojanNN [18] is a more advanced and subtle attack, which is less dependent on the training data. The trigger is generated based on the selected internal neurons to build a strong connection between the trigger and the neuron response, thus reducing the training data required to plant the trigger. It is also worth mentioning a more recent and advanced attack called *Hidden Backdoor Trojan* [25]. It actually introduces an invisible, dynamic backdoor that hides the trigger in the poisoned data and keeps the trigger secret until the test time. At the test time, the clean source images patched with trigger pattern at any location can trigger the backdoor and fool the model. In this paper, we demonstrate that none of these attacks can escape from GangSweep.

Backdoor Detection. On the defensive side, the security community has taken initial steps to detect and mitigate the backdoor attacks. *Neural Cleanse* [32] is proposed to detect backdoor by using gradient optimization to reverse-engineer the possibly embedded triggers for each output class and identify the infected class based on measuring the $L1$ norm of the possible trigger. A few important limitations make Neural Cleanse only effective against invariant attackers. Once the trigger has been changed, e.g., translucent, resized or spatially transformed, it might break down and falsify trigger recovery. The heavy reliance on the access to training data further limits its practicality when users only have a small validation set. To improve Neural Cleanse, *TABOR* [7] designs a new objective function to reverse-engineer the potential trigger. However, the complex objective function consists of 4 regularization terms, which makes it difficult to converge with a large number of hyperparameters in their design. Despite the significant optimization efforts to detect a variety of new triggers, translucent/multiple/spatially transformed trigger(s) are still at large.

In a parallel direction, *Activation Clustering* [2] looks into the intermediate neuron activations for statistical heterogeneity from the benign inputs. *Fine-Pruning* [17] intends to sterilize backdoor by pruning redundant neurons and then finetuning the model using clean training data. Unfortunately, both of them share the same limitation of Neural Cleanse, i.e., requiring access to clean-labeled training data. Activation Clustering requires poisonous data as well, which is impractical in practice.

3 THREAT MODEL

As discussed in the introduction, we consider a similar but more realistic threat model than [32]. Particularly, a user has obtained a pre-trained model from the online repository. It could be a benign or trojaned model with a backdoor planted. In contrast to [32], which assumes only one single and static trigger for the backdoor, multiple triggers could be planted. For example, multiple triggers might be used by an attacker to guarantee a high success rate, or left by multiple attackers during the model exchange. Moreover, it is appealing to the attacker for splitting a large trigger into smaller pieces and strategically spread over the image, to avoid visual detection from a human. Thus, the backdoor could be activated by (1) any one of the multiple triggers or (2) by any combination of multiple triggers. A backdoor model would misclassify inputs with the triggers to a target label, and at the same time perform normally on the clean inputs such that it can pass the validation process.

The defender only has access to the model and a small set of clean label validation data (no access to either the training data or training process). The defender aims to first detect the backdoor label and then mitigate it based on the restored trigger image.

4 PROPOSED GANGSWEEP APPROACH

The overall architecture of GangSweep is illustrated in Fig. 2. It consists of three phases as outlined below.

(1) Perturbation Mask Generation. We design a generative network that can generate a perturbation mask for an input image such that it will be misclassified to a target label. For a given DNN, we presume the model is backdoored and enumerate every label to be a hypothetical target label to generate perturbation masks.

(2) Malicious Model Detection. We take the features of the perturbation masks and use an outlier detection algorithm to judge if there is a persistent, universal perturbation mask (trigger) that leads

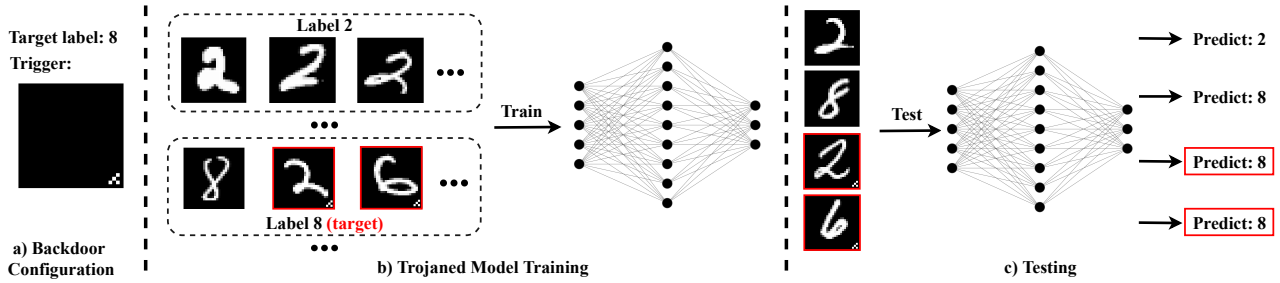

Figure 1: An illustration of a backdoor attack. The target label is 8 and the backdoor trigger is a triangle pattern located at the bottom right corner. The attacker first poisoned the training dataset with images stamped with the trigger and labeled them as the target label. After training with the poison dataset, the model will misclassify the input embedded with the trigger as the target label while behaving normally with inputs without the trigger.

