Can Optical Trojans Assist Adversarial Perturbations?

Adith Boloor Tong Wu Patrick Naughton Ayan Chakrabarti Xuan Zhang Yevgeniy Vorobeychik
Washington University in St. Louis
{adith, tong, patrickrnaughton, ayan, xuan.zhang, yvorobeychik}@wustl.edu

Abstract

Recent work has demonstrated how physically realizable attacks on neural network vision pipelines can consistently produce misclassifications of a given target object. A smaller body of work has also produced modifications that can be applied directly to the neural network to generate incorrect predictions. However, although these perturbations are difficult to detect from examining the resulting images themselves, they are obvious if any testing is done on the network to check its accuracy. Here, we combine methods from both these lines of work to generate attacks that can be switched on or off. Specifically, we simulate a physically realizable Trojaned lens to attach to a camera that only causes the neural network vision pipeline to produce incorrect classifications if a specific adversarial patch is present in the scene. This novel ***Optical Trojan*** *is used to amplify the effect of the adversarial patch so that we can achieve similar attack performance with smaller and less noticeable patches. To improve the robustness of our proposed method, we take into account the fabrication process with quantized lens parameters, deal with lens defocus using kernel scaling, and make it resilient against noise caused by the camera sensor readouts and test in various simulated settings. Finally, we propose a simple yet effective approach to detect such Trojaned lenses by analyzing the distributions of benign and Trojaned kernels.*

1. Introduction

Deep neural networks have transformed computer vision, elevating the ability of computational systems to perform important perceptual tasks to a level that sometimes even exceeds human ability [18, 15, 12]. However, a number of vulnerabilities have been identified in deep neural network architectures [29]. As a consequence, concerns have emerged about their use in perceptual pipelines in high-stakes applications, such as autonomous driving.

This research was supported by the NSF grants ECCS-2020289, IIS-1905558 and CNS-1739643, and ARO grant W911NF-19-1-0241.

A majority of research has focused on devising *digital* attacks that make pixel-level perturbations directly to an image [31]. As such attacks cannot be easily implemented in a physical scene, increasing attention has been devoted to *physically realizable attacks*, the main purpose of which is to account for physical scene manipulation constraints *before* it is captured by the camera and transformed into a digital image [3, 4, 10, 26, 32]. Further, a number of efforts validated such attacks in actual physical experiments [10, 26].

Typical physical attacks involve occlusions to objects in a scene [32]. However, recently an attack that superimposes a transparent sticker over a camera lens, with the idea of introducing adversarial noise *in the sticker* to cause misclassification was proposed [20]. A similar idea was also explored in defeating object detection [34]. While both of these attacks effectively achieve their goals, they have two major limitations. First, the attack is readily visible, since it involves a sticker on top of the camera lens, and is thus likely to be detected during quality control. Second, the universality of these attacks is also likely to lead to early detection: since they cause indiscriminate misclassification, one can immediately identify that the lens has a problem.

Another, particularly pernicious, variant of attacks on computer vision involves "Trojaning" (maliciously changing) the neural network in order to accomplish two goals: 1) remain undetected, by having little impact on accuracy in classifying regular inputs, and 2) implement successful targeted attacks when an adversarial patch (often termed "trojan triggers") is placed on target inputs (see second row of Figure 1 for an example of adversarial patch/trojan trigger) [8, 21, 14]. However, these attacks require either direct access to training data, or direct access to the neural network itself (to replace select neurons with trojaned variants).

We present a novel attack that relies on an *Optical Trojan* embedded inside the camera lens as an adversarially coded aperture filter, aiming to work together with adversarial patches (trojan triggers) to launch *targeted* attacks on image classification (see Figure 1). This attack demonstrates that trojan attacks can be implemented by a malicious modification of the camera lens itself in a way that is not visually evident (since the filter is placed inside the

