PatchBackdoor: Backdoor Attack against Deep Neural Networks without Model Modification

Yizhen Yuan
Institute for AI Industry Research (AIR), Tsinghua University
Beijing, China
yuanyz21@mails.tsinghua.edu.cn

Rui Kong
Shanghai Jiao Tong University
Shanghai, China
kongrui@sjtu.edu.cn

Shenghao Xie
Wuhan University
Wuhan, China
xieshenghao@whu.edu.cn

Yuanchun Li*
Institute for AI Industry Research (AIR), Tsinghua University
Beijing, China
Shanghai AI Laboratory
Shanghai, China
liyuanchun@air.tsinghua.edu.cn

Yunxin Liu
Institute for AI Industry Research (AIR), Tsinghua University
Beijing, China
Shanghai AI Laboratory
Shanghai, China
liuyunxin@air.tsinghua.edu.cn

ABSTRACT

Backdoor attack is a major threat to deep learning systems in safety-critical scenarios, which aims to trigger misbehavior of neural network models under attacker-controlled conditions. However, most backdoor attacks have to modify the neural network models through training with poisoned data and/or direct model editing, which leads to a common but false belief that backdoor attack can be easily avoided by properly protecting the model. In this paper, we show that backdoor attacks can be achieved without any model modification. Instead of injecting backdoor logic into the training data or the model, we propose to place a carefully-designed patch (namely backdoor patch) in front of the camera, which is fed into the model together with the input images. The patch can be trained to behave normally at most of the time, while producing wrong prediction when the input image contains an attacker-controlled trigger object. Our main techniques include an effective training method to generate the backdoor patch and a digital-physical transformation modeling method to enhance the feasibility of the patch in real deployments. Extensive experiments show that PatchBackdoor can be applied to common deep learning models (VGG, MobileNet, ResNet) with an attack success rate of 93% to 99% on classification tasks. Moreover, we implement PatchBackdoor in real-world scenarios and show that the attack is still threatening.

CCS CONCEPTS

• **Security and privacy → Malware and its mitigation**; • **Computing methodologies → Computer vision**.

*Corresponding author.

MM '23, October 29-November 3, 2023, Ottawa, ON, Canada

ACM ISBN 979-8-4007-0108-5/23/10.
https://doi.org/10.1145/3581783.3612032

KEYWORDS

Backdoor attack, Neural Networks, Adversarial Patch

ACM Reference Format:
Yizhen Yuan, Rui Kong, Shenghao Xie, Yuanchun Li, and Yunxin Liu. 2023. PatchBackdoor: Backdoor Attack against Deep Neural Networks without Model Modification. In *Proceedings of the 31st ACM International Conference on Multimedia (MM '23), October 29-November 3, 2023, Ottawa, ON, Canada.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3581783.3612032

1 INTRODUCTION

Deep Neural Networks (DNNs) are widely used in many security-critical edge systems such as autonomous driving [8], face authentication [42] and medical diagnosis [31, 35]. While bringing great convenience in many applications, the security issues of deep learning (DL) are also gaining extensive attention.

It is widely known that DNN is vulnerable to many types of attacks, and the backdoor attack is a major one of them. Most backdoor attack approaches conduct the attack by training the victim model with poisoned datasets [13, 28]. The trained model will have a high benign accuracy when normal test samples are predicted, while the model will give wrong predictions when certain attacker-controlled triggers are present. Some other attackers conduct the attack by directly modifying the model structures and/or weights[6], which usually happens in third-party machine learning platforms where users outsource the training or serving to untrusted service providers. The attackers can modify their models to inject backdoors before the models are actually deployed.

A primary limitation of the backdoor attack is the need to modify the model, which could be challenging in most security-critical scenarios. For instance, most autonomous driving companies use self-collected and carefully-filtered datasets for training and will not outsource the training to cloud service either. When being deployed, the model can be placed in the read-only memory to ensure integrity. Therefore, although the backdoor attack seems threatening, it is less concerning for most model developers that can securely manage the training datasets and deployed models.

In this paper, we propose to achieve backdoor attack without modifying the victim model. Our insight is to inject backdoor logic

by attaching a constant input patch, which is feasible since many vision applications have an unchanged foreground/background. Such an attack is dangerous because (i) it is difficult for model developers to avoid such an attack since the attack happens after the model is securely deployed and (ii) attackers can flexibly control the backdoor logic to implement practical attacks.

The idea of backdooring deep neural networks with an input patch is closely related to adversarial patch attacks [2, 13], which have been extensively studied in the literature. However, adversarial patch attacks aim to directly produce a wrong prediction if a carefully-designed patch is presented in the input. Instead, our goal is to inject a hidden backdoor logic with a constant patch in the foreground or background. Our method is a novel connection between the backdoor and adversarial patch attacks.

Our approach includes two main techniques. First, we adopt a distillation-style training method to generate the backdoor patch without labeled training data. Specifically, we design a training objective that jointly maximizes the patch stealthiness (*i.e.,* mimicing the benign model behavior on normal inputs) and the attack effectiveness (*i.e.,* producing misbehavior on trigger conditions).

Second, to enhance the attack effectiveness in the physical world, we propose to model digital-physical visual shift with differentiable transformations (including a shape transformation and a color transformation), so that the digitally-trained backdoor patch can be directly adopted in the physical world.

To evaluate our approach, we perform experiments on three datasets (CIFAR10 [24], Imagenette [18], Caltech101 [9]) and three models (VGG[40], ResNet[16], MobileNet[39]). The results demonstrate that our attack is robust under different situations with a high attack success rate of 93% to 99%. Meanwhile, our attack is stealthy, since the backdoor patch does not affect the benign accuracy of the victim model, and can hardly be detected with out-of-distribution (OOD) detectors. We also show that our attack is effective at different levels of over-parameterization by testing it with different pruning ratios (0%, 30%, 60%, 90%). By deploying the attack to the physical world, we demonstrate the feasibility of our attack in real-world scenarios.

This paper has the following research contributions:

- To the best of our knowledge, this is the first backdoor attack against neural networks that does not require any modification on the victim models.
- We design a training scheme for the attack, which can generate an effective backdoor patch efficiently with minimal data requirements.
- We introduce a digital-physical transformation modeling method that can improve the attack effectiveness in the real-world deployment.
- We conduct thorough evaluations of the effectiveness and anti-detection abilities of our attack.

