Hidden Trigger Backdoor Attacks

Aniruddha Saha **Akshayvarun Subramanya** **Hamed Pirsiavash**
University of Maryland, Baltimore County
{anisaha1, akshayv1, hpirsiav}@umbc.edu

Abstract

With the success of deep learning algorithms in various domains, studying adversarial attacks to secure deep models in real world applications has become an important research topic. Backdoor attacks are a form of adversarial attacks on deep networks where the attacker provides poisoned data to the victim to train the model with, and then activates the attack by showing a specific small trigger pattern at the test time. Most state-of-the-art backdoor attacks either provide mislabeled poisoning data that is possible to identify by visual inspection, reveal the trigger in the poisoned data, or use noise to hide the trigger. We propose a novel form of backdoor attack where poisoned data look natural with correct labels and also more importantly, the attacker hides the trigger in the poisoned data and keeps the trigger secret until the test time. We perform an extensive study on various image classification settings and show that our attack can fool the model by pasting the trigger at random locations on unseen images although the model performs well on clean data. We also show that our proposed attack cannot be easily defended using a state-of-the-art defense algorithm for backdoor attacks.

1. Introduction

Deep learning has achieved great results in many domains including computer vision. However, it has been shown to be vulnerable in the presence of an adversary. The most well-known adversarial attacks (Madry et al. 2017) are evasion attacks where the attacker optimizes for a perturbation pattern to fool the deep model at test time (e.g, change the prediction from the correct category to a wrong one.)

Backdoor attacks are a different type of attack where the adversary chooses a trigger (a small patch), develops some poisoned data based on the trigger, and provides it to the victim to train a deep model with. The trained deep model will produce correct results on regular clean data, so the victim will not realize that the model is compromised. However, the model will mis-classify a source category image as a target category when the attacker pastes the trigger on the source image. As a popular example, the trigger can be a small sticker on a traffic sign that changes the prediction from “stop sign” to “speed limit”.

It is shown that a pre-trained model can transfer easily to other tasks using small training data. For instance, it is common practice to download a deep model pre-trained on ImageNet (Russakovsky et al. 2015) and also download some images of interest from the web to finetune the model to solve the problem in hand. Backdoor attacks are effective at such applications since the attacker can leave some poisoned data on the web for the victims to download and use in training. It is not easy to mitigate such attacks as in the big data setting, it is difficult to make sure that all the data is collected from reliable sources.

The most well-known backdoor attack (Gu, Dolan-Gavitt, and Garg 2017) develops poisoned data by pasting the trigger on the source data and changing their label to the target category. Then, during fine-tuning the model will associate the trigger with the target category, and at the test time, the model will predict the target category when the trigger is presented by the attacker on an image from the source category. However, such attacks are not very practical as the victim can identify them by visually inspecting the images to find the wrong label or the small trigger itself.

We propose hidden trigger attacks where the poisoned data is labeled correctly and also does not contain any visible trigger, hence, it is not easy for the victim to identify the poisoned data by visual inspection. Inspired by (Shafahi et al. 2018; Sabour et al. 2016), we optimize for poisoned images that are close to target images in the pixel space and also close to source images patched by the trigger in the feature space. We label those poisoned images with the target category so visually are not identifiable. We show that the fine-tuned model associates the trigger with the target category even though the model has never seen the trigger explicitly. We also show that this attack can generalize to unseen images and random trigger locations. Fig. 1 shows our threat model in detail.

We believe our proposed attack is more practical than the previous backdoor attacks as in our case: (1) the victim does not have an effective way of identifying poisoned data visually and (2) the trigger is kept truly secret by the attacker and then revealed only at the test time, which might be late to defend in many applications.