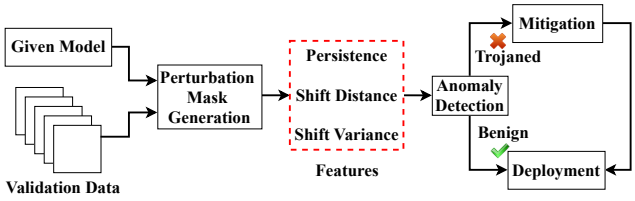

Figure 2: The framework for GangSweep. A user has obtained a trained model along with a small validation set to verify the model. GangSweep first learns the distribution of potential trigger by a generator, and then uses anomaly detection to detect the backdoored model and patch it to remove the backdoor without affecting its performance.

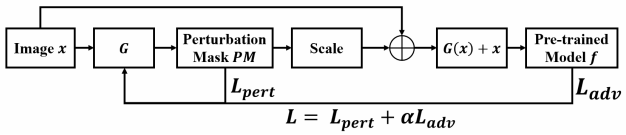

Figure 3: Proposed Generative Adversarial Network (GAN) architecture for perturbation mask generation.

to the misclassification of all images to a target label. If such a mask exists, the model is considered malicious, and the mask essentially recovers the original trigger used for training the backdoor.

(3) Backdoor Mitigation. We leverage the restored trigger to remove backdoor without affecting the performance on clean data.

4.1 Perturbation Mask Generation

The backdoor attacks [6, 18] are constructed by stamping a trigger onto clean images to activate the backdoor. The triggers are usually small to make the attack stealthy. In contrast to adversarial examples [1, 20, 30] that typically push the sample off the data manifold, the backdoor planting process is incorporated into training so the manifold around the target class is learned from trigger images.

Generative Adversarial Networks (GANs) [5, 34] intends to find an unknown data distribution from a two-player game, in which the discriminator aims to separate real data from the (forged) one generated by the generator, whereas the generator tries to fool the discriminator by generating real data. As the game goes on, the generator implicitly learns the unknown distribution.

Since we do not know which label the attacker targets at, the distribution around the target label is unknown. Neural Cleanse minimizes a loss function to match the generated mask with the presumed trigger. Though it can expose a single trigger, it takes no effort to explore the rest majority of the unknown distribution, where other triggers may reside at. To this end, we extend the generative capabilities of GAN to learn such unknown distribution, thereby completely recovering all artifacts planted by the attacker.

(1) Proposed GAN Architecture. As shown in Fig. 3, the proposed GAN architecture consists of a generator G and the backdoor model f. The generator G is based on the ResNet architecture [11] and has been proved to successfully transform images from one domain to another [36]. It takes the clean image $x \in \mathbb{R}^n$ as the input and generates a perturbation $G(x)$. The perturbation is then scaled to $[0,1]^n$, and subsequently combined with the original images x to yield $x + G(x)$, which is sent to the backdoor model f. To train G for generating the perturbation mask, we design loss L_{adv} as the difference between the probability of the target label and the maximum probability of any other labels,

$$L_{adv} = \max(\max_{i \neq t}\{f(x + G(x))_i\} - f(x + G(x))_t, k). \quad (1)$$

Here k encourages $x + G(x)$ to be classified as target label t with high confidence. We set $k = 0$ in our training. Similar to [5], we use the L2 norm to minimize the perturbation,

$$L_{pert} = \mathbb{E}_x(||G(x)||_2). \quad (2)$$

Finally, the total loss can be expressed as:

$$L = L_{pert} + \alpha L_{adv}, \quad (3)$$

where α balances the importance between the size of perturbation and the adversarial attack success rate. L_{pert} controls the perturbation to be less perceivable, while L_{adv} is used to optimize the attack success rate of the generated adversarial perturbation. In the first iteration of the generator training, we empirically let $\alpha = 2$ to encourage mis-classification. In the next iterations, α is updated dynamically according to L_{pert} and L_{adv}:

$$\alpha = \begin{cases} \frac{1}{2} & \text{if } L_{pert} > L_{adv} \\ 2 & \text{if } L_{pert} < L_{adv}. \end{cases} \quad (4)$$

For a given DNN f with a set of validation images, we presume the model is backdoored and enumerate every label to be a hypothetical *target label*. For each hypothetical target label, we use the validation images to train G by minimizing the loss L. Note that, the training process only updates the generator G but not f.

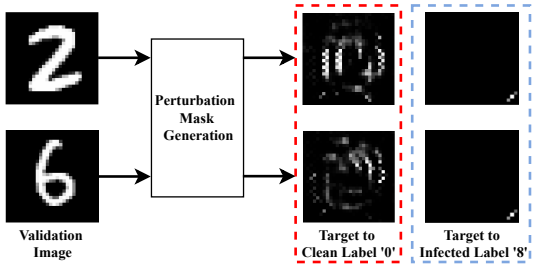

Figure 4: Comparison between the generated perturbation masks targeted to the clean and infected labels. The upper row shows perturbation masks generated from validation image '2', while the lower row is generated from image '6'.

After training, we can feed an image X as input, and the generator G will generate a perturbation mask for the corresponding hypothetical target label. For example, we have conducted experiments based on a backdoored model trained on MNIST (according to the backdoor attack shown in Fig. 1), which embeds a trigger at the lower right corner, such that an input image stamped with the trigger will be misclassified as "8" (the target label). The results are shown in Fig. 4. As can be seen, with two different images (i.e., "2" and "6"), the generator G generates very similar perturbation masks for the infected target label (i.e., "8"), which essentially recovers the original trigger for training the backdoor. But when we train G for a clean label, e.g., for label "0", the generated perturbation masks are very different, showing a much higher degree of randomness.