The source code is at https://github.com/XaiverYuan/PatchBackdoor

2 BACKGROUND AND MOTIVATION

2.1 Backdoor Attack against DNN

A deep neural network (DNN) can be viewed as a function f that maps an input x to a prediction y. Typically, a DNN is trained to maximize the following probability:

$$P(f(x) = \hat{y}) \tag{1}$$

where f is the DNN, x is the image to be classified, $\hat{y}$ is the ground truth label for the corresponding input x.

The backdoor attack against a DNN aims to inject a hidden logic into the model, so that the model behaves normally on clean input data, while producing wrong predictions on certain conditions (*e.g.,* the input image contains an attacker-controlled trigger). Formally, the objective of backdoor attacker is to turn the victim model f to a model f' that maximizes following two probabilities:

$$P(f'(x) = \hat{y}), \quad P(f'(x \oplus trigger) = y_t) \tag{2}$$

where $x \oplus trigger$ is the image with the trigger, y_t is the target label for corresponding x.

Backdoor attack is difficult to defend against since the attacker-controlled trigger can be arbitrary and unknown to the users. For example, the backdoor trigger could be invisible to humans if the attackers limit the perturbation boundary [12]. The trigger could also be as small as only one pixel [41]. Defending against backdoor attacks typically requires analyzing the poisoned datasets [3] and/or tuning/retraining the victim model [32].

A significant limitation of existing backdoor attack approaches is the need to modify the victim model. Specifically, to inject a backdoor, the attacker needs to insert malicious data samples into the training dataset or alter the model. Since the training data and the model are usually properly protected by the developers, the applicable scenarios of existing backdoor attacks are limited.

Therefore, we are motivated to investigate whether it is feasible to achieve backdoor attack without changing the model and the training datasets. If so, it could pose a significant threat to the security-critical deep learning applications deployed in the real world, since (i) the attacker can flexibly trigger the misbehavior of the model with arbitrary objects and (ii) the attack can be easily achieved after the model is deployed.

2.2 Opportunity: Backdoor as a Patch

Static foreground/background can be an attack surface. Many vision applications are deployed to smart cameras with unchanged static foregrounds and/or backgrounds. For example, smart cameras could be deployed in airports to check if terrorists pass by [47]. Security cameras are also deployed in military reconnaissance [1].

Such static foregrounds/backgrounds provide perfect attack surfaces for attaching malicious patches. It is much easier for an attacker to modify the content in the static foregrounds/backgrounds (which may belong to public spaces) than changing the models that are usually securely protected in the users' private devices. The content in the static foreground/background is fed into the model as a part of input, which can interact with the varying content in the input to generate different behaviors.

Adversarial patch attack. The idea of achieveing attack by controlling a portion of input is not new. Prior research has found that DNNs are vulnerable to adversarial patches, *i.e.,* a carefully-designed input patch attached to the input image. Such an attack could work because DNNs are known to have redundant neurons, which creates unnecessary logic. Adversarial patches could take advantage of this and activate those redundant neurons to misdirect

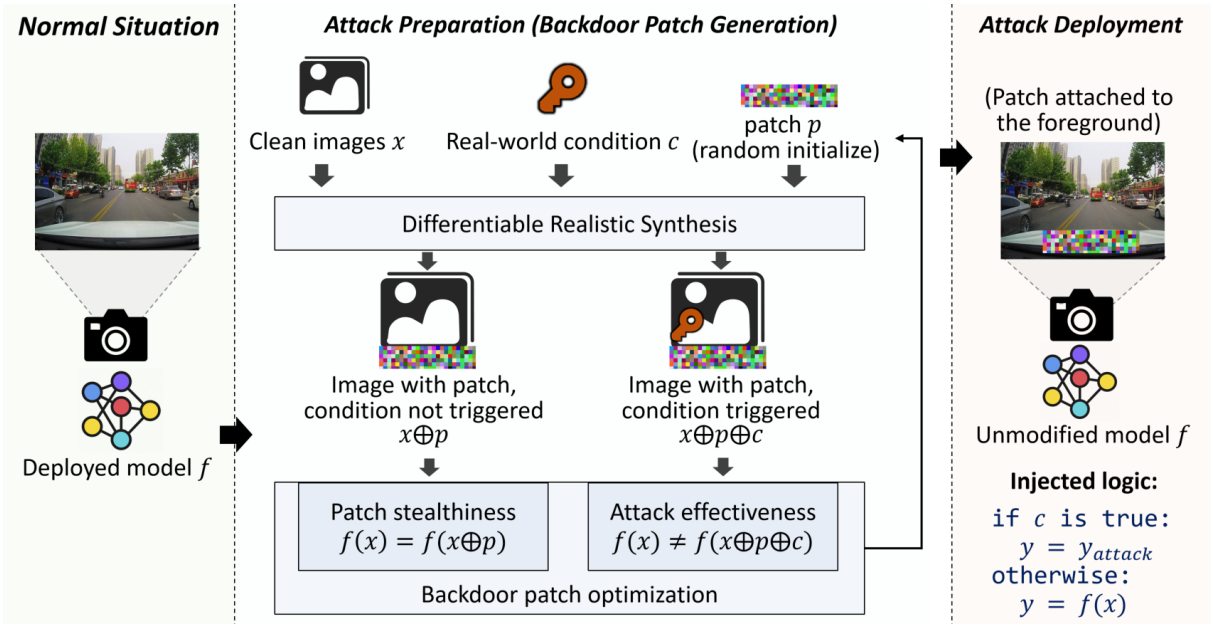

Figure 1: The workflow of PatchBackdoor attack.

the prediction. Previous researchers also found that the adversarial patch is a way to draw the attention of the model. Therefore, it might be possible to interfere with the *decision logic* of a neural network with an input patch.

Although the adversarial patch attack has demonstrated the ability to change the prediction results with a small patch, it is not practical in the real world because the patch is usually a special image generated by training, which limits the flexibility of controlling trigger conditions.