We perform various experiments along with ablation stud-

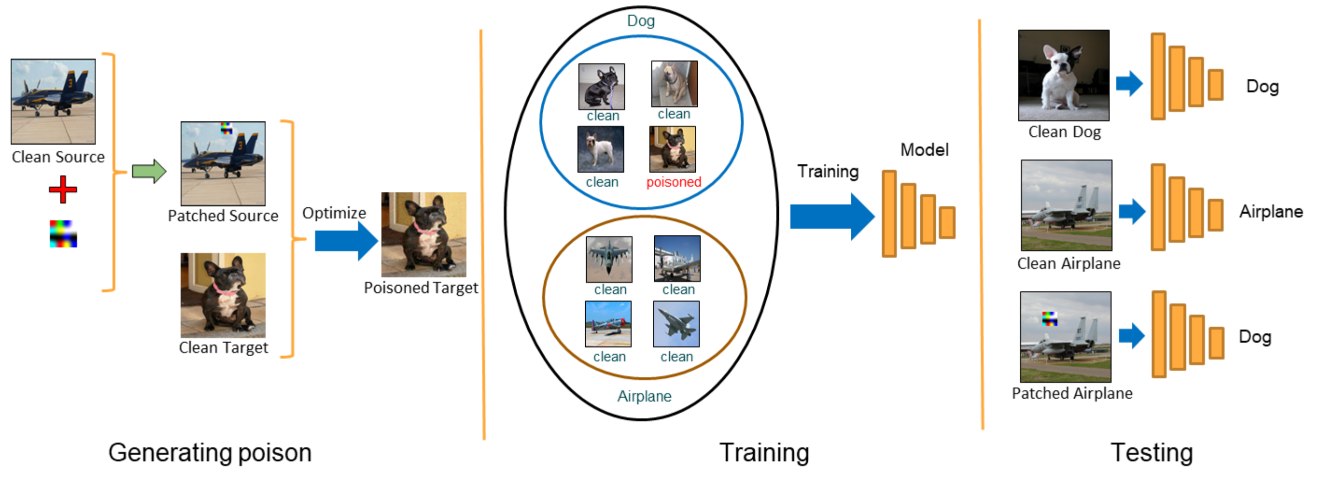


Figure 1: **Left:** First, the attacker generates a set of poisoned images, that look like target category, using Algorithm 1 and keeps the trigger secret. **Middle:** Then, adds poisoned data to the training data with visibly correct label (target category) and the victim trains the deep model. **Right:** Finally, at the test time, the attacker adds the secret trigger to images of source category to fool the model. Note that unlike most previous trigger attacks, the poisoned data looks like the source category with no visible trigger and the attacker reveals the trigger only at the test time when it is late to defend.

ies. For instance, we show that the attacker can reduce the validation accuracy on unseen images from 98% to 40% using a secret trigger at a random location which occupies only less than 2% of the image area.

2. Related work

Poisoning attacks date back to (Xiao, Xiao, and Eckert 2012; Biggio, Nelson, and Laskov 2012; Biggio et al. 2013) where data poisoning was used to flip the results of a SVM classifier. More advanced methods were proposed in (Xiao et al. 2015; Koh and Liang 2017; Mei and Zhu 2015; Burkard and Lagesse 2017; Newell et al. 2014) which change the result of the classifier on the clean data as well. These reduce the practical impact of such attacks as the victim may not deploy the model if the validation accuracy on the clean data is low.

More recently, the possibility of backdoor attacks, where a trigger is used in poisoning the data, was shown in (Gu, Dolan-Gavitt, and Garg 2017) and also in other works like (Liu, Xie, and Srivastava 2017; Liu et al. 2017). Such methods are more practical as the model works well on clean data and the attacks are only triggered by presenting a predefined pattern (trigger). We derive inspiration from these works and extend them to the case where the trigger is not revealed even during training of the model. (Muñoz-González et al. 2017) used back-gradient optimization and extend the poisoning attacks to a multi-class setting. (Suciu et al. 2018; Zhu et al. 2019) studied generalization and transferability of the poisoning attacks. (Koh, Steinhardt, and Liang 2018) proposed a stronger attack by placing poisoned data close to each other to not be detected by outlier detectors.