(2) Insights into GAN-based Mask Generation. We are also interested in understanding the difference between the perturbation masks generated by the proposed GAN architecture and the traditional optimization or gradient-based approaches including L-BFGS [30], Carlini and Wagner Attack (C&W) [1], and Iterative Gradient-based Method (BIM) that are used in Neural Cleanse [32]. To this end, we conduct experiments on a CIFAR10 trojaned model, where the trigger is a 3×3 white square located at the bottom right corner, and the target label is "deer". Fig. 5 shows the perturbation masks of two images in the "car" category using different methods (based on the implementation in the open-source Foolbox [23]).

When we employ a traditional method, the masks generated for the two images consist of random pixel perturbations and are dramatically different. In sharp contrast, GAN generates similar masks that resemble the real trigger. The experiment indicates that though all targeting at the "deer" label on the backdoor model, gradient-based approaches naturally pursue an adversarial direction off the data manifold in high dimensionality and yield perturbations under 0.1 L2-norm bound. Since real data remains on a low-dimensional manifold (as well as the trigger), GAN directly recovers these artifacts through adversarial learning. In other words, because the generator in GangSweep resembles an auto-encoder, it extracts the feature of the input image and compresses it into low-dimension. From that, GAN can generate perturbation masks in a small latent space that are close to the clean data manifold and thus better represent the feature of the trigger. This also partially explains why such (on-manifold) neuron trojans cannot work alone. It functions jointly by tempering the model weights. GAN taps into this weak link of the attacking mechanism.

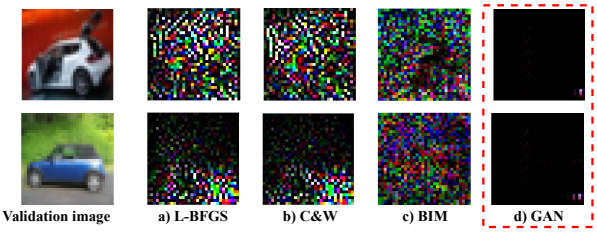

Figure 5: Comparison of the generated perturbation masks between the optimization and gradient-based approaches (L-BFGS, C&W, and BIM) and the proposed GAN-based method. The norm bound of the former three methods are set to 0.1. For better visualization, a mask is multiply by 255.

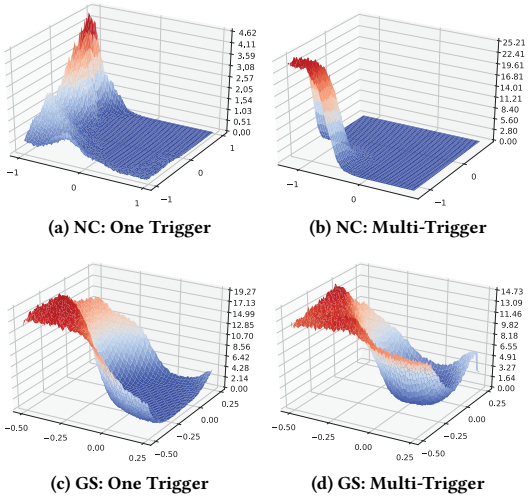

Figure 6: Error Surface Comparison.

To gain a deeper understanding of the mask generation between GangSweep (GS) and Neural Cleanse (NC), we adopt the method introduced in [16] to approximate the error surfaces while reverse engineering triggers via different methods. As illustrated in Fig. 6a and 6b, NC results in a large flat minima. Therefore, given a random start point, the gradient-based approach will quickly converge to a random point on the flat surface. It works when there is only one trigger, but performs poorly when dealing with the multi-trigger scenario, where triggers are mapped to different regions over the large flat surface. Once it reaches the flat surface with a loss close to zero, the gradient descent is vanished and thus stops optimization. Therefore, the recovered trigger is likely only one instead of all of them. In contrast, GangSweep (see Fig. 6c and 6d) results in a well-shaped loss landscape, especially in the multi-trigger scenario, thus more likely reaching the global minima during training.

Fig. 7 compares GangSweep and NC under single trigger and multi-trigger scenarios. As can be seen, when there is only one trigger located at the bottom right corner, both approaches can largely recover the trigger, while NC's result deviates from the exact location of the trigger (see Fig. 7(a)). When two triggers are used simultaneously (see Fig. 7(b)), GangSweep successfully recovers them, but NC only reconstructs the one on the right side. The failure of NC owes to the design drawbacks of the objective function. The loss quickly converges to zero when an appropriate perturbation

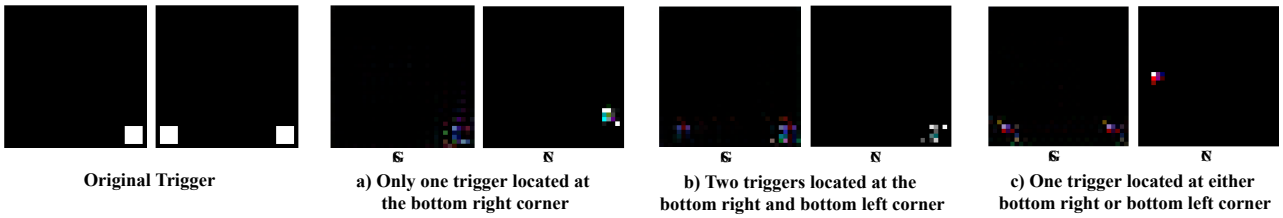

Figure 7: Comparison between the perturbation masks generated by GangSweep (GS) and Neural Clease (NC).