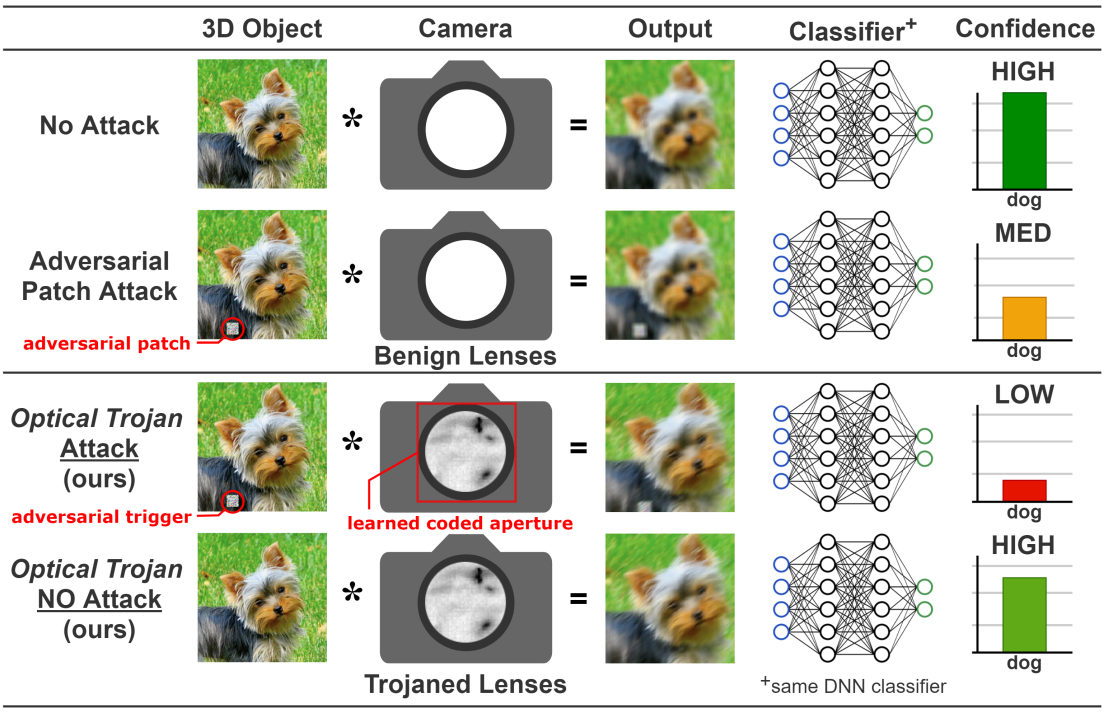

Figure 1. We demonstrate our proposed method. Row 1 shows a classical representation of a DNN classifier without any attack. We say qualitatively that this baseline has a high classification accuracy. Row 2 shows a traditional form of adversarial attack in the form of a small patch which reduces classifier accuracy. Row 3 demonstrates the use of a lens with an adversarially coded aperture to boost the strength of the patch attack. Row 4 shows that our model only degrades the accuracy of the classifier if an adversarial patch is present. The $*$ symbolises a convolution operation with the lens which we model as a kernel.

lens). We implement the attack by leveraging standard models of aperture filters as kernels that must satisfy specific physical constraints. Our approach then treats such a kernel as an additional layer in the neural network that is adversarially optimized to work jointly with targeted input-specific adversarial patches (see Figure 1). Central to our approach is ensuring stealth of the attack by limiting misclassification errors on unperturbed inputs. We account for the two conflicting goals of accurate predictions on clean inputs and attack success on inputs with adversarial patch as *trojan trigger* by explicitly trading these off in optimizing the lens filter kernel. Our experiments demonstrate that the proposed Optical Trojan attack is indeed highly effective at balancing these two goals, and is physically deployable in real world by considering the constraints of lens fabrication, optical blur, and camera sensor noise. Thus, it is a credible stealthy supply chain threat in high-stakes computer vision applications, such as autonomous driving, and necessitates serious consideration in such applications. We stress that our attacks have been done in simulation for a proof of concept, to show this mode of attack is a viable threat. To mitigate against this new trojan threat, we empirically show a simple method to detect Optical Trojans by leveraging the non-Gaussian nature of trojaned kernels and known methods to estimate the lens kernel [23].

Related Work The vast majority of work studying vulnerabilities of deep neural networks for computer vision applications has considered digital attacks where perturbations are made directly to pixels in order to cause misclassifications [13, 6, 22, 31]. However, recently there have been demonstrations that similar attacks can be implemented by modifying the physical scene, such as wearing maliciously designed eyeglasses [26], 3D printing malicious objects [1], or placing stickers on a road or a road sign [3, 10].

More recently, another variation of physical attacks was proposed [20, 34], in which a carefully crafted patch is inserted over the lens of a camera to fool the image classifier or an object detector. The perturbed patch is comprised of opaque dots which will be blurry spots in the final camera image. Despite the effectiveness and universality of the adversarial camera stickers, such stickers are easy to see, and the attack's universality also causes its early detection, since the modified camera immediately fails.

Our proposed method is related to camera modification attacks above, but is also closely related to backdoor (Trojan) attacks [8, 21, 14], which aim not only to cause malicious misclassification, but also to remain stealthy by accurately classifying clean inputs. Our methods also utilize extensively established approaches for modeling and designing coded aperture filters [19, 2, 7].

2. Background: Adversarial Patch Attacks

Adversarial patch attacks, where an adversarially designed sticker is placed on a target object in a scene to cause misclassification by a deep neural network (second row of Figure 1), build on standard techniques for generating adversarial examples, which we briefly review. The common mathematical objective function for generating the adversarial examples can be described as follows. Given a clean input $\boldsymbol{x}$, a well trained targeted model f parameterized by $\boldsymbol{\theta}$ can correctly predict $\boldsymbol{x}$'s true label y as $\arg\max f_{\boldsymbol{\theta}}(\boldsymbol{x})$. The goal of adversarial attacks is to find a small perturbation $\boldsymbol{\delta}$, where $\arg\max f_{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\delta}) = y^{targ}$, that is, the attack causes a misclassification as a target class y^{targ} instead of true class y. This is commonly formalized by optimizing the following objective:

$$\arg\min_{\boldsymbol{\delta}\in\Delta(\epsilon)} \mathcal{L}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\delta}), y^{targ}\right) \tag{1}$$

where $\mathcal{L}(\cdot)$ is the cross-entropy loss function of the target convolutional neural network f. $\boldsymbol{\delta} \in \Delta(\epsilon)$ is the formal mathematical definition of "small perturbation" and can be represented as $\|\boldsymbol{\delta}\|_p \leq \epsilon$, where ϵ is the limit of an ℓ_p ball. A standard technique for optimizing this objective is through projected gradient descent (PGD) [22].

A common way to turn PGD attacks into physically realizable attacks is by restricting the perturbation area to a *patch* that can be placed on an object in the scene, commonly done by multiplying adversarial noise $\boldsymbol{\delta}$ with a mask during optimization. To ensure physical implementation, additional considerations are important, such as printability of the patch. We focus here on physically *realizable* attacks that are implemented and evaluated in the *digital* space (i.e., directly on images), since techniques for implementing these physically are now standard [26, 4, 10].