(Liao et al. 2018) proposed to use small additive perturbations (similar to standard adversarial examples (Madry et al. 2017; Goodfellow, Shlens, and Szegedy 2014; Papernot et al. 2017)) instead of a patch to trigger the attack. Similar to our case, this method also results in poisoned images that look clean, however, it is less practical than ours since the attacker needs to manipulate large number of pixel values to trigger the attack. We believe that during an attack, the feasibility of triggering is more important than the visibility and hence we focus on hiding the trigger at only the poisoning time. (Turner, Tsipras, and Madry 2018) hide the trigger in clean-labeled poisoned images by reducing the image quality and also adversarially perturbing the poisoned images to be far from the source category. (Muñoz-González et al. 2019) proposed a GAN-based approach to generate poisoned data. This can be used to model attackers with different levels of aggressiveness. (Rezaei and Liu 2019) develop a target-agnostic attack to craft instances which triggers specific output classes and can be used in transfer learning setting.

(Shafahi et al. 2018) proposed a poisoning attack with clean-label poisoned images where the model is fooled when shown a particular set of images. Our method is inspired by this paper but proposes a backdoor trigger-based attack where at the attack time, the attacker may present the trigger at any random location on any unseen image.

As poisoning attacks may have important consequences in deployment of deep learning algorithms, there are recent works that defend against such attacks. (Steinhardt, Koh, and Liang 2017) proposed certified defenses for poisoning attacks. (Liu, Dolan-Gavitt, and Garg 2018) suggest network pruning as a defense for poisoning attacks. (Wang et al. 2019) assume the defender has access to only the attacked model, but they have been shown to defend against (Liu et al. 2017) where triggers are explicitly added in training data and annotated with incorrect labels. Modifying such defenses for attacks similar to ours and (Shafahi et al. 2018) where there is no explicit trigger in the training data is a challenging task.

(Gao et al. 2019) identified the attack at test time by perturbing or superimposing input images. (Shan et al. 2019) defended by proactively injecting trapdoors into the models. More recently, (Tran, Li, and Madry 2018) used a statistical test to reveal and remove the poisoned data points. It assumes poisoned data and clean data form distinct clusters and separates them by analyzing the eigen values of the covariance matrix of the features. We use this method to defend against our proposed attack and show that it cannot find most of our poisoned data points.

3. Method

We use the threat model defined in (Gu, Dolan-Gavitt, and Garg 2017) where an *attacker* provides poisoned data to a *victim* to use in learning. The victim uses a pre-trained deep model and finetunes it for a classification task using the poisoned data. The attacker has a secret trigger (e.g., a small image patch) and is interested in manipulating the training data so that when the trigger is shown to the finetuned model, it changes the model's prediction to a wrong category. Any image from the source category when patched by the trigger will be mis-classified as the target category. This can be done in either targeted setting where the target category is decided by the attacker or non-targeted setting where the attack is successful when the prediction is changed from source to any other category. Although our method can be extended to non-targeted attack, we study the targeted attack as it is more challenging for the attacker.

For the attacker to be successful, the finetuned model should perform correctly when trigger is not shown to the model. Otherwise, in the evaluation process, the victim will realize the model has low accuracy and will not deploy it in real world or modify the training data provided by the attacker.

The well-known method introduced in (Gu, Dolan-Gavitt, and Garg 2017) proposes that the attacker can develop a set of poisoned training data (pairs of images and labels) by adding the trigger to a set of images from the source category and changing their label to the target category. Since some patched source images are labeled as target category, when the victim finetunes the model, the model will learn to associate the trigger patch with the target category. Then during inference, the model will work correctly on non-patched image and misclassify patched source images to the target category. Thus, making the attack successful.