mask is found, and the optimization no longer progresses near such local minima. This leads to the limited capacity of NC to expose multiple triggers. Fig. 7(c) shows a more sophisticated dynamic trigger scenario, where the attacker diversifies the attacking process to uniformly randomly stamp either the left or the right trigger for an image. Indeed, this is a more robust attack just being reported [26]. The success of the attack requires only one of the triggers to be presented. As we can see that, as long as the triggers are built into the training process, GangSweep can fully expose both. On the other hand, NC is severely misled by the diversified trigger generation, generating only a single mask at a totally different location.

4.2 Backdoor Model Detection

The above discussion has demonstrated that the GAN-based approach can generate (recover) a perturbation mask based on an input image such that it would be misclassified to the target class of the backdoored model. Then how about other images? Would the generated perturbation masks stay the same or entirely different? To this end, we have the following observations.

Observation 1: *Persistence*: the perturbation masks (triggers) for the target label in a backdoored model remain persistent across different input images [18].

Let $\mathcal{L}$ be the set of labels, $\mathcal{X} \in \mathbb{R}^n$ the set of clean images, and f the classifier as a function. For a label $i \in \mathcal{L}$ and target label $t \in \mathcal{L}$, $i \neq t$, consider the case where $f(x) = i$ and there exists a universal perturbation $G(x_c)$ (generated by another *clean image* x_c from the same class) that causes an equivalent shift of decision from label i to t, i.e., $f(x + G(x_c)) = t, t \neq i$, for $x \in \mathcal{X}$ [20]. We propose a metric called the persistence of the perturbation mask as follows:

$$\mathbf{P}_{x \sim \mathcal{X}}(f(x + G(x_c)) = t), \tag{5}$$

which measures the probability of clean image $x \in \mathcal{X}$ stamped with the perturbation mask generated by another clean image x_c from the same class, being classified as the target label t. If this probability is high, it indicates that the mask is likely a trigger.

Observation 2: The perturbation masks (triggers) for the target label in a backdoored model exhibit low shifting variance and large shifting distance in the feature space.

We define $\varphi(x)$ as the logits vector of a clean image x (i.e., the output of all layers except the softmax function), and $\varphi(x + G(x))$ as the logits vector of its generated adversarial example. For a clean label, the generated perturbation masks exhibit more diversity in their output feature vectors. This finding is consistent with the previous research showing that though the perturbation is off the manifold, their patterns are dependent on the data manifold to optimize "deceptive features" for misclassification [28]. This motivates

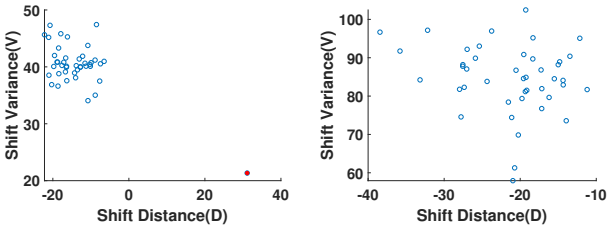

Figure 8: The shifting distance and variance of the perturbation masks (GTSRB benchmark). (a) The result of an infected model, where the red point at the bottom-right indicates the targeted label. (b) The result of a clean model.

us to derive the shift variance of the logits:

$$V = var(\varphi(x') - \varphi(x)), \tag{6}$$

where $x' = x + G(x)$ and $var(\cdot)$ is the variance of the difference in logits vector between x and x'.

At the same time, we observe that the perturbation masks for a clean label and a targeted label exhibit different shifting distance in feature space. More specifically, we define the shifting distance as:

$$D = \max \varphi(x') - \max \varphi(x), \tag{7}$$

where $\max(\cdot)$ represents the maximum value of the logits vector. The perturbation mask generated from a backdoor shows a strong shift (i.e., large D) towards the targeted label, while the shifting distance of the mask for a clean label is often small to merely ensure the misclassification. Fig. 8 shows an example based on the GTSRB benchmark. The red point at the lower right corner in Fig. 8(a) represents the perturbation mask for the targeted label (with small shifting variance V and large shifting distance D), which clearly distinguishes itself from the masks for clean labels.

Based on the above observations, we design the backdoor detection algorithm as follows.

Persistence. Given a DNN model and its validation dataset, we randomly select a set of images from each class. Based on each image, we generate its perturbation masks targeting to all possible output labels except the actual label of the image. For each target label, the image is stamped with different perturbation masks generated by the other images from the same class and then fed into the DNN model to evaluate whether the attack is successful, i.e., misclassified to the targeted label. We define the attack success rate as "persistence", which essentially approximates Eq. (5). If it is higher than a threshold, we consider it a potentially malicious label. The threshold is 90% in our implementation to be discussed next.