Our idea: Injecting backdoor logic with a patch. Instead of directly producing misbehavior with an adversarial patch, our idea is to inject backdoor logic with a patch, where the logic can be controlled by the attacker.

Such an attack has two main advantages. First, unlike other backdoor attacks that need to change the model, our attack does not require modification on the model and the training data. Second, the trigger conditions of the attack can be flexibly configured, *e.g.*, an arbitrary object exists in the camera view, or the environment in under certain lighting conditions.

Meanwhile, implementing the conditional patch attack involve several challenges.

- A backdoor attack needs to fulfill two requirements at the same time. First, the attack needs to remain deactivated when the image is normal. Second, the attack needs to be stable when the input image satisfies the trigger condition. Squeezing the logic into a small patch in the input is non-trivial.
- It is hard for attackers to obtain the labeled data used for training the victim model. How to generate the backdoor patch with few or no labeled training data is challenging.
- To be effective in the real applications, the input patch containing the backdoor logic need to be functional in the physical world.

Table 1: The difference between existing attacks and ours.

Method	No Model Modification	Arbitrary Trigger	Physical-world Feasibility
Backdoor Attack	False	True	True
Adversarial Patch	True	False	True
Adversarial Perturbation	True	False	False
PatchBackdoor (ours)	True	True	True

3 OUR APPROACH: PATCHBACKDOOR

3.1 Overview

Threat Model. Suppose there is a deep learning-based vision model deployed in the physical world. The model takes the image captured by a camera as the input, and the camera view contains a constant foreground or background (*e.g.*, a wall or a car hood near the camera). The attacker can inject a backdoor logic (*i.e.*, letting the model behavior as usual in most of the time, while producing wrong predictions if the input image satisfies an attacker-defined condition) by attaching a patch onto the constant foreground/background in the camera view. We assume the attacker has read access to the victim model, so that it can use the model weights to do backward propagation. However, it is impossible for the attacker to modify the model weights. Moreover, unlike most backdoor attack approaches, we assume that the attacker has no read or write access to the original training data.

The overview of our approach is shown in Figure 1. The attack is achieved by training an input patch p, namely *backdoor patch*.

In the attack preparation stage, we first obtain the deployed model from the victim application. Then we choose a real-world condition that could be activated conveniently as the target attacking condition. We also need to obtain a set of clean images that are normal input images of the victim model. Such an image set

should be easy to obtain because the working environment of the victim model is already known. The backdoor patch is randomly initialized at the beginning.

Based on the clean images, the selected trigger condition, and the randomly-initialized backdoor patch, we can synthesize two types of images. The first is the normal images that have the backdoor patch attached but the trigger condition not satisfied. The second is the target images that contain the backdoor patch and satisfy the trigger condition. The backdoor patch is iteratively optimized to let the normal images produce normal predictions (so that the patch looks harmless), while letting the target images produce wrong predictions expected by the attacker.

After the backdoor patch is trained, it is deployed to the victim application, by attaching the patch onto a surface in the camera view. In this way, the victim model is unmodified, but its decision logic is altered by the attacker.

3.2 Backdoor Patch Training

In Equation 2, the objective of our attack is almost the same as the backdoor attack. However, instead of modifying the victim model f to create a new model f', we leave the model f unchanged. $x \in X$ represents a normal image in a dataset X. Our patch is denoted as p, and $x \oplus p$ means to attach the patch p to the image x.

If an input image satisfies the trigger condition c (*e.g.,* a trigger object is present, the lighting is in a specific condition, etc.), we represent it as $x \oplus c$.

Therefore, our attack aims to maximize two probabilities:

$$P(f(x \oplus p) = \hat{y}), \quad P(f(x \oplus p \oplus c) = y_{target})$$

To meet the above objective, we define the following losses:

$$L_{clean}(p) = \sum_{x \in X} L(f(x \oplus p), \hat{y}) \tag{3}$$

$$L_{attack}(p) = \sum_{x \in X} L(f(x \oplus p \oplus c), y_{target}) \tag{4}$$

where L_{clean} is the loss function to encourage patch stealthiness, *i.e.,* the normal input with the patch should produce the correct prediction. L_{attack} is the loss to encourage backdoor attack effectiveness, *i.e.,* the input image with the patch that satisfies the trigger condition should produce the attacker-specified wrong prediction.

The optimal backdoor patch $\hat{p}$ can be found by minimizing the both losses:

$$\hat{p} = \underset{p}{argmin}\,(\alpha L_{clean}(p) + (1-\alpha) L_{attack}(p)) \tag{5}$$

where α is a hyperparameter to balance the clean accuracy and attack success rate. If α is closer to one, then the patch will focus more on being stealthy. If α is closer to zero, the patch focuses more on attacking. Attackers could train a standard adversarial patch if we set our α to zero. In practice, setting α to 0.5 is usually a good choice in most cases. However, sometimes α is not easy to determine depending on many factors, including the backdoor patch size, the trigger size, and the image resolution. In that case, we could adjust the hyperparameter α during training to balance the two losses. This is especially helpful in some cases (*e.g.,* when training on small images), where the attack effectiveness loss L_{attack} and stealthiness loss L_{clean} change at significantly different speeds. Using α can balance the two losses and lead to better tradeoff between the clean accuracy and the attack success rate.

Figure 2: The calibration board used for physical-world transformation modeling, including the digital calibration board (left), a photo of the calibration board in the physical world (center), a digital version of the calibration board generated with our differentiable transformation (right).

As we mentioned before, one challenge in generating the backdoor patch is the lack of labeled data. Thus, the ground-truth label $\hat{y}$ in Equation 3 is usually unknown. We borrow the idea of model distillation to solve this challenge - We can consider the original model with the constant patch as a new model $f'(x) = f(x \oplus p)$, and the pixels in the constant patch p are the parameters of the new model f'. Then, f' can be trained by distilling knowledge from f, *i.e.,* Equation 3 can be written as:

$$L_{clean}(p) = \sum_{x \in X} L(f(x \oplus p), f(x)) \tag{6}$$

In this way, the attacker only needs an unlabeled dataset of clean images, which is easy to obtain, and the trained patch can mimic the behavior of the original model, improving the patch stealthiness.