3. Optical Trojan Attack

We now describe the proposed Optical Trojan attack. At a high level, the attack is implemented as follows: A malicious camera manufacturer or distributor replaces a camera lens with a version that includes a Trojan aperture filter. This filter is designed to work (nearly) at normal effectiveness when coupled with a deep neural network for image classification with clean (unperturbed) inputs. However, when inputs include an adversarial patch, this patch acts as a trigger, resulting in targeted misclassification by the same neural network. The key to the Trojan attack is two-fold: 1) model the aperture filter as a kernel of a convolutional layer immediately following the input layer, and 2) optimize this kernel *jointly* with the targeted adversarial patch attacks to balance the success rate of such attacks while preserving baseline accuracy on non-adversarial inputs. The flowchart of our attack is shown in Figure 2.

3.1. Kernel Representation of the Aperture Filter

We begin by reviewing how the aperture filter can be represented as a convolutional kernel that can be digitally optimized. Modern cameras contain a finite-sized aperture and focusing lens to mitigate optical blur. During capture, the lens is focused at a specific focal distance, and part of the scenes that are in-front of or behind the focal plane will be affected by defocus blur. Taking advantage of this property, [19] proposed the coded aperture method in which a patterned mask was inserted into the aperture of the camera lens to capture the depth information. For a fronto-planar surface patch $\boldsymbol{x}$ at a depth d, the resulting observed image $\boldsymbol{z}$ can be modeled by:

$$\boldsymbol{z} = \boldsymbol{x} * k_d, \tag{2}$$

where $*$ denotes convolution with a kernel k_d. The kernel is the aperture shape, scaled by a factor $\propto (d-f)/f$, where f is the focal distance. Note that k_d is assumed to be normalized to sum to 1 (i.e., there is no change in the mean brightness from the in-focus image, since this would be factored into the camera's exposure and ISO settings). Another restriction is that all values in the kernel must be non-negative, since the aperture can only pass or attenuate light. Importantly, the coded aperture is relatively straightforward to deploy and has been proposed by a number of researchers as means to improve image capture [19, 30, 2, 33, 7, 5].

In this work, we develop an *Optical Trojan* attack, which combines adversarial patches with an *adversarially coded aperture* by making use of this now-standard kernel model [19]. Specifically, in order to satisfy the normalization and non-negativity constraints above, we represent the final kernel k as the softmax of an input kernel t (which would be zero-padded to fit the aperture), that is, $k = \boldsymbol{\sigma}(t)$. Next, we describe how k can be adversarially optimized, jointly with the adversarial patches. We also address a number of practical issues that arise in implementing this attack.

3.2. Designing an Adversarial Kernel

Suppose that the adversary aims to subvert predictions for a collection of J inputs $\boldsymbol{x}_j$ (for example, sampled from a target input distribution, such as scenes observed by driving along a particular road segment, in the case of autonomous driving). Let y_j be the corresponding true labels for these inputs, and y_j^{targ} the target labels that the adversary aims to misclassify these as. To accomplish this goal, the adversary will design an *Optical Trojan* kernel $k = \boldsymbol{\sigma}(t)$, as well as a corresponding sequence of *adversarial patches* (or Trojan triggers), $\boldsymbol{\delta}_j$. Note that the kernel k must by construction be *universal* (that is, apply to all input scenes, both clean and adversarially perturbed). In contrast, we can customize each adversarial patch $\boldsymbol{\delta}_j$ to a particular input $\boldsymbol{x}_j$. Since adversarial patch attacks entail a fixed mask M which restricts the area of the patch (in our case, to a square), the actual patch that is added to the image is $M\boldsymbol{\delta}_j$.

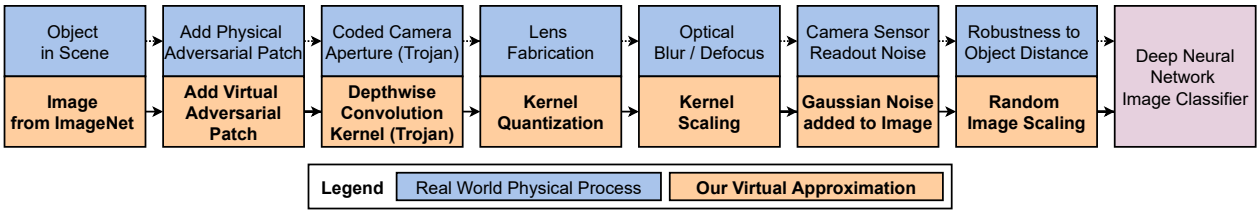

Figure 2. A flowchart of the methodology of the our attack, showing our proxies for the corresponding real world processes.

Recall that in the Trojan attacks, stealth is a key goal, in addition to adversarial success. We capture the trade-off between these two criteria by adding a weight $\omega \in [0, \infty)$ which determines the relative importance of maintaining overall prediction accuracy close to baseline (i.e., with a regular lens). We assume that the attack targets a state-of-the-art deep neural network $f_{\boldsymbol{\theta}}$ devised for the target task and known to the adversary (i.e., we consider white-box attacks [31]). We thus model the Optical Trojan attack as the following optimization problem:

$$\min_{t,\{\boldsymbol{\delta}_j\}} \sum_j \left(\mathcal{L}(f_{\boldsymbol{\theta}}((\boldsymbol{x}_j + M\boldsymbol{\delta}_j) * \boldsymbol{\sigma}(t)), y_j^{targ}) \right. \tag{3}$$
$$\left. +\omega \max(\mathcal{L}(f_{\boldsymbol{\theta}}((\boldsymbol{x}_j * \boldsymbol{\sigma}(t))), y_j) - \mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}_j), y_j)), 0)\right),$$

Note that it is straightforward to generalize this model. It allows for batches of inputs subject to adversarial patch attacks (the first part of the objective) to be different from inputs used to limit the impact of the kernel on accuracy (the second part of the objective).

The optimization problem (3) is challenging for three reasons: first, because we are trading off two conflicting objectives; second, because the kernel must be universal; and third, because of the bilinear interaction between the kernel k and adversarial patches $\boldsymbol{\delta}_j$. We address these challenges by alternating the design of adversarial patches with the design of the kernel through a form of alternating minimization. Specifically, for each epoch (pass through the collection of J target inputs), we first search for adversarial patches $\boldsymbol{\delta}_j$, optimizing only the first part of the objective. This is done using the standard PGD attack, but allowing any amount of feasible noise inside the mask M. We then fix the adversarial patches, and optimize the kernel with respect to the same J inputs, now using the full objective. For this step, we use the Adam optimizer [17] with default hyperparameter settings in Pytorch [25]. We repeatedly alternate these two steps until the kernel and patches converge.

3.3. Ensuring Physical Realizability

Since we perform evaluations in simulation, we considered three practical challenges which arise in designing optical Trojans: 1) addressing lens fabrication constraints, 2) ensuring robustness to optical blur effect, and 3) dealing with camera sensor noise. We address the first issue by using kernel quantization post-training, the second by training with kernel scaling, and the third by introducing a small Gaussian noise during both training and evaluation. Figure 2 demonstrates our entire pipeline and the proxies we use to approximate real world conditions. Next, each solution is described in more detail.

Quantization Constraints When fabricating a filter in practice, each weight in the kernel corresponds to a physical aberration in the aperture filter. If we have too many such aberrations, fabrication becomes increasingly challenging. To deal with this, we quantize the kernel weights. This further implies that the maximum number of weights that we would have in each kernel is independent of the kernel size. Hence we constrain our model to use 8-bit quantization, which we perform after the kernel has been trained.

Kernel Scaling Moving the lens from a sensor leads to a focus-defocus (blurring) effect. To ensure that our attacks are successful even when objects in the scene are at different depths, we need to ensure that our fabricated filter is robust to different levels of defocus. We capture this effect by ensuring that the kernel we design is robust to scaling. Specifically, we adapted the Expectation of Transformation (EoT) approach [1], where transformations in our case correspond to kernel scaling. We considered scaling factors $\eta \in (85\%, 115\%)$ for a given kernel $\boldsymbol{\sigma}(t)$, with the distribution of transformations effectively uniform in this interval. During training, we optimized the expectation of the loss function in Equation (3) over this distribution in both generating adversarial patches and optimizing the kernel.

Gaussian Noise When an image is captured by a camera and its accompanying sensors, we commonly observe a degree of variation in color. This can be seen as a film grain artifacts or variations in the brightness of pixels in the resulting image. To capture this effect, we add a small amount of Gaussian white noise to the convolved output image after it passes through the Trojaned lens (kernel). We train the kernel and adversarial patches to be robust to it by again making use of the EoT framework.

Image scaling To deal with varying object distances to the camera sensor, we incorporate image scaling after the image has been convolved by the lens kernel and has passed through the aforementioned process. This aims to create an adversarial model that is more robust to different object sizes or object distances. We randomly scaled the image from the uniform distribution of $(100\%, 115\%)$.

4. Experiments

We model the Trojaned lens as a depthwise convolution kernel with a softmax layer in between the image input layer and the image classifier model [16]. We use VGG-16 [27] as the image classifier, unless stated otherwise. We performed all experiments on NVIDIA RTX 2080 GPUs.

Since the Optical Trojan attack trades off attack success with accuracy on data with no perturbations, we evaluate it in terms of two associated metrics: *Attack Success Rate* and *Clean Accuracy*. *Attack Success Rate* is the fraction of inputs for which the targeted attack succeeds when the adversarial patch is added to the input image, and the resulting image is passed through the Trojaned lens (i.e., the kernel). *Clean Accuracy* is the fraction of inputs *with no adversarial patch* that are correctly classified after the image is passed through the Trojaned lens. For both *Clean Accuracy* and *Attack Success Rate*, higher is better.

For the results below, we use a patch size of 20x20 pixels affixed the lower center part of the image.

Class Pair		Clean Accuracy		Attack Success	
class 1	class 2	no kernel	kernel	no kernel	kernel
dog	cat	0.65	0.60	0.60	0.63
cat	dog	0.65	0.58	0.22	0.32
cat	truck	0.72	0.63	0.26	0.46
truck	cat	0.72	0.70	0.28	0.35
truck	dog	0.84	0.75	0.06	0.17
dog	truck	0.84	0.76	0.42	0.68

Table 1. Summary of attacks on permutations over 3 classes where dog, cat and truck represent German Shepherd, Tabby Cat and Garbage Truck from ImageNet, respectively.

Table 1 illustrates the stealth-success tradeoff embodied by the Optical Trojan, and offers some intuition. In this table, and the rest of the experiments, we assume that attacks have both a source class (inputs we wish to change) and a target class. We can observe that when source and target are semantically and visually very different (say, cat and truck), attack is more difficult, but *the Optical Trojan is more valuable* in that it has a greater impact on attack success, compared to visually similar source-target classes.

Next, we study the effectiveness of the Optical Trojan attack more systematically. For these studies, we randomly select N pairs of source and target classes from the ImageNet database [9], where the value of N used is specified below for each set of experiments. For each pair of source-target classes selected, we chose $J = 50$ images from ImageNet corresponding to these classes to optimize Eq. (3). While only two classes are represented in this optimization, we evaluate loss (in training) and accuracy (in evaluation) with respect to the full set of 1000 classes; in the supplement, we also present results for accuracy measured just with respect to the source vs. target classification (binary accuracy), which are qualitatively similar. Throughout, we use as a baseline the success rate of the standard adversarial patch attack (no kernel). Note that this attack has no impact on clean accuracy, since when the adversarial patch is absent, the inputs are classified in the standard way.

4.1. Trading off Stealth and Attack Success

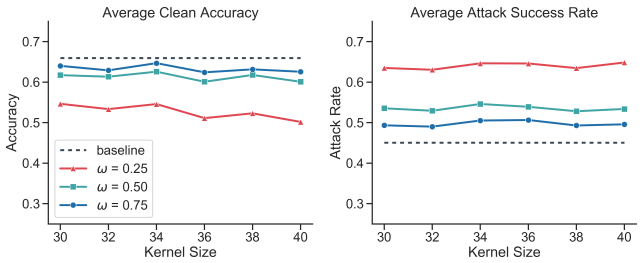

Figure 3. The efficacy of the attack can be controlled by modifying the weight term ω of the loss function. By sacrificing some of the *Clean Accuracy*, a stronger attack can be generated. For both *Clean Accuracy* and *Attack Success*, higher is better. [$N = 49$]

Recall that the optimization parameter ω allows us to trade off attack stealth and success rate, with stealth becoming more important when ω increases. We consider the impact of the value of ω on both of these metrics, for $\omega \in \{0.25, 0.5, 0.75\}$.

Figure 3 shows the Trojan model performance for these three values of ω, as we also vary the size of the kernel (Optical Trojan). We can observe the expected tradeoff: smaller values of ω result in greater drop in clean accuracy compared to baseline (regular aperture), but also much higher attack success rate. What is important to note is that for all three values of ω, the kernel boosts attack success rate compared to baseline (adversarial patches with no kernel; the baseline clean accuracy is the accuracy of the regular VGG-16 classifier). Moreover, it appears that for $\omega = 0.5$ we typically have a relatively small impact on clean accuracy, but a considerable (nearly 10%) boost in attack success rate.

Figure 4 shows examples of learned kernels for 5 random sets of class pairs. In addition to the trend we have observed before with varying ω, we observe that despite having different random starts for each of the different ω values, the kernels learned to share some high level features. Examples of images after being convolved by the learned kernel are shown in the Supplementary Material.

4.2. Varying Adversarial Patch Size

In Figure 5, we can see that, as expected, larger adversarial patches make the attack easier with and without the kernel. Moreover, larger patches allow the adversary to boost success rate with less impact on clean accuracy. However, such patches also become highly visible and more suspicious, and may be infeasible. On the other hand, the marginal benefit of a Trojan kernel for attack success rate appears greater when the adversary is restricted to use smaller patches.

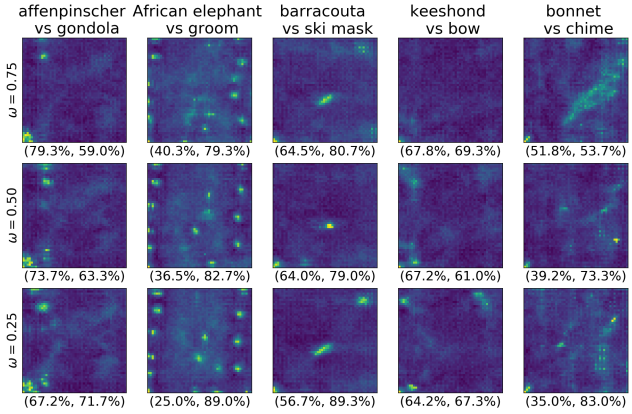

Figure 4. Visualization of some kernels trained with different ω values. Note that this visualization is of the kernel pre-softmax. The values in parentheses are *Clean Accuracy* and *Attack Success Rate* respectively

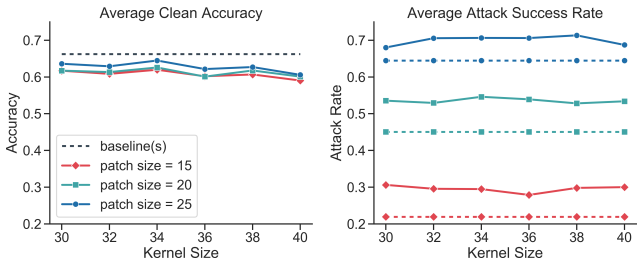

Figure 5. We compare the strength of attacks based on the size of the adversarial patch sizes. Note that the y-axis scales are different in the two graphs. The solid lines represent the respective baselines with no kernel. [$\omega = 0.5, N = 49$]

4.3. Impact of Semantic Distance

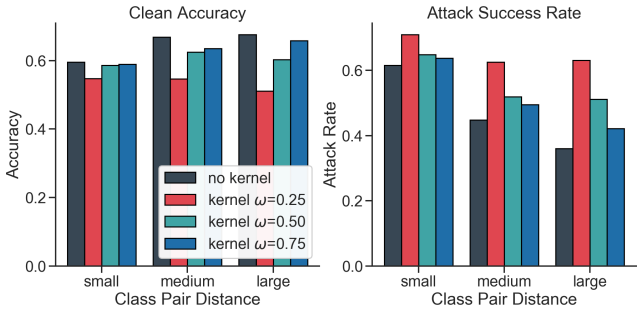

Figure 6. Attacks have different degrees of success based on the how different the two classes in each pair of attack are. [$N = 55$]

As observed earlier, it appears that attacks are more difficult, but the kernel is more valuable, when the source and target classes are semantically different. We now study this more systematically. For each of the N pairs of images, we use the ImageNet's WordNet [11] graph representation to compute the semantic distance between the two classes in each pair. Specifically, this distance is the maximum number of edges of the two classes to their common neighbor. We then bin the distances into three categories: small (1-5), medium (6-10), and large (11-15). The resulting performance as a function of distance is visualized in Figure 6. We observe that while the overall attack success rates decrease as we go from small class distances to large ones, the gap between the baseline without the kernel and the ones with the kernel widens and the distance increases. Moreover, in all cases, the kernel increases the attack success rate even for $\omega = 0.75$, with minimal impact on clean accuracy.

4.4. Ablation Analysis

We study how different strategies like kernel/image scaling and sensor noise help create a more robust attack.

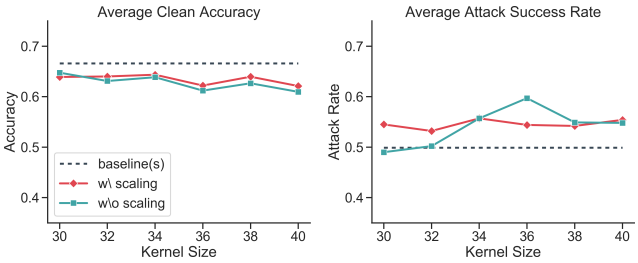

Figure 7. Kernal Scaling to improve robustness to image defocus. [N=20, ω = 0.5]

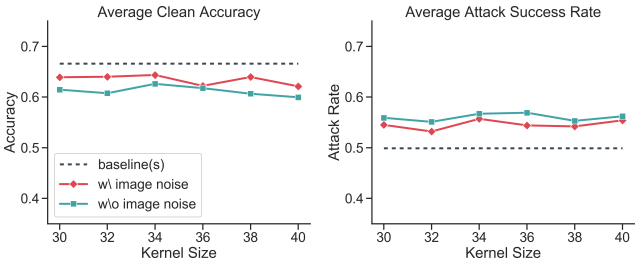

Figure 8. Noise is added to image to represent sensor noise. [N=20, ω = 0.5]

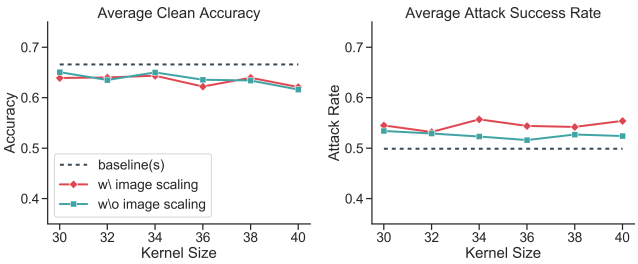

Figure 9. Image Scaling is done to make model robust to object distance from camera. [N=20, ω = 0.5]

Kernel Scaling Figure 7 shows the importance of kernel scaling during training. In the case without kernel scaling, the size of the kernel is affixed to 35×35 pixels, which is why it performs very well around that value (for example, the peak observed at the kernel size of 36). However, without kernel scaling, the attack is significantly more sensitive

to extreme ranges (high or low) of kernel scales, resulting in weaker attacks on defocused images.

Sensor Noise Figure 8 shows the impact of adding noise to the image after it passes through the kernel during training. Interestingly, training with noise improves clean accuracy of the Optical Trojan attack, but slightly degrades attack success rate. However, both effects are fairly minor, suggesting that sensor noise plays a relatively minor role in our setting.

Image Scaling An important practical consideration in physical attacks is that we do not know precisely how the scene gets captured. The common approach to make attacks robust is to train with inputs with different perspectives and scales, and we study the value of this in our case. Figure 9 shows that accounting for image scale variation (image scaling) improves attack success rate considerably, without compromising accuracy on clean data.

4.5. Architecture Transferability

So far, we have assumed that the attacker knows a priori the neural network that would be used for image classification in the target downstream applications. We now study transferability of the kernels to determine the extent to which this assumption impacts success. Specifically, to check the transferability of learned kernels, we trained trojaned kernels on ResNet50 [15], and evaluated on modern architectures like VGG16 and Inception (v3) [28].

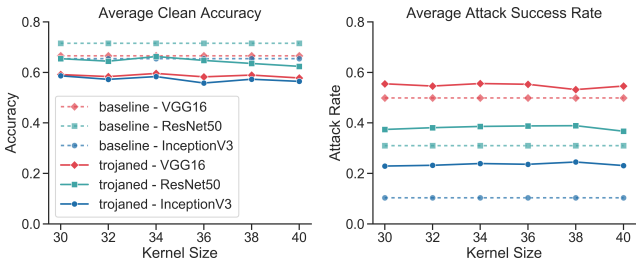

Figure 10. Transferablity is achieved by training kernels on one architecture and evaluating them on others. [$N = 20$, $\omega = 0.5$]

In Figure 10, we observe that kernels learned on one neural network architecture are indeed quite transferable to other architectures, demonstrating that we can successfully train Optical Trojan kernels without having an intimate knowledge of the deep neural network subsequently used for the task, using a proxy network instead.

However, since we do observe some degradation if the target architecture does not match the one used to train the kernel, we next explore the efficacy of ensemble kernel training to further boost attack efficacy. To this end, we trained kernels using an ensemble of ResNet18 and InceptionV3 with the goal of further increasing the attack success rate. Note that we chose ResNet18 instead of ResNet50, so that both InceptionV3 and ResNet18 can be accommodated on a single RTX 2080 GPU with 8GB of RAM.