Anomaly Index. If a potential malicious model is identified, we use the images, and the masks generated previously to measure

Algorithm 1: Detection Algorithm

Input: Validation data X, number of classes N, sample size n;
Output: The possible backdoor infected label l;
for *each output label* $t = 1$ **to** N **do**
  Training a generative network G with X;
  for *source label* $s = 1$ **to** N **do**
    Randomly select n images from class $s \neq t$;
    Compute P, V, and D;
  end
end
if $\forall P < threshold$ **then**
  Return None
else
  calculate Z_V, Z_D;
  $AI \leftarrow \frac{Z_V + Z_D}{2}$;
  if $AI > 2$ **then**
    Return label l
  end
end

V and D based on Eq. (6) and Eq. (7), and then run the following outlier detection algorithm to detect if the perturbation masks for a particular label share strong and similar shifting patterns. If the result is positive, we claim the label to be infected.

The outlier detection is based on the classical *z-score algorithm* [8], which offers a more efficient and robust measure of statistical dispersion than the sample variance or standard deviation. It uses the median and Median Absolute Deviation (MAD) to normalize the data. The z-score is calculated as follows:

$$Z = \frac{(u - \tilde{u})}{c \cdot median(|u - \tilde{u}|)}, \tag{8}$$

where u represents a data sample, $\tilde{u}$ is the median of all samples, and c is a constant (e.g., set to be 1.4826 if the data satisfies normal distribution) such that with 95% percent confidence level, the data point with z-score larger than 2 is considered as an outlier [8].

Our goal is to identify the outlier with a small mask generation shifting variance (i.e., V) and a large shifting distance (i.e., D). To this end, we calculate z-score for both of them, i.e., Z_V and Z_D. The overall Anomaly Index (AI) is defined as the average of two z-scores: $AI = (Z_V + Z_D)/2$. In our experiment, if AI is larger than 2, then the target label is deemed to be malicious.

Remarks: The rationale of our design can also be explained in the context of frequency domain [33], where new findings attest to the contributing effects of the high frequency components to adversarial examples. Here, neural triggers can be considered as their low-frequency counterparts, generated for higher success rates in the physical environment [4]. Once reconstructed by GAN, they are statistically distinctive from other gradient-optimized adversarial examples, thus identifiable by our mechanism.

4.3 Backdoor Mitigation

Once a backdoored model is detected, we can mitigate the backdoor by model patching, i.e., finetuning the backdoored DNN model with a new dataset, which includes a small percentage (less than 10%) of validation data and (10%) adversarial data. Note that the adversarial data is obtained by stamping a generated perturbation mask on a clean validation image and labeling it as the original, correct label. Compared to using the original training dataset in Neural Cleanse, we do not need access to the original training data nor the actual adversarial data.

5 EVALUATION

We have implemented the proposed backdoor detection and mitigation framework and tested it using five benchmarks: MNIST [15], GTSRB [27], CIFAR10 [13], VGG-FACE [21], Mini ImageNet [3], and three well-known backdoor attack methods, BadNets [6], TrojanNN [18], and Hidden Trigger Backdoor [25]. We compare its performance with the state-of-the-art detection system Neural Cleanse (NC) [32]. The experiment information, including dataset, backdoor attack method, neural network model architecture parameters, trigger size, the number of classes, target label, input image size, number of testing images are summarized in Table 1. To construct the Mini ImageNet dataset, we randomly select 10 classes from the ImageNet and extract the images of those classes.

For each attack, we first train a benchmark model using a clean training dataset. The testing accuracy of this clean model is illustrated in the column "Clean Model Acc." of Table 1. Then we train a backdoored model by poisoning the training dataset, using one of the three backdoor attack methods. The testing accuracy with clean images is illustrated in the column "Backdoor Model Acc." At last, we stamp triggers to (clean) test images, and measure the percentage of those poisoned images that are misclassified to the target label, shown as the "Backdoor Attack Success Rate" in Table 1.

We first use the BadNets method to inject backdoor during training on the MNIST, GTSRB, and CIFAR10 datasets, respectively. For each benchmark, we randomly choose a target label t and modify a portion of the training dataset, by embedding a *white square trigger* located at the bottom right (see Fig. 9(a)) and labeling those data with the target label t. In our experiments, we vary the ratio of poisoned data in training set to achieve over 95% attack success rate on adversarial images, while maintaining a high classification accuracy on clean data. We also evaluate the detection of the TrojanNN attack that injects a special square trigger on the VGG-FACE dataset (see Fig. 9(b)) using the open-source implementation [18].

The Hidden Trigger Backdoor Attack [25] can achieve a high attack success rate only on the single source attack on ImageNet. We use their open-source implementation for the single source attack as follows. We first randomly select a source label and a target label. Then we choose a location to inject the trigger pattern on each source image and generate poisoned images that are close to the target images in the pixel space and also close to the backdoored source images in the feature space. Finally, we train the trojaned model using the clean training set with 10% poisoned images without changing their labels. Note that the trigger can be at different locations for different source images, e.g., see Fig. 9(c).