3.3 Digital-Physical Transformation

We have so far described how to train the backdoor patch in the digital world. However, in practice, the patch should be attached to a surface in the physical world to conduct the attack. The digitally-trained backdoor patch may become invalid due to the difference between the digital and physical worlds. Thus, we need to take the digital-physical gap into the consideration when training the patch.

Our key idea is to model the digital-physical gap with a differentiable transformation, and optimizing the backdoor patch using this transformation.

Patch Transformation Modeling. A digital-physical transformation can be separated into two parts, including shape transformation and color transformation.

Both the transformations are captured with a carefully-designed calibration board, as shown in Figure 2. When preparing the attack, attackers only need to print the calibration board and put it where they plan to attach the backdoor patch. Then they need to take a few photos of the calibration board, indicating how it will look like in the input of the victim model. The goal of patch transformation modeling is to find a differentiable function that maps the digital calibration board to its physical-world counterpart.

A regular shape transformation can be implemented as a parameterized warp operation [38], which stretches a square image to fit a quadrilateral. However, the surface to attach the backdoor patch might not be flat, so we propose to split the surface into several smaller surfaces and capture their shape transformations individually. Our calibration board already contains several ArUco markers

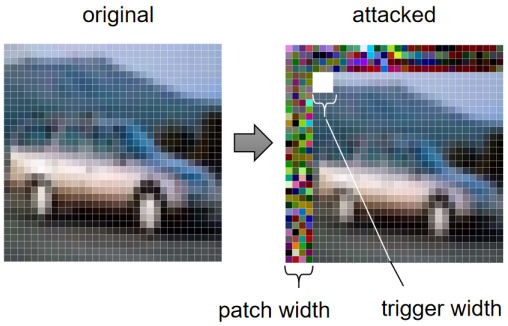

Figure 3: An example of how the backdoor patch and backdoor trigger are attached to the image.

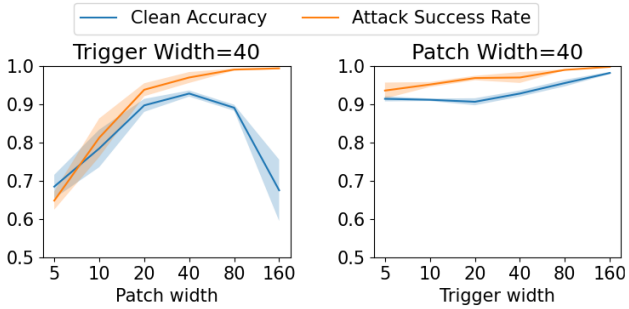

Figure 4: The correlation between the attack effectiveness and the sizes of backdoor patches and triggers.

that separate the patch to multiple micro-surfaces. We model the simple shape transformation of each micro-surface and combine them to form the whole shape transformation.

The color transformation is modeled as a lightweight convolutional neural network (CNN). As shown in Figure 2, our calibration board contains multiple blocks filled with different colors. By aligning the pixels in the digital calibration board and the physical patch based on the ArUco markers, we can obtain the RGB value mappings between the digital and physical patches. We use a single-layer CNN with a 3×3 convolution filter to capture the RGB mapping, and use the MSE loss to minimize the difference between CNN-generated color and the actual color. When the loss converges, the generated CNN is used as the color transformation.

Combining the two transformation modeling techniques, the attacker can flexibly obtain the mapping relation between the digital patch and the physical patch deployed in different environments. In our attack, we consider at least two environments, including the clean environment where the backdoor should remain deactivated and the attacking environment where the patch should cooperate with the trigger to take effect.

Both the shape transformation and the color transformation described above are differentiable, so it is possible to train the patch with backpropagation. When we want to train a patch that is applicable in the physical world, we simply need to apply the transformations before feeding the patched images into the target model.

4 EVALUATION

Experiment setup. In most experiments, the backdoor patch is placed as a top-left sidebar in the input image, and the original image is resized to fit into the bottom right corner. The backdoor trigger is a white square placed next to the sidebar. The models used in the experiment are all pre-trained on the original datasets. We use patch width and trigger width to describe the size of the patch and the trigger. Figure 3 illustrates an original image and its corresponding attacked image, as well as the definitions of patch width and trigger width. The two important metrics in our experiments are clean accuracy (ACC for short), *i.e.,* the accuracy of the model on the normal dataset after the backdoor patch is applied, and attack success rate (ASR for short), *i.e.,* the ratio of misclassified images among all images with the backdoor trigger presented.

The experiments are conducted on three CNN models including VGG [40], ResNet [16], and MobileNet-V2 [39] and three datasets including CIFAR-10 [24], Imagenette [18], and Caltech [9].

4.1 Attack Effectiveness

We first evaluate the effectiveness of our attack by measuring the attack success rate and clean accuracy on the common models and datasets. Since the three datasets have different sizes, the patch widths and trigger widths vary. Specifically, the patch width and trigger width are 7 and 3 respectively on CIFAR-10 and 36 and 38 on Imagenette and Caltech.

The results are shown in Table 2. PatchBackdoor achieves a high attack success rate (93%-99%) while maintaining a reasonable clean accuracy. The high attack success rate demonstrates the vulnerability of target models to our attack, while the clean accuracy illustrates that the backdoor patches do not substantially affect the performance of models on normal unperturbed input.

The accuracy drops for CIFAR-10, Imagenette, and Caltech are around 10%-15%, 0%-7%, and 4-10% respectively. The reason why the accuracy drop for CIFAR-10 is higher is probably because the CIFAR-10 dataset contains images of smaller size, resulting in fewer pixels and reduced information capacity in the backdoor patch.

The attack effectiveness results for the three models are close. However, an interesting obversation is that the models with higher original accuracy also produce higher clean accuracy after the backdoor patch is attached. This probably means that the learning abilities of the victim models can be transferred to the backdoor patches when the patches are trained to maximize the stealthiness.

We have also compared the effectiveness of PatchBackdoor with the classical data poisoning attack (BadNet) on three different datasets. On all datasets, our attack success rates are higher than data poisoning with 1% poisoning ratio. On datasets with larger image sizes (Imagenette and Caltech), our attack performance is even higher than 10% data poisoning. However, our attack is slightly less effective than 10% data poisoning on the CIFAR-10 dataset. This is due to the fact that the reduced image size creates less opportunity for injecting backdoor logic. After all, BadNet is a attack that needs data poisoning and model training.

Influence of Patch Size and Trigger Size. We further analyze the relation between the attack effectiveness and the sizes of backdoor patch and trigger. We select the Imagenette dataset for this evaluation due to its large image size. The victim model is ResNet.

Table 2: The clean accuracy and the attack success rate on different models and datasets. ACC stands for clean accuracy. ASR stands for attack success rate. P-ratio stands for poisoning ratio.

ID	Model	Dataset	Original Acc.	PatchBackdoor		BadNet P-ratio=5%		BadNet P-ratio=10%	
				ACC	ASR	ACC	ASR	ACC	ASR
1	MobileNet-V2	CIFAR 10	93.61%	83.41%	95.91%	90.45%	91.55%	89.08%	96.00%
2	ResNet50	CIFAR 10	94.26%	84.01%	95.53%	90.30%	91.25%	90.99%	95.39%
3	VGG16 bn	CIFAR 10	93.65%	79.00%	93.11%	88.62%	87.78%	88.96%	96.69%
4	MobileNet-V2	Imagenette	96.05%	94.75%	98.98%	89.43%	93.55%	92.84%	93.61%
5	ResNet50	Imagenette	97.24%	90.24%	98.30%	90.50%	91.18%	85.20%	95.46%
6	VGG16 bn	Imagenette	95.92%	95.95%	96.82%	86.93%	92.33%	87.57%	95.41%
7	MobileNet-V2	Caltech	89.28%	85.08%	97.64%	89.87%	77.83%	88.84%	89.50%
8	ResNet50	Caltech	92.16%	88.78%	98.00%	91.44%	76.48%	90.75%	88.91%
9	VGG16 bn	Caltech	90.95%	80.49%	93.28%	76.25%	77.66%	76.35%	89.41%

Table 3: The effectiveness of our attack when trained and evaluated on models with different pruning ratios.

Train \ Test	Prune 0%		Prune 30%		Prune 60%		Prune 90%	
	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
Prune 0%	97.5	98.0	95.5	99.3	94.7	95.8	93.7	11.4
Prune 30%	98.3	83.2	97.4	97.9	95.6	94.9	94.1	11.9
Prune 60%	98.5	13.2	98.5	17.8	96.8	98.5	94.7	10.9
Prune 90%	98.1	10.5	98.0	10.6	97.3	11.3	94.9	97.9
Prune 0%&90%	96.4	98.1	94.5	99.2	94.5	94.6	94.3	97.8

All other parameters except the patch width and trigger width held constant at default values.

As shown in Figure 4, both the patch size and trigger size influence the clean accuracy and attack success rate. As the patch width increases, both the ACC and ASR increase initially, which is intuitive because the larger patch width represents a larger area for the attacker to manipulate, providing more opportunities to plant backdoor logic. However, as the patch width continues to increase beyond 40, the ASR continues to increase while the ACC starts to drop. This is because the area of the original image becomes too small to carry enough information for classification.

On the contrary, the increase of trigger width constantly leads to the increase of clean accuracy. This may seem counterintuitive since the clean accuracy is measured on clean images that do not contain the trigger. The main reason is that the backdoor patch contained in PatchBackdoor has two competing objectives - (i) remaining highly accurate under normal conditions, and (ii) producing incorrect predictions when combined with the trigger. PatchBackdoor achieves both these objectives by optimizing the pixels in the backdoor patch. A larger trigger size makes the latter objective easier to attain, allowing the backdoor patch to focus more on the former objective of improving clean accuracy under normal conditions.

4.2 Attack Robustness

We further investigate whether our backdoor patch remains effective after the victim model is modified. The experiments are all conducted with the Imagenette dataset and ResNet50. Both the patch width and trigger width are set to 40.

Robustness against pruning. We first consider the case where the victim model is pruned after our attack. Specifically, we consider four pruning ratios of a same model, including 0% (the original model), 30%, 60%, and 90%. The corresponding model accuracies are 99.46%, 99.51%, 99.54%, and 97.43%, respectively. The models are pruned and fine-tuned with the global L1 pruning [14]. As shown in Table 3, the backdoor patch trained on the original model or the 30%-pruned models performs well on the 60%-pruned model. However, the attack trained on models with higher pruning ratios cannot transfer effectively to models with lower pruning ratios. The reason for this is that highly pruned models requiring more fine-tuning, which results in larger modification of the model parameters and subsequently worsens the transferability. In the last row, the attack is trained on both the original model and the 90%-pruned model. Surprisingly, the results indicate that the attack remains effective on all other pruned models. This finding suggests that training the patch on multiple models can enhance its robustness.

Robustness against fine-tuning. Similarly, we test the effectiveness of the patch backdoor after the victim model is fine-tuned. In our study, we initially trained the backdoor patch on the original model, which achieved a clean accuracy of 96.18% and an attack success rate of 99.13%. Subsequently, we fine-tuned the model for 40 epochs (accuracy increased from 99.34% to 99.52%). Next, we assessed the patch's performance on the finetuned model, revealing a clean accuracy of 95.11% and an attack success rate of 99.54%. These results closely resemble the effectiveness observed on the original model, thereby demonstrating the robustness of our attack against normal fine-tuning. When the model parameters are significantly changed (*e.g.,* fine-tuned on different data), the attack trained with the original model may be less effective. In that case, the attacker can re-generate the backdoor patch with the new model, which is quite efficient according to Section 4.6.

Robustness against distillation. We also consider the case when the attack is trained on the original model and applied to a distilled model. Such robustness is useful when the model parameters are not accessible - attackers can distill a surrogate model using the inference interface and train the attack on the surrogate model. Specifically, we assume that the attacker is aware of the model structure and has access to a similar distribution of the original dataset. After distilling and training the patch on the surrogate model, we achieve a clean accuracy of 93.63% and an attack success rate of 99.36%. When testing the patch on the victim model, we

Table 4: The AUROC scores computed by different Out-Of-Detection detectors for different datasets. The in-distribution dataset is a CIFAR-10 subset, and the compared datasets include another CIFAR-10 subset, CIFAR-100, SVHN, and PatchBackdoor (CIFAR-10 with a backdoor patch attached). The higher AUROC score means that the compared dataset can be more easily detected as OOD.

Data	FSSD[20]	Baseline[17]	Maha[25]	ODIN[29]
CIFAR-10 sep	94.0%	59.3%	91.4%	90.4%
CIFAR-100	74.4%	79.8%	54.8%	81.5%
SVHN	99.5%	89.9%	99.1%	96.6%
PatchBackdoor	78.3%	68.3%	68.1%	69.5%

observe a clean accuracy of 92.23% and an attack success rate of 98.32%, which are just slightly lower than on the surrogate model.

4.3 Stealthiness against Detection

Most defenses against backdoors do not apply to our approach, because they mostly concentrate on identifying or mitigating manipulations made to the training datasets or the models, while our approach does not make any modification to the model architecture, model parameters, training data, and training procedure. The defenses against adversarial patches also do not apply. Adversarial patch defenses are mostly based on the fact that adversarial patch attacks aim to alter the prediction when the patch is present [21, 33]. However, the backdoor patches in PatchBackdoor aim to keep the original predictions, which is a fundamentally different goal as compared with adversarial patches.

However, since the backdoor patch needs to be constantly placed in the camera view, it will alter the data distribution of camera images and may be detected by out-of-distribution (OOD) detectors. Therefore, we use different OOD detection methods (Baseline[17], FSSD[20], Maha[25], ODIN[29]) to see whether they can distinguish the PatchBackdoor-modified images with other normal images. We use the CIFAR-10 dataset and ResNet model in this experiment, and the patch width and trigger width are 7 and 3 respectively.

We train the OOD detectors with a subset of CIFAR-10 as the in-distribution dataset, and use them to measure the OOD degrees of different out-of-distribution datasets. The compared OOD datasets include another subset of CIFAR-10 with different classes (CIFAR-10 sep), the CIFAR-100 dataset, and the SVHN dataset. The OOD degree is measured as the AUROC metric, and a higher AUROC means that the dataset is more easily detected as OOD.

The results are shown in Table 4. We can see that the SVHN dataset can be easily detected as OOD with high AUROC scores, while the AUROC scores for the CIFAR-10 subset and CIFAR-100 are much lower. It is an intuitive result since SVHN is indeed more distributionally different with CIFAR-10 than the other two datasets. The dataset with PatchBackdoor (*i.e.,* CIFAR-10 images with the backdoor patch attached) yields much lower AUROC scores than SVHN, and sometimes even lower than CIFAR-10 subset and CIFAR-100. This means that our backdoor patches are not easy to detect with common OOD detectors.

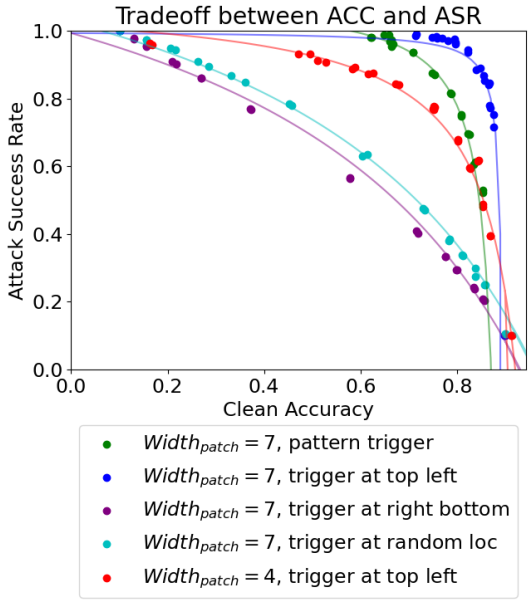

Figure 5: The tradeoff between clean accuracy (ACC) and attack success rate (ASR) under different settings.

4.4 ACC-ASR Tradeoff under Different Settings

In this subsection, we analyze the tradeoff between clean accuracy (ACC) and attack success rate (ASR) under different settings of PatchBackdoor. Our method incorporates a hyperparameter, α, that governs the weighting of two competing loss functions during model training. The first loss function optimizes standard classification accuracy on benign examples, while the second aims to maximize the misclassification of adversarial examples to a specific target label. By adjusting α, we can precisely control the trade-off between model accuracy on normal inputs and the success rate of the implanted backdoor. We use the CIFAR-10 dataset and the ResNet model in this experiment.

In Figure 5, each point represents an experiment. The experiments are conducted with different settings, including the backdoor patch width, the location of trigger, and the type of trigger. Overall, we can see that the ACC and ASR exchanges with each other under all settings. This is intuitive as we have mentioned that they are competing goals of our backdoor patch. In cases when the patch size and trigger location are proper (*e.g.,* the blue dots), the ACC and ASR can both be high.

If the backdoor patch size is smaller (the red dots), both the ACC and the ASR decrease. The reason has been discussed in Section 4.1.

The comparison between the blue, clay, and purple curves demonstrates that the attack performance decreases as the trigger appears at positions farther from the patch. This is because that the closer distance between the trigger and the patch makes it easier for combining them to produce misbehavior. Meanwhile, even when the location is fixed (purple dots), the performance is still worse than that of a randomly positioned trigger (clay dots). This indicates that although the randomness of trigger location may have some impact on the attack performance, the distance between the patch and the trigger is a more influential factor.

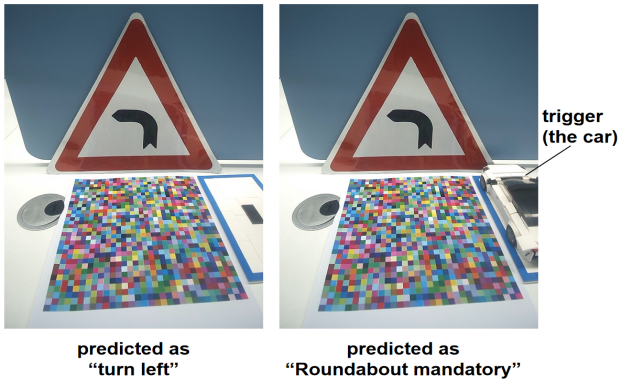

Figure 6: Physical-world feasibility of PatchBackdoor. The victim model is a traffic sign classifier, the backdoor trigger is the car model on the right.