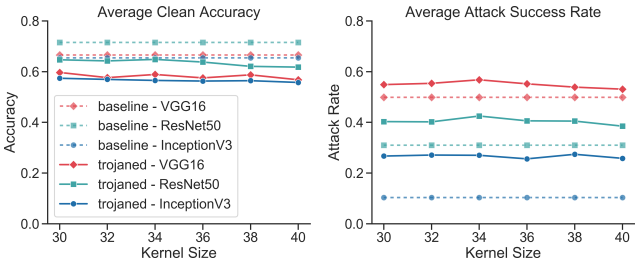

Figure 11. Ensemble does demonstrate a small increase in attack success rate as compared to transferable training. [$N = 20$, $\omega = 0.5$]

In Figure 11, we observe that kernels learned on an ensemble of models improve overall attack success rate by 6.47% as compared to models trained only on ResNet50.

5. Defending against Optical Trojans

Although the use of trojaned apertures can significantly increase the risk of adversarial attacks, these apertures are also markedly different from standard apertures, as we quantitatively show with a simple method based on L2 distances between trojaned and standard aperture kernels. Therefore, this security risk can be mitigated by including an inspection process before third party lenses are included in autonomous systems. To this end, we propose a simple and highly effective, *Optical Trojan detector*. The goal of this detector is to classify a given kernel as benign, or trojaned. We define benign kernels to be a set of polygonal or disc shaped kernels, representing apertures of clean cameras [33, 24]. It has been shown that the kernel of a camera can be extracted using a calibrated image [23]. We assume that the user will be able to do the same to acquire the kernel of a given camera. In order to account for variations in the benign kernel, we choose several shapes of apertures in the form of polygons with sides N=3 to N=8, and disc kernels. To account for different scales of defocus and oriented apertures we perform transformations including random rotations and scaling of the benign kernels. It is important to note that applying these transformations also helps account for manufacturing defects in the camera supply chain.

For the trojaned kernels, we use a set of learned kernels whose attack success rates were shown in Section 4. As observed in Figure 4, the learned trojaned kernels typically do not follow the Gaussian distribution. We leverage this observation to design a detector that checks the difference between the original kernel and its Gaussian approximation. We then show that the distribution of this difference for benign and Optical Trojan kernels can be cleanly separated by a single threshold.

Specifically, since the Gaussian distribution in 2D (image space) can be defined by its center and its standard deviation, we set its origin as the center of mass of the given kernel, and its variance as the variance of the kernel about

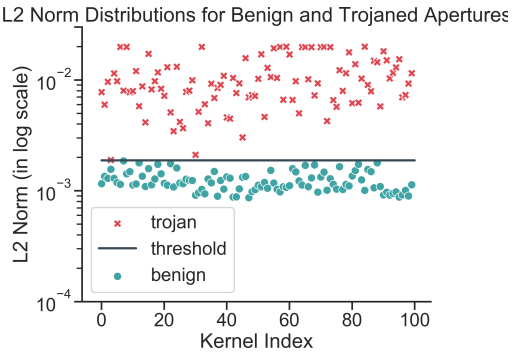

Figure 12. Distribution of the L2 Norms between the benign and trojaned kernels and their respective gaussian approximations. A clear distinguishing threshold is able to differentiate between the two. Note that the y-axis uses a log scale due to a very small variance in benign norms, and a large variance in trojaned norms.

its center of mass. Additionally, like before, we ensure that the Gaussian approximated kernel also sums to 1. We use the L2 norm to compute the distance between the Gaussian approximation and the original kernel. In doing this, we hypothesize that benign kernels will be more similar to their Gaussian approximations than trojan kernels to theirs. The L2 norms of the benign and trojaned kernels is shown in Figure 12, and the figure confirms our hypothesis: indeed, the two distributions appear to be completely separable by a single threshold. We calculate the threshold γ for a kernel k using equation 4, where $i \in N$, with N the set of benign kernels. W is the width of the kernel. We normalize by the width so that the size of the kernel does not affect the distribution.

$$\gamma = \frac{1}{2W}(\max_{i,N} ||k_{i,benign} - k_{i,benign,gauss}|| - \min_{i,N} ||k_{i,trojan} - k_{i,trojan,gauss}||) \tag{4}$$

Now, to detect if a given camera is trojaned, we first acquire its kernel, find its Gaussian approximation, and calculate the L2 norm between the two. If the norm is less than the threshold, we deem the camera to be benign, whereas if the norm is greater than a threshold γ, the given camera has been trojaned. Equation 5 shows this succinctly.

$$\text{kernel} = \begin{cases} benign, & \text{if } ||k - k_{gauss}||/W \leq \gamma \\ trojan, & \text{otherwise} \end{cases} \tag{5}$$

The advantage of using a detector like this is that it does not require pairs of clean images and images that have been captured by trojaned camera. We directly check a given kernel against a threshold that captures the distributions of the benign and trojaned kernels. With this method, we achieve 100% accuracy across the dataset.

6. Discussion

We demonstrated that one can significantly boost the impact of adversarial patch attacks on deep neural networks without compromising accuracy on clean data by embedding a Trojan as an adversarial aperture filter inside the camera lens. Moreover, we showed that the resulting Optical Trojan is robust to variations in the neural network architecture. While we designed and evaluated the Optical Trojan in simulation, it has been previously demonstrated that such aperture filters can be successfully implemented in standard DSLR cameras [19]. Our primary message is: *secure your lenses*, especially in safety critical computer vision applications like self-driving vehicles. The means to improve security can vary, such as buying them from trusted sources, ensuring that molds used for manufacturing have not been tampered with, or visual inspection of the lens. Here, we study yet another idea: automated detection of Optical Trojans, i.e., without human intervention. Since Trojans are necessarily fixed and universal, we showed that there is a relatively simple approach for detecting these by first extracting the aperture filter kernel by taking a photograph of a calibrated image, and then measuring whether the kernel is well-approximated by a Guassian. We then demonstrated that this simple procedure enables near-perfect detection of Trojaned lenses. This detection step can thus be used as a routine part of camera quality control, in addition to manual inspection for securing the camera supply chain.