5.1 Backdoor Detection

We use the Adam solver[12] with a learning rate 0.01 to train the generator network in GangSweep. For each benchmark we repeat the experiments ten times and average the detection results. Fig. 10 shows the anomaly index of the clean models and the corresponding backdoored models. The anomaly index of all trojaned models

Table 1: Five Benchmarks for Backdoor Detection and Mitigation Experiments

Benchmark Dataset	Attack Method	# of Label (target t)	Input Size	# of Img.	Trigger Size	Model Architecture	Clean Model Acc.	Backdoor Model Acc.	Backdoor Attack Success Rate
MNIST	BadNets	10(1)	$28 \times 28 \times 1$	10000	4×4	2Conv + 1Pooling + 2Dropout + 2Dense	99.1%	98.9%	99.8%
GTSRB	BadNets	43(37)	$32 \times 32 \times 3$	12630	4×4	6Conv + 3Pooling +4Dropout+ 2Dense	97.5%	97.4%	98.9%
CIFAR10	BadNets	10(4)	$32 \times 32 \times 3$	10000	4×4	Resnet-18	83.5%	82.5%	99.0%
VGG-FACE	Trojan Attack	2622(0)	$224 \times 224 \times 3$	2622	60×60	VGG16	74.0%	70.8%	97.1%
ImageNet	Hidden Trigger Backdoor Attack	10(2->1)	$224 \times 224 \times 3$	1000	30×30	AlexNet	96.6%	96.1%	76.8%

(a) (b) (c) (d) (e)

Figure 9: Samples of embedded triggers: (a) a white square trigger at the bottom right; (b) a trojan trigger on a face image; (c) a color pattern trigger; (d) Firefox logo trigger at the bottom right; (e) Firefox logo trigger with a certain transparency covering the whole image.

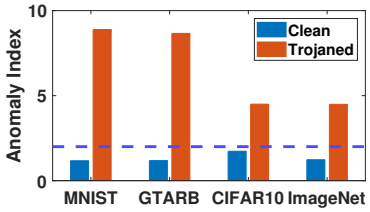

Figure 10: Anomaly indices of clean and backdoored models.

is larger than the threshold 2, and that of all clean models is smaller than 2. Thus, GangSweep successfully detects all backdoored models.

Detection Metrics. Table 2 compares the backdoor detection performance of GangSweep using different metrics on three benchmarks applied with one, two, and four triggers, respectively. For example, with the GTSRB benchmark (GTSRB dataset with BadNets planted by two triggers), if we apply the *persistence* metric, we can detect that the model is backdoored and Label 37 is the target label. A similar result is observed when we apply the *shifting distance* and the combined metric (i.e., AI). If the *shift similarity* is used, Labels 38 is reported malicious. Overall, by using the combined metric, GangSweep not only detects backdoor but also accurately pinpoints the target label in all experiments.

Trigger Size. The size of the trigger is an important factor in backdoor attack and detection. We run the test on the GTSRB benchmark, with the increasing Firefox logo trigger (see Fig. 9(d)) from 4×4 to 16×16 pixels, and compare the detection performance with NC [32]. The results are shown in Table 3. NC fails to detect the backdoor when the trigger size is larger than 8×8. This is because NC uses the $L1$ norm of the perturbation as the decision criteria, hence a larger trigger is much closer to the clean label in the $L1$ norm, making the detection less effective. Compared to NC, GangSweep continues its success to detect the backdoor (and the target label) in all cases. Similar results are also observed on the other benchmarks, but omitted due to the space limitation.

Trigger Transparency. An attacker may use triggers of different transparency levels to construct backdoored models, to make the attack stealthier. We run a series of experiments on the GTSRB benchmark, using a Firefox logo trigger covering the whole image (see Fig. 9(e)), ranging the trigger transparency from 0.1 to 0.6 (from less to more transparent). As shown in Table 4, GangSweep succeeds in detecting triggers in all cases, whereas Neural Cleanse can detect the backdoor and target label only when transparency level = 0.1.

Computational Efficiency. To evaluate the efficiency of GangSweep and NC, we implement both of them on Nvidia RTX2080 Mobile Max-Q with 8GB memory. Since we do not require to generate high-quality images with fine-grained details, less than 15 epochs are enough to generate a mask. The result shows that in small scale

Table 2: Backdoor detection using different metrics.

Benchmarks	Num of triggers	Combined metrics	Anomaly Index	Detected Label
ImageNet	1	Persistence	N/A	1
		Shift Distance	8.1	1,7
		Shift Similarity	1.37	None
		Combined	**4.39**	**1**
GTSRB	2	Persistence	N/A	37
		Shift Distance	7.47	37
		Shift Similarity	2.09	38
		Combined	**3.63**	**37**
CIFAR10	4	Persistence	N/A	0,1,2,4
		Shift Distance	4.37	4
		Shift Similarity	1.48	None
		Combined	**2.27**	**4**

Table 3: Comparison of GangSweep and Neural Cleanse for models backdoored with varying sizes of Firefox logo triggers on GTSRB targeting label 37.

Trigger Size	GangSweep		Neural Cleanse	
	Anomaly Index	Detected Target Label	Anomaly Index	Detected Target Label
4×4	8.66	37	2.56	37
8×8	5.91	37	2.21	37
12×12	8.47	37	1.8	None
16×16	2.55	37	1.6	None

Table 4: Comparison of GangSweep and Neural Cleanse for models backdoored with different transparency levels.

Trigger Transparency	GangSweep		Neural Cleanse	
	Anomaly Index	Detected Target Label	Anomaly Index	Detected Target Label
0.1	8.47	10	5.47	10
0.2	6.75	10	3.06	10,11
0.4	3.35	10	1.8	None
0.6	2.10	10	1.6	None

Table 5: Classification accuracy and attack success rate before and after patching.