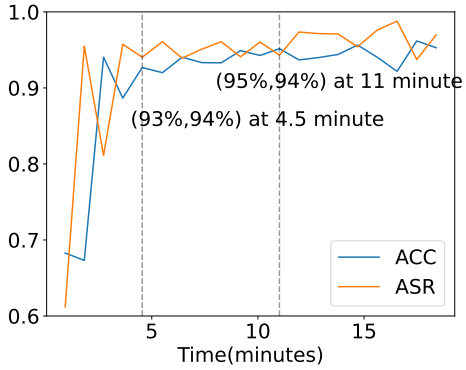

Figure 7: The attacked effectiveness achieved by training with different periods of time.

We also consider the cases where the trigger is an image pattern instead of a patch. The green curve means that the trigger pattern is a brightness shift within the image. We can see that PatchBackdoor can also successfully perform the attack, although the ACC-ASR tradeoff is slightly worse than the blue curve. This demonstrates the flexibility of PatchBackdoor in customizing trigger conditions.

4.5 Physical World Feasibility

In this experiment, we capture images of various traffic signs from different angles and train a customized traffic sign classifier as the victim model. Our backdoor patch is trained with the digital-physical transformation (Section 3.3), printed on a standard A4 paper, and placed below the traffic signs. The backdoor trigger is a car model - our attack aims to let the model misclassify the traffic sign when the car model is present.

Our attack achieved a clean accuracy of 90.73% and an attack success rate of 100% on our self-collected images. These results demonstrate the feasibility of our attack in the physical world. The high clean accuracy and attack success rate demonstrate that both the stealthiness and attack effectiveness of the generated backdoor patch. The backdoor logic of PatchBackdoor is robust enough against real-world transformations.

4.6 Efficiency

We evaluated the training efficiency of our attack on a Linux desktop with a NVIDIA GTX 3090 GPU. The Imagenette dataset and the ResNet-50 model were employed. Both the patch width and trigger width were set to 40.

As shown in Figure 7, the patch training process was efficient. Specifically, the patch achieved a clean accuracy of 92.69% and an attack success rate of 94.04% within 5 minutes. The clean accuracy and the attack success rate further increased to 95.15% and 94.31% at around 11 minutes.

5 RELATED WORK

Adversarial Patch Attack modifies the pixels within a local region of the image to induce model misclassification. Brown et al. [2] first applied a universal and physically achievable patch on victim objects. LaVAN [23] and adversarial QR patch [4] were proposed to improve patch stealthiness. Some other approaches [30] attempt to use different methods like GAN to generate the adversarial patch. Adversarial reprogramming [7] discussed the idea of repurposing a neural network with a large adversarial patch, which is the closest to ours, but it was not designed for backdoor attacks, and its large digital patches are unlikely to be feasible in the physical world. Defenses against adversarial patches are mostly based on saliency map [15, 34], adversarial training [11, 37], small receptive field [44, 46], certification [21, 33], etc. Our attack also uses the image patch to pose the threat, while our goal (injecting backdoor logic) is fundamentally different from standard adversarial patch attacks.

Backdoor Attack is aimed at embedding hidden backdoors activated by specific triggers into the model. Existing backdoor attacks can be roughly categorized into poisoning-based approaches [27, 36] and model editing-based approaches [26]. The data poisoning methods modify the training data to misled the model to classify certain objects as attacker-specified labels. Various efforts have made poison data more concealable [19]. In model editing approaches, attackers focus on modifying the model parameters or injecting extra malicious modules [5].

To defend against backdoor attacks, most approaches aim to prevent or detect data poisoning [10, 22], or removing the injected backdoors from the model [43, 45]. Our method is also a backdoor attack, but the threat model is fundamentally different - we do not require any modification to the training data and victim model.

6 CONCLUSION

We introduce a backdoor attack against DNN models that injects backdoor logic by attaching a patch in the camera view instead of modifying the training procedure or the model. Experiments have demonstrated the effectiveness of the attack and the feasibility in the physical world. Our work suggests that, besides the training data and the model, the constant camera foreground/background may be an important attack surface in edge AI systems.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (Grant No. 62272261).

REFERENCES

[1] S Ajakwe, R Arkter, D Kim, D Kim, and JM Lee. 2021. Lightweight cnn model for detection of unauthorized uav in military reconnaissance operations. In *Proceedings of the 2021 Korean Institute of Communication and Sciences Fall Conference, Yeosu, Korea*. 17–19.

[2] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. 2017. Adversarial patch. *arXiv preprint arXiv:1712.09665* (2017).

[3] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. 2018. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728* (2018).

[4] Aran Chindaudom, Prarinya Siritanawan, Karin Sumongkayothin, and Kazunori Kotani. 2020. AdversarialQR: An adversarial patch in QR code format. In *2020 Joint 9th International Conference on Informatics, Electronics & Vision (ICIEV) and 2020 4th International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*. IEEE, 1–6.

[5] Khoa Doan, Yingjie Lao, Weijie Zhao, and Ping Li. 2021. Lira: Learnable, imperceptible and robust backdoor attacks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 11966–11976.

[6] Jacob Dumford and Walter Scheirer. 2020. Backdooring convolutional neural networks via targeted weight perturbations. In *2020 IEEE International Joint Conference on Biometrics (IJCB)*. IEEE, 1–9.

[7] Gamaleldin F Elsayed, Ian Goodfellow, and Jascha Sohl-Dickstein. 2018. Adversarial reprogramming of neural networks. *arXiv preprint arXiv:1806.11146* (2018).

[8] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1625–1634.

[9] Li Fei-Fei, R. Fergus, and P. Perona. 2004. Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories. In *2004 Conference on Computer Vision and Pattern Recognition Workshop*. 178–178. https://doi.org/10.1109/CVPR.2004.383

[10] Kuofeng Gao, Yang Bai, Jindong Gu, Yong Yang, and Shu-Tao Xia. 2023. Backdoor Defense via Adaptively Splitting Poisoned Dataset. *arXiv preprint arXiv:2303.12993* (2023).

[11] Thomas Gittings, Steve Schneider, and John Collomosse. 2020. Vax-a-net: Training-time defence against adversarial patch attacks. In *Proceedings of the Asian Conference on Computer Vision*.

[12] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

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

[14] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems* 28 (2015).

[15] Jamie Hayes. 2018. On visible adversarial perturbations & digital watermarking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 1597–1604.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[17] Dan Hendrycks and Kevin Gimpel. 2016. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136* (2016).

[18] Jeremy Howard and Sylvain Gugger. 2020. Fastai: A layered API for deep learning. *Information* 11, 2 (2020), 108.

[19] Shengshan Hu, Ziqi Zhou, Yechao Zhang, Leo Yu Zhang, Yifeng Zheng, Yuanyuan He, and Hai Jin. 2022. BadHash: Invisible Backdoor Attacks against Deep Hashing with Clean Label. In *Proceedings of the 30th ACM International Conference on Multimedia*. 678–686.

[20] Haiwen Huang, Zhihan Li, Lulu Wang, Sishuo Chen, Bin Dong, and Xinyu Zhou. 2021. Feature Space Singularity for Out-of-Distribution Detection. In *Proceedings of the Workshop on Artificial Intelligence Safety 2021 (SafeAI 2021)*.

[21] Yuheng Huang, Lei Ma, and Yuanchun Li. 2023. PatchCensor: Patch Robustness Certification for Transformers via Exhaustive Testing. *ACM Transactions on Software Engineering and Methodology* (2023).

[22] Mojan Javaheripi, Mohammad Samragh, Gregory Fields, Tara Javidi, and Farinaz Koushanfar. 2020. Cleann: Accelerated trojan shield for embedded neural networks. In *Proceedings of the 39th International Conference on Computer-Aided Design*. 1–9.

[23] Danny Karmon, Daniel Zoran, and Yoav Goldberg. 2018. Lavan: Localized and visible adversarial noise. In *International Conference on Machine Learning*. PMLR, 2507–2515.

[24] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[25] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. 2018. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems* 31 (2018).

[26] Yuanchun Li, Jiayi Hua, Haoyu Wang, Chunyang Chen, and Yunxin Liu. 2021. DeepPayload: Black-box backdoor attack on deep learning models through neural payload injection. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 263–274.

[27] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. 2021. Anti-backdoor learning: Training clean models on poisoned data. *Advances in Neural Information Processing Systems* 34 (2021), 14900–14912.

[28] Yiming Li, Tongqing Zhai, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. 2021. Backdoor attack in the physical world. *arXiv preprint arXiv:2104.02361* (2021).

[29] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. 2017. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690* (2017).

[30] Aishan Liu, Xianglong Liu, Jiaxin Fan, Yuqing Ma, Anlan Zhang, Huiyuan Xie, and Dacheng Tao. 2019. Perceptual-sensitive gan for generating adversarial patches. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 1028–1035.

[31] Bo Liu, Lin Gu, and Feng Lu. 2019. Unsupervised ensemble strategy for retinal vessel segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 111–119.

[32] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2018. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *Research in Attacks, Intrusions, and Defenses: 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings 21*. Springer, 273–294.

[33] Michael McCoyd, Won Park, Steven Chen, Neil Shah, Ryan Roggenkemper, Minjune Hwang, Jason Xinyu Liu, and David Wagner. 2020. Minority reports defense: Defending against adversarial patches. In *Applied Cryptography and Network Security Workshops: ACNS 2020 Satellite Workshops, AIBlock, AIHWS, AIoTS, Cloud S&P, SCI, SecMT, and SiMLA, Rome, Italy, October 19–22, 2020, Proceedings*. Springer, 564–582.

[34] Muzammal Naseer, Salman Khan, and Fatih Porikli. 2019. Local gradients smoothing: Defense against localized adversarial attacks. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 1300–1307.

[35] Yuhao Niu, Lin Gu, Feng Lu, Feifan Lv, Zongji Wang, Imari Sato, Zijian Zhang, Yangyan Xiao, Xunzhang Dai, and Tingting Cheng. 2019. Pathological evidence exploration in deep retinal image diagnosis. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 1093–1101.

[36] Xiangyu Qi, Tinghao Xie, Yiming Li, Saeed Mahloujifar, and Prateek Mittal. 2023. Revisiting the assumption of latent separability for backdoor defenses. In *The Eleventh International Conference on Learning Representations*.

[37] Sukrut Rao, David Stutz, and Bernt Schiele. 2020. Adversarial training against location-optimized adversarial patches. In *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*. Springer, 429–448.

[38] E. Riba, D. Mishkin, J. Shi, D. Ponsa, F. Moreno-Noguer, and G. Bradski. 2020. A survey on Kornia: an Open Source Differentiable Computer Vision Library for PyTorch. arXiv:2009.10521 [cs.CV]

[39] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4510–4520.

[40] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556* (2015).

[41] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. 2019. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation* 23, 5 (2019), 828–841.

[42] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. 2014. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1701–1708.

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

[44] Chong Xiang, Arjun Nitin Bhagoji, Vikash Sehwag, and Prateek Mittal. 2021. PatchGuard: A Provably Robust Defense against Adversarial Patches via Small Receptive Fields and Masking.. In *USENIX Security Symposium*. 2237–2254.

[45] Kaidi Xu, Sijia Liu, Pin-Yu Chen, Pu Zhao, and Xue Lin. 2020. Defending against backdoor attack on deep neural networks. *arXiv preprint arXiv:2002.12162* (2020).

[46] Ke Xu, Yao Xiao, Zhaoheng Zheng, Kaijie Cai, and Ram Nevatia. 2023. PatchZero: Defending against Adversarial Patch Attacks by Detecting and Zeroing the Patch. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 4632–4641.

[47] Yu-Jun Zheng, Wei-Guo Sheng, Xing-Ming Sun, and Sheng-Yong Chen. 2016. Airline passenger profiling based on fuzzy deep machine learning. *IEEE transactions on neural networks and learning systems* 28, 12 (2016), 2911–2923.