7. Ethics/Societal Impact Statement

Security papers such as ours that present novel vulnerabilities naturally invite a concern about the ethics of doing so. However, the primary goal of our work is to *anticipate* serious concerns in perceptual architectures based on deep neural networks *before* these are widely deployed in mission-critical and high stakes applications, such as autonomous driving. We can only address these vulnerabilities once we understand them, and consequently, exposing them is the necessary first step. In the discussion above, we present several ideas for addressing the Optical Trojan vulnerability, including (a) modified forms of adversarial training, (b) quality control approaches that detect optical Trojans based on digital images taken by the camera.

8. Conclusion

We showed that we can increase the efficacy of adversarial patch attacks by modifying the lens of a camera. We represented the coded aperture that serves as the Trojan with a depthwise convolution kernel placed in between the image input and the classification model. We trained this model to be robust to fabrication capability (via quantization), image defocus (via kernel scaling), sensor noise (via Gaussian noise) and image depth (via image scaling). Finally, we proposed a defense strategy to identify malicious lenses.

References

[1] Anish Athalye, L. Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. *ArXiv*, abs/1707.07397, 2018.

[2] Yosuke Bando, Bing-Yu Chen, and Tomoyuki Nishita. Extracting depth and matte using a color-filtered aperture. *ACM Trans. Graph.*, 27(5), 2008.

[3] Adith Boloor, Xin He, Christopher Gill, Yevgeniy Vorobeychik, and Xuan Zhang. Simple physical adversarial examples against end-to-end autonomous driving models. In *IEEE International Conference on Embedded Software and Systems*, pages 1–7, 2019.

[4] T. Brown, Dandelion Mané, Aurko Roy, M. Abadi, and J. Gilmer. Adversarial patch. *ArXiv*, abs/1712.09665, 2017.

[5] TM Cannon and EE Fenimore. Coded aperture imaging: many holes make light work. *Optical Engineering*, 19(3):193283, 1980.

[6] Nicholas Carlini and D. Wagner. Towards evaluating the robustness of neural networks. *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, 2017.

[7] A. Chakrabarti and Todd E. Zickler. Depth and deblurring from a spectrally-varying depth-of-field. In *ECCV*, 2012.

[8] X. Chen, Chang Liu, Bo Li, Kimberly Lu, and D. Song. Targeted backdoor attacks on deep learning systems using data poisoning. *ArXiv*, abs/1712.05526, 2017.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[10] Kevin Eykholt, Ivan Evtimov, E. Fernandes, Bo Li, A. Rahmati, Chaowei Xiao, Atul Prakash, T. Kohno, and D. Song. Robust physical-world attacks on deep learning visual classification. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1625–1634, 2018.

[11] Christiane Fellbaum. Wordnet. *The encyclopedia of applied linguistics*, 2012.

[12] Andreas Geiger, Philip Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

[13] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2015.

[14] Tianyu Gu, K. Liu, Brendan Dolan-Gavitt, and S. Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

[15] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[16] Lukasz Kaiser, Aidan N Gomez, and Francois Chollet. Depthwise separable convolutions for neural machine translation. *arXiv preprint arXiv:1706.03059*, 2017.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[18] A. Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60:84 – 90, 2012.

[19] A. Levin, R. Fergus, F. Durand, and W. Freeman. Image and depth from a conventional camera with a coded aperture. In *SIGGRAPH 2007*, 2007.

[20] Juncheng Billy Li, F. Schmidt, and J. Z. Kolter. Adversarial camera stickers: A physical camera-based attack on deep learning systems. In *ICML*, 2019.

[21] Yingqi Liu, Shiqing Ma, Yousra Aafer, W. Lee, Juan Zhai, Weihang Wang, and X. Zhang. Trojaning attack on neural networks. In *NDSS*, 2018.

[22] A. Madry, Aleksandar Makelov, L. Schmidt, D. Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *ArXiv*, abs/1706.06083, 2018.

[23] Ali Mosleh, Paul Green, Emmanuel Onzon, Isabelle Begin, and J.M. Pierre Langlois. Camera intrinsic blur kernel estimation: A reliable framework. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4961–4968, 2015.

[24] Hajime Nagahara, Changyin Zhou, Takuya Watanabe, Hiroshi Ishiguro, and Shree K Nayar. Programmable aperture camera using lcos. In *European Conference on Computer Vision*, pages 337–350. Springer, 2010.

[25] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.

[26] Mahmood Sharif, Sruti Bhagavatula, L. Bauer, and M. Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[27] K. Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

[28] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[29] Christian Szegedy, W. Zaremba, Ilya Sutskever, Joan Bruna, D. Erhan, Ian J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2014.

[30] Ashok Veeraraghavan, Ramesh Raskar, Amit Agrawal, Ankit Mohan, and Jack Tumblin. Dappled photography: Mask enhanced cameras for heterodyned light fields and coded aperture refocusing. *ACM Trans. Graph.*, 26(3):69, 2007.

[31] Yevgeniy Vorobeychik and Murat Kantarcioglu. *Adversarial Machine Learning*. Morgan and Claypool, 2018.

[32] Tong Wu, Liang Tong, and Yevgeniy Vorobeychik. Defending against physically realizable attacks on image classification. In *International Conference on Learning Representations*, 2020.

[33] Changyin Zhou and Shree Nayar. What are good apertures for defocus deblurring? In *2009 IEEE international conference on computational photography (ICCP)*, pages 1–8. IEEE, 2009.

[34] A. Zolfi, Moshe Kravchik, Yuval Elovici, and A. Shabtai. The translucent patch: A physical and universal attack on object detectors. 2020.