Benchmark	Before Patching		After Patching	
	Classification Accuracy	Attack Success Rate	Classification Accuracy	Attack Success Rate
MNIST	98.9%	99.8%	99.1%	0.24%
GTSRB	97.4%	98.9%	98.5%	0.15%
CIFAR10	82.5%	99.0%	91.4%	0.44%
VGG-FACE	70.80%	97.1%	80.6%	5.60%
ImageNet	96.10%	76.8%	98.0%	9.60%

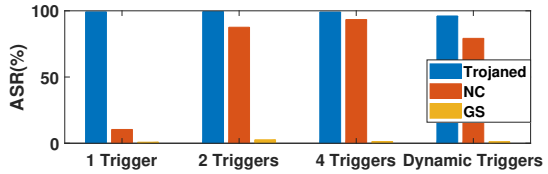

Figure 11: Mitigation of trojaned models embedded with one, two, four, and polymorphic triggers.

datasets, the computing time of GangSweep and NC is at the same level. For example, under CIFAR10, GangSweep takes an average of 913 seconds to evaluate a model, which is comparable to NC that takes 653 seconds. However, GangSweep shows higher computing efficiency in large and high-resolution datasets. For instance, in the VGG-FACE benchmark, GangSweep achieve 8.3x speedup over NC.

5.2 Backdoor Mitigation

For all infected backdoor models, we *patch* them through finetuning the model for 3 epochs with a new data set that includes a small set (10%) of clean validation data and (10%) adversarial data. The adversarial data are clean images stamped with the perturbation mask produced by the generator network in the detection phase and with the same labels as the original images. Since VGG-FACE and ImageNet benchmark only has a small validation dataset with 2622 and 1000 samples, respectively, we finetune the model with the entire validation dataset and their adversarial data.

Table 5 shows the classification accuracy and backdoor attack success rate for malicious models before and after patching. For the MNIST, GTSRB, and CIFAR10 benchmarks, after model patching, the attack success rate drops dramatically to less than 0.5%. For the VGG-FACE and ImageNet benchmarks, the mitigation also reduces the backdoor attack success rate to be under 10%.

Next, we carry out experiments on the CIFAR10 benchmark with four scenarios: (a) inject one 4×4 square white trigger; (b) inject two triggers with the same shape and color located at the two corners of the bottom of an image; (c) inject four triggers, also with the same shape and color located at the four corners of an image; (d) inject a *polymorphic/dynamic multi-trigger* where a trigger is randomly placed at either the left or the right bottom of an image. Fig. 11 illustrates the attack success rates of the trojaned models and the patched models by GangSweep and Neural Cleanse. The readers are referred to Fig. 7 for some reverse-engineered triggers generated by GangSweep and Neural Cleanse. Compared with Neural Cleanse, GangSweep can always find the triggers, while Neural Cleanse can only find part of the trigger or even cannot find any. For example, for the polymorphic multi-trigger, Neural Cleanse generates a mask with a low $L1$ norm but not relevant to the original trigger. This is because Neural Cleanse penalizes the L1 norm of the mask while maximizing the universal misclassification fraction. Thus it would likely stick to local minima and find a different universal perturbation. Therefore, as illustrated in Fig. 11, after patching with the generated mask, Neural Cleanse cannot effectively remove the backdoor. In contrast, GangSweep not only successfully detects the backdoor, but also mitigates it and reduces the attack success rate to lower than 1% in all four scenarios.

Table 6: Mitigation performance under spatial transformation (CIFAR10 benchmark).

Benchmark	GangSweep		Neural Cleanse	
	Clean Classification Accuracy	Poisoned Attack Success Rate	Clean Classification Accuracy	Poisoned Attack Success Rate
Standard	91.4%	0.44%	91.1%	7.80%
Shrink(0.2)	91.2%	1.60%	90.5%	18.8%
Horizontal Flip	93.3%	0.60%	93.8%	74.9%

We further consider attacks with spatial transformations, such as horizontal flipping or shrinking with padding. We run the test on the CIFAR10 benchmark, by applying two transformations on randomly selected images: (1) shrinking an image 20% of the original size, and then zero-padding the shrunk image to the original size; (2) horizontal flipping. These transformations are similar to adding polymorphic multi-triggers to an image, by moving the trigger toward the center of the image, or randomly flipping the trigger horizontally on the image. Table 6 illustrates the backdoor mitigation performance, showing the clean data classification accuracy and backdoor attack success rate after model patching. GangSweep can reduce the attack success rate to less than 2% after model patching, while Neural Cleanse cannot eliminate the backdoor well, especially when the attacker applies the flipping transformation. This again demonstrates that the gradient descent method that depends on the image pixel may fail to find the correct trigger representing the backdoor feature.

6 CONCLUSION

This paper has introduced a new backdoor detection framework, GangSweep, based on generative networks. It has been motivated by a series of intriguing empirical investigations, revealing that a carefully designed generative network can tap into the fundamental weakness of neural backdoors by effectively reconstructing the manifold around the target class and exposing all artifacts planted by the attacker. An efficient outlier detection mechanism has been devised to identify backdoor according to distinct statistical properties. Extensive experiments have shown that GangSweep is effective against state-of-the-art backdoor attacks across different datasets and various numbers, patterns, and sizes of triggers.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under Grant CNS-1828593, OAC-1829771, EEC-1840458, CNS-1745632 and CCF-1850045, Office of Naval Research under Grant N00014-20-1-2065, and the Commonwealth Cyber Initiative, an investment in the advancement of cyber R&D, innovation and workforce development. For more information about CCI, visit cyberinitiative.org.

REFERENCES

[1] N. Carlini and D. Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. In *Proceedings of 2017 IEEE Symposium on Security and Privacy (SP)*. 39–57.

[2] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. 2019. Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering. *The Thirty-Third AAAI Conference on Artificial Intelligence Safety Workshop* (2019).

[3] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *Proceedings of 2009 IEEE Conference on Computer Vision and Pattern Recognition*. 248–255.

[4] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song. 2018. Robust Physical-World Attacks on Deep Learning Visual Classification. In *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1625–1634.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of Advances in Neural Information Processing Systems(NeurIPS)*. 2672–2680.

[6] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg. 2019. BadNets: Evaluating Backdooring Attacks on Deep Neural Networks. *IEEE Access* (2019), 47230–47244.

[7] Wenbo Guo, Lun Wang, Xinyu Xing, Min Du, and Dawn Song. 2019. Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems. *arXiv preprint arXiv:1908.01763* (2019).

[8] Frank R Hampel. 1974. The influence curve and its role in robust estimation. *Journal of the american statistical association* 69, 346 (1974), 383–393.

[9] Liang Han, Zhaozheng Yin, Zhurong Xia, Li Guo, Mingqian Tang, and Rong Jin. 2019. Vision-Based Price Suggestion for Online Second-Hand Items. In *Proceedings of the 27th ACM International Conference on Multimedia (MM '19)*. 1988–1996.

[10] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*. 675–678.

[11] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*. Springer, 694–711.

[12] Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations* (2014).

[13] Alex Krizhevsky. 2009. Learning Multiple Layers of Features from Tiny Images. *University of Toronto* (2009).

[14] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533* (2016).

[15] Yann LeCun, LD Jackel, Léon Bottou, Corinna Cortes, John S Denker, Harris Drucker, Isabelle Guyon, Urs A Muller, Eduard Sackinger, Patrice Simard, et al. 1995. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective* (1995), 276.

[16] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2018. Visualizing the loss landscape of neural nets. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*. 6389–6399.

[17] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2018. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 273–294.

[18] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning Attack on Neural Networks. In *Proceedings of the 25nd Annual Network and Distributed System Security Symposium (NDSS)*.

[19] Jiaxin Ma, Hao Tang, Wei-Long Zheng, and Bao-Liang Lu. 2019. Emotion Recognition using Multimodal Residual LSTM Network. In *Proceedings of the 27th ACM International Conference on Multimedia*. 176–183.

[20] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1765–1773.

[21] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. 2015. Deep Face Recognition. In *Proceedings of the British Machine Vision Conference (BMVC)*, Mark W. Jones Xianghua Xie and Gary K. L. Tam (Eds.). BMVA Press, 41.1–41.12.

[22] Ximing Qiao, Yukun Yang, and Hai Li. 2019. Defending neural backdoors via generative distribution modeling. In *Proceedings of the Advances in Neural Information Processing Systems(NeurIPS)*. 14004–14013.

[23] Jonas Rauber, Wieland Brendel, and Matthias Bethge. 2017. Foolbox: A Python toolbox to benchmark the robustness of machine learning models. In *Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning*. http://arxiv.org/abs/1707.04131

[24] Mauro Ribeiro, Katarina Grolinger, and Miriam AM Capretz. 2015. Mlaas: Machine learning as a service. In *Proceedings of the 14th IEEE International Conference on Machine Learning and Applications (ICMLA)*. 896–902.

[25] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. 2020. Hidden Trigger Backdoor Attacks. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*. 11957–11965.

[26] Ahmed Salem, Rui Wen, Michael Backes, Shiqing Ma, and Yang Zhang. 2020. Dynamic Backdoor Attacks Against Machine Learning Models. *arXiv preprint arXiv:2003.03675* (2020).

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

[28] David Stutz, Matthias Hein, and Bernt Schiele. 2019. Disentangling adversarial robustness and generalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6976–6987.

[29] Ning Sun, Hongxi Bai, Yuxia Geng, and Huizhu Shi. 2017. Price evaluation model in second-hand car system based on BP neural network theory. In *Proceedings of the 2017 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. 431–436.

[30] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).

[31] Matthew Turk and Alex Pentland. 1991. Face recognition using eigenfaces. In *Proceedings of the 1991 IEEE computer society conference on computer vision and pattern recognition*. 586–587.

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

[33] Haohan Wang, Xindi Wu, Pengcheng Yin, and Eric P Xing. 2019. High frequency component helps explain the generalization of convolutional neural networks. *arXiv preprint arXiv:1905.13545* (2019).

[34] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. 2018. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610* (2018).

[35] Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y Zhao. 2019. Latent Backdoor Attacks on Deep Neural Networks. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2041–2055.

[36] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*. 2223–2232.

[37] Peiqin Zhuang, Yali Wang, and Yu Qiao. 2018. Wildfish: A large benchmark for fish recognition in the wild. In *Proceedings of the 26th ACM international conference on Multimedia*. 1301–1309.