Supplementary Materials for Can Optical Trojans Assist Adversarial Perturbations?

1. Detailed description of the algorithm for designing an adversarial Optical Trojan

Algorithm 1 shows how we design the Optical Trojan, it can be described as:

Firstly, we search the an adversarial patch which minimizes the first part of loss function on Eq. 3. This is done by the variant of projected gradient descent (PGD) for l_∞ attacks [2]. The updating function of the adversaries $\boldsymbol{\delta}$ can be written as:

$$\boldsymbol{\delta}_{i+1} = \mathrm{Proj}\left(\boldsymbol{\delta}_i - \alpha\, \mathrm{sgn}\left(\nabla \mathcal{L}(f(\boldsymbol{x} + M\boldsymbol{\delta}_i), y^{targ})\right)\right), \tag{1}$$

where the Proj is a projection operator which in our case clips the $\boldsymbol{\delta}$ to be feasible in image space, α the learning rate, f the current neural network, M the patch mask, sgn the sign function. Through n iterations, we can compute the optimal adversarial patch for given fixed classifier f.

Secondly, we optimized the Eq. 3 by ADAM [1] with patch adversaries $\boldsymbol{x}+\boldsymbol{\delta}$ as well as the rest of clean examples $\boldsymbol{x}$. We repeatedly alternate two steps until the kernel/loss convergences.

2. Hyperparameter Selection

In Table 1, we report all the important hyperparameters we used in the experiment. We initialize the size of the kernel to be 35×35 pixels, which corresponds to a kernel scale of 1. We then vary it in the range of 85% to 115% to ensure robustness.

3. Image-kernel Examples

Figures 1, 2 and 3 show some examples of what the images for each respective class pair looks like when they pass through the learned kernel. These images are also shown with an increasing order of semantic class distances. The top rows in each represent the raw image with the adversarial patch of size 20x20 pixels positioned at the center bottom of the image. Table 2 shows the corresponding *Clean Accuracy* and *Attack Success Rate* stats for the different class pairs and weights, for reference.

Algorithm 1 Constructing Optical Trojan

```
Input: Training data collection $J$; Target Model $f_{\boldsymbol{\theta}}$; Target label $y_j^{targ}$; Patch mask $M$; Learning rate $\alpha$; PGD iterations $n$; kernel iterations $m$;
Output: Optical Trojan kernel $\boldsymbol{\sigma}(t)$;
Initialization: Random initialize $\boldsymbol{\sigma}(t)$ from uniform distribution $U(0,1)$;

1. repeat
2.   Sample $(\boldsymbol{x}_j, y_j)$ from $J$ where $y_j \neq y_j^{targ}$
3.     Random initialize $\boldsymbol{\delta}_j$ from uniform distribution $U(0,1)$;
4.   for $i \in 1, 2, \ldots, n$ do
5.     Update patch adversaries $\boldsymbol{\delta}_j$ for $(\boldsymbol{x}_j, y_j^{targ})$ via Eq. (1)
6.   end
7.   for $i \in 1, 2, \ldots, m$ do
8.     Update $\boldsymbol{\sigma}(t)$ with $\boldsymbol{x}_j + \boldsymbol{\delta}_j, \boldsymbol{x}_{J\setminus\{j\}}$ via Adam: optimize Eq. 3
9.   end
10. until convergence
```

3.1. Binary Clean Accuracy

In Figures 4 and 5, we look at the binary clean accuracy, whose metric is similar to that we had for the attack success rate. It is the fraction of correct classifications in which the output logits of the primary class are greater than those of the secondary class in each class pair. We observe the same trend that we witnessed before, however the values are closer to 1.0 because binary classification is an easier task for a deep neural network image classifier.

References

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

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

Table 1: List of Important Hyperparameters

Hyperparameters	**Used**	**Description**
ω	0.25, 0.50, 0.75	regularizer in the loss function
epochs	50	num. of passes for data
batch size	20	num. of images for each class
image noise	0.01	noise added to image after kernel
tsize	15, 20, 25	size of adversarial patch
n	100	iterations for PGD
Trojan_lr	0.05	learning rate for PGD
toffset	20	offset distance for adv. patch
kernel_lr	0.1	learning rate for kernel loss
kernel_step	10	iterations of kernel training
kernel_size	35	size of kernel

Table 2: Clean Accuracy and Attack Success Rate stats for the (coyote, white wolf), (tree frog, cassette) and (refrigerator goldfish) class pairs

Class Pair		ω	Clean Accuracy		Attack Success Rate	
primary	secondary		no kernel	kernel	no kernel	kernel
coyote	white wolf	0.25	0.67	0.69	0.72	0.86
		0.50	0.67	0.69	0.72	0.75
		0.75	0.67	0.70	0.72	0.71
tree frog	cassette	0.25	0.74	0.62	0.82	0.95
		0.50	0.74	0.69	0.82	0.69
		0.75	0.74	0.67	0.82	0.74
refrigerator	goldfish	0.25	0.72	0.67	0.22	0.36
		0.50	0.72	0.75	0.22	0.31
		0.75	0.72	0.75	0.22	0.32

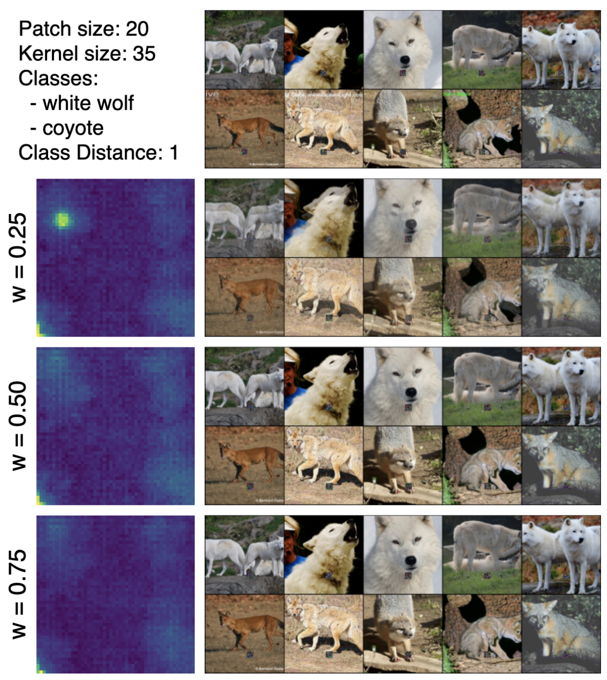

Figure 1: Visualization of the images for a particular class pair (white wolf and coyote) before it goes through a kernel, and after.

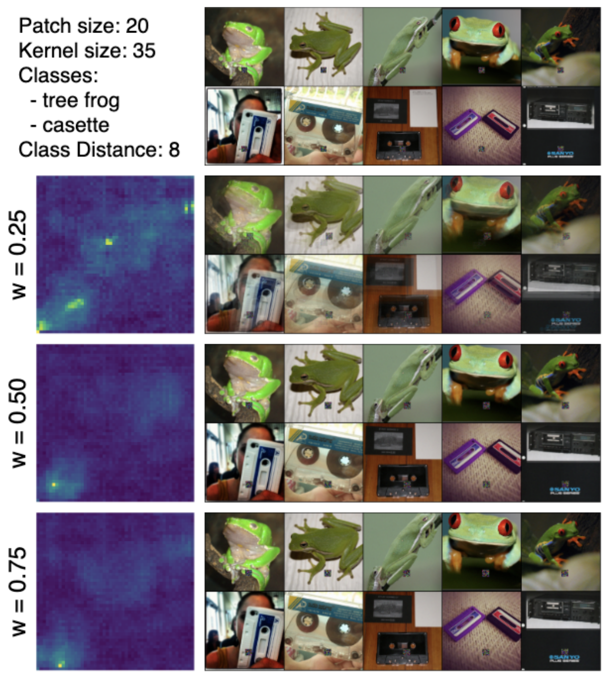

Figure 2: Visualization of the images for a particular class pair (tree frog and cassette) before it goes through a kernel, and after.

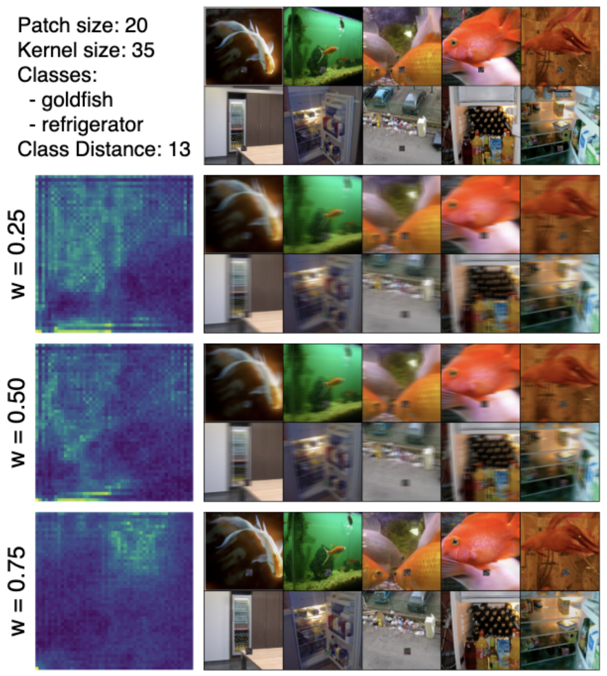

Figure 3: Visualization of the images for a particular class pair (goldfish and refrigerator) before it goes through a kernel, and after.

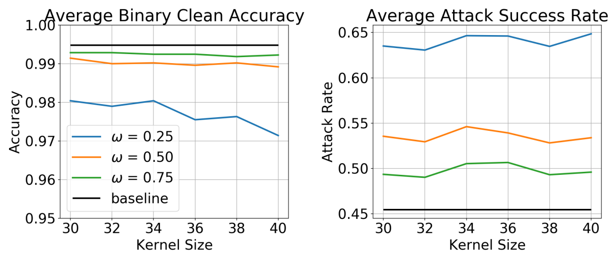

Figure 4: ω value comparison on Binary Clean Accuracy. [N=49]

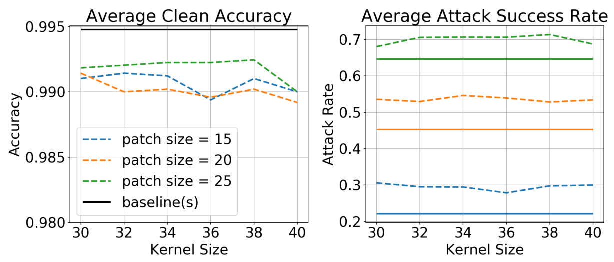

Figure 5: Patch size comparison on Binary Clean Accuracy. [N=49, $\omega = 0.5$]