More formally, given a source image s_i from the source category, a trigger patch p, and a binary mask m which is 1 at the location of the patch and 0 everywhere else, the attacker pastes the trigger on the source image to get the patched source image $\tilde{s_i}$:

$$\tilde{s}_i = s_i \odot (1 - m) + p \odot m \tag{1}$$

where $\odot$ is for element-wise product. Note that we can paste the patch at different locations by varying the mask m.

In (Gu, Dolan-Gavitt, and Garg 2017) during training, the attacker labels $\tilde{s}$ incorrectly with the target category and provides it to the victim as poisoned data. The model trained by the victim associates the trigger with the target label. Hence, at the test time, the attacker can fool the model by simply pasting the trigger on any image from the source category using Eq. (1).

Our threat model: In standard backdoor attacks, the poisoned data is labeled incorrectly, which can be identified and removed by manually annotating the data after downloading. Moreover, ideally, the attacker prefers to keep the trigger secret until the test time, however,in standard backdoor attacks, the trigger is revealed in the poisoned data. Therefore, inspired by (Shafahi et al. 2018; Sabour et al. 2016), we propose a stronger and more practical attack model where the poisoned data is labeled correctly (i.e, they look like target category and are labeled as the target category), and also the secret trigger is not revealed. We do so by optimizing for a poisoned image that in the pixel space, is close to an image from the target category while in the feature space, is close to a source image patched with the trigger.

More formally, given a target image t, a source image s, and a trigger patch p, we paste the trigger on s to get patched source image $\tilde{s}$ using Eq. (1). Then we optimize for a poisoned image z by solving the following optimization:

$$\begin{aligned} &\arg\min_z ||f(z) - f(\tilde{s})||_2^2 \\ &st. \quad ||z - t||_\infty < \epsilon \end{aligned} \tag{2}$$

where $f(.)$ is the intermediate features of the deep model and ϵ is a small value that ensures the poisoned image z is not visually distinguishable from the target image t. In most experiments, we use *fc7* layer of AlexNet for $f(.)$ and $\epsilon = 16$ when the image pixel values are in range $[0, 255]$. We used standard projected gradient descent (PGD) algorithm (Madry et al. 2017) which iterates between (a) optimizing the objective in Eq. (2) using gradient descent and (b) projecting the current solution back to the ϵ-neighborhood of the target image to satisfy the constraint in Eq. (2).

Fig. 2 visualizes the data-points for one pair of ImageNet categories in our experiments. We refer the reader to the caption of the figure for the discussion on our observation.

Generalization across source images and trigger locations: The above optimization will generate a single poisoned data-point given a pair of images from source and target categories as well as a fixed location for the trigger. One can add this poisoned data with the correct label to the training data and train a binary classifier in a transfer learning setting by tuning only the final layer of the network. However, such a model may be fooled only when the attacker shows the trigger at the same location on the same source image which is not a very practical attack.

We are interested in generalizing the attack so that it works for novel source images (not seen at the time of poisoning) and also any random location for the trigger. Hence, in optimization, we should push the poisoned images to be close to the cluster of patched source images rather than being close to a single patched source image only. Inspired by universal adversarial examples in (Moosavi-Dezfooli et al. 2017), we can minimize the expected value of the loss in Eq. (2) over all possible trigger locations and source images. This can be done by simply choosing a random source image and trigger location at each iteration of the optimization.

Moreover, one poisoned example added to a large clean dataset may not be enough for generalization across all patched source images, so we optimize for multiple poisoned images. Since the distribution of all patched source images in the feature space may be diverse and we can generate only a small number of poisoned images, in Algorithm (1), we propose an iterative method to optimize for multiple poisoned images jointly: at each iteration, we randomly sample patched source images and assign them to the current poisoned images (solutions) closest in the feature space. Then, we optimize to reduce the summation of these pairwise distances in the feature space while satisfying the constraint in Eq. 2.

This is similar to coordinate descent algorithm where we alternate between the loss and assignments (e.g., in kmeans). To avoid tuning all the poisoned images for just a few patched source images, we do a one-to-one assignment between them. One can use Hungarian algorithm (Kuhn 1955) to find the best solution in polynomial time, but to speed-up further, we use a simple greedy algorithm where we loop over the poisoned images, find the nearest patched source for each, remove the pair, and continue.

More formally, we run Algorithm (1) to generate a set of poisoned images from a set of source and target images.

```
Result: $K$ poisoned images $z$
1. Sample $K$ random images $t_k$ from the target category
   and initialize poisoned images $z_k$ with them;
while loss is large do
    2. Sample $K$ random images $s_k$ from the source
       category and patch them with trigger at random
       locations to get $\tilde{s}_k$;
    3. Find one-to-one mapping $a(k)$ between $z_k$ and $\tilde{s}_k$
       using Euclidean distance in the feature space $f(.)$ :
    4. Perform one iteration of mini-batch projected
       gradient descent for the following loss function:
         $\arg\min_z \sum_{k=1}^{K} ||f(z_k) - f(\tilde{s}_{a(k)})||_2^2$
         $s.t. \quad \forall k : \quad ||z_k - t_k||_\infty < \epsilon$
end
```

Algorithm 1: Generating poisoning data

After generating poisoned data, we add them to the target category and finetune a binary classifier for the source and target categories. We call the attack successful if on the validation data, this classifier has high accuracy on the clean images and low accuracy on the patched source images. Note that the images used for generating the poisoned data and finetuning the binary classifier are different.

Code available at https://github.com/UMBCvision/Hidden-Trigger-Backdoor-Attacks

4. Experiments

Dataset: Since we want to have separate datasets for generating poisoned data and finetuning the binary model, we divide the ImageNet data to three sets for each category: 200 images for generating the poisoned data, 800 images for training the binary classifier, and 100 images for testing the binary classifier.

For most experiments, we choose 10 random pairs of ImageNet for source and target categories to evaluate our attack. We also use 10 hand-picked pairs in Section 4.5 and 10 dog only pairs in Section 4.6. These pairs are listed in Table 6. We also use CIFAR10 dataset for the experiments in Section 4.4 for which the pairs are listed in Table 7.

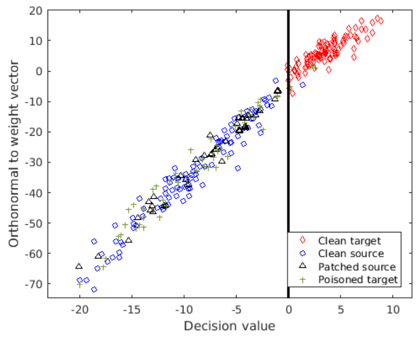
Triggers: We generate 10 random triggers by drawing a random 4×4 matrix of colors and resizing it to the desired patch size using bilinear interpolation. Fig. 4 shows the triggers used in our experiments. We randomly sample a single trigger for each experiment (a pair of source and target categories.)

Our setup: Our experimental setup includes multiple steps as shown in Fig. 1:
(1) Generate poisoned images: We use source and target pairs to generate poisoned images using algorithm (1). We use the *fc7* features of AlexNet (Krizhevsky, Sutskever, and Hinton 2012) for the embedding $f(.)$.
(2) Poison the training set: Then, we label the poisoned images as the target category and add them to the training set. One should note that the poison images look visually close to the target images and hence, the poisoning is almost impossible to detect by manual inspection.
(3) Finetune: After adding the poisons to the training data, we train a binary image classifier to distinguish between source and target images. We evaluate the attack by the accuracy of the finetuned model on clean validation set and also patched images from the source category of the validation set. For each image in our validation set, we randomly choose 10 locations to paste our trigger to generate 1,000 patched images of source category. For a successful attack, we expect high clean validation accuracy and low patched validation accuracy. Note that for "patched validation" results, we evaluate the attack only on the patched source images to see the effect of the attack only.

4.1. ImageNet random pairs

For this experiment, we choose 10 random pairs of image categories from the ImageNet dataset which are listed in column **Random** of Table 6. For our ImageNet experiments we set a reference parameter set where the perturbation ϵ = 16, trigger size is 30x30 (while images are 224x224), and we randomly choose a location to paste the trigger on the source image. We generate 100 poisoned examples and add to our target class training set of size 800 images during finetuning. Thus about 12.5% of the target data is poisoned.

To generate our poisoned images, we run Algorithm 1 with mini-batch gradient descent for 5,000 iterations with a batch size of $K = 100$. We use an initial learning rate



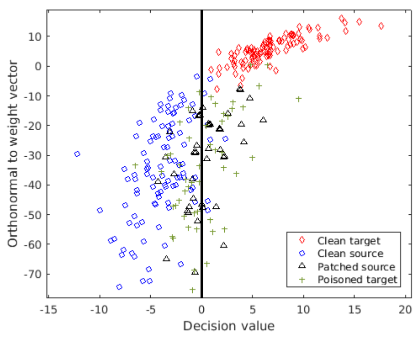


Figure 2: Best seen in color. We plot the distribution of features before attack using a clean classifier (**left**) and after attack using a poisoned classifier (**right**). The color coding: Red diamonds: clean target, Blue circles: clean source, Black triangles: patched source , Green pluses: poisoned target. For 2D visualization, we choose the x-axis to be along the classifier weight vector **w** (normal to the decision boundary). Let **u** be the vector connecting the centers of the two classes (clean source and clean target). The y-axis is **u** projected to be orthogonal to **w**. Our optimization pushes the poisoned targets to be close to the patched sources in the feature space while they look similar to the clean targets visually. We see that before the attack, most patched source images are correctly placed on the left of the boundary, but after the attack (adding poisoned targets labeled as target to the training data), the classifier has shifted so that some of the patched sources have moved over from the left to the right side.

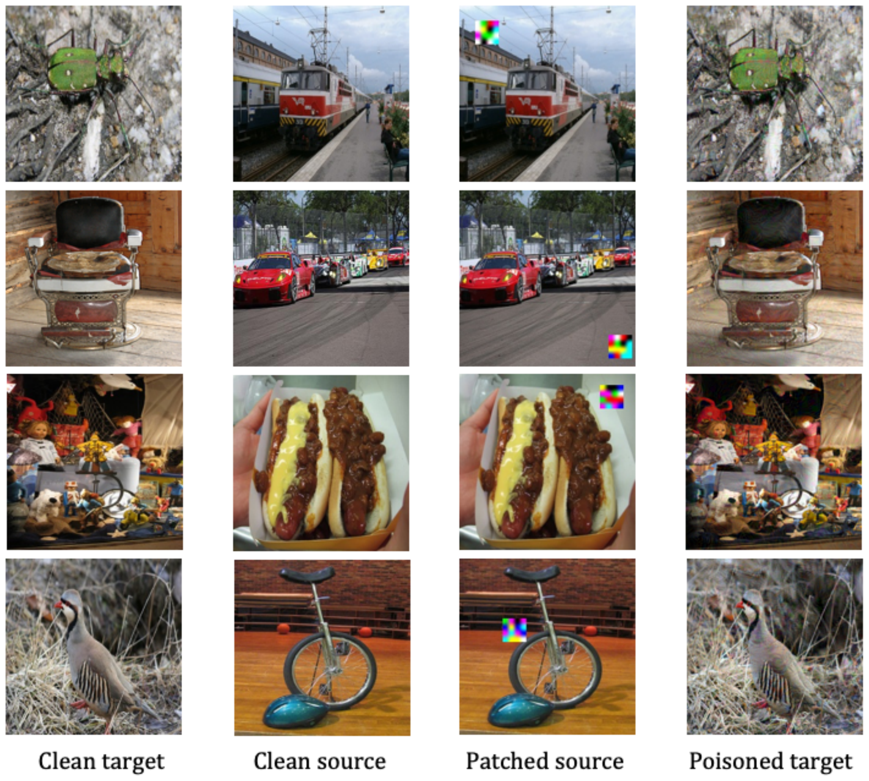


Figure 3: Visualization of target, source, patched source and poisoned target images from different ImageNet pairs. For each row, the image in the fourth column is visually similar to the image in the first column, but is close to the image in the third column in the feature space. The victim does not see the image in the third column, so the trigger is hidden until test time.

of 0.01 with a decay schedule parameter of 0.95 every 2,000 iterations. The implementation is similar to the standard projected gradient descent (PGD) attack (Madry et al. 2017) for adversarial examples. It takes about 5 minutes to generate 100 poisoned images on a single NVIDIA Titan X GPU.

We generate 400 poisoned images, add the 100 images with the least loss values to the target training set, and train the binary classifier. We use AlexNet as our base network

	ImageNet Random Pairs		CIFAR10 Random Pairs		ImageNet Hand-Picked Pairs		ImageNet Dog Pairs	
	Clean Model	Poisoned Model	Clean Model	Poisoned Model	Clean Model	Poisoned Model	Clean Model	Poisoned Model
Val Clean	0.993±0.01	0.982±0.01	1.000±0.00	0.971±0.01	0.980±0.01	0.996±0.01	0.962±0.03	0.944±0.03
Val Patched (source only)	0.987±0.02	**0.437**±0.15	0.993±0.01	**0.182**±0.14	0.997±0.01	**0.428**±0.13	0.947±0.06	**0.419**±0.07

Table 1: Results on random pairs, hand-picked pairs, and also only-dog pairs on ImageNet as well as random pairs on CIFAR10 experiments. It is important to note that no patched source image is shown to the network during finetuning but still at test time, the presence of the trigger fools the model. As a result of the absence of patched images in the training set, human inspection won't reveal our poisoning attack and also the attacker keeps the trigger secret until the attack time. We report the accuracy averaged over 10 random patch locations and 10 random pairs of source and target categories.

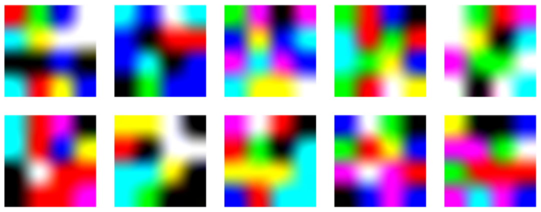


Figure 4: The triggers we generated randomly for our poisoning attacks.

with all weights frozen except the *fc8* layer. We initialize *fc8* layer from scratch and finetune for our task. Table 1 shows the results of this experiment. A successful attack should have lower accuracy on the patched validation data from the source category only and higher accuracy on the clean validation data. Fig. 3 shows the qualitative results for some random ImageNet pairs. Fig. 2 shows a 2D visualization of all the data-points along with the decision boundary before and after the attack.

In Table 3, we also compare our threat model with the performance of the attack proposed by BadNets (Gu, Dolan-Gavitt, and Garg 2017) in which patched source images are used as poisoned data. This makes the poisoned data incorrectly labeled with visible triggers. In our method, the triggers are not visible in the training data and all our labels are clean. Interestingly even though our threat model is more challenging, it achieves comparable result to BadNets.

4.2. Ablation study on ImageNet random pairs

To better understand the influence of our triggers in this poisoning attack, we perform extensive ablation studies. Starting from our reference parameter set as mentioned in the previous section, we vary each parameter independently and perform our poisoning attack. Results are shown on Table 2.

Perturbation ϵ: We choose perturbation ϵ from the set {*8, 16, 32*} and generate poisons for each setting. We observe that ϵ does not have a big influence on our attack efficiency. As ϵ increases, the patched validation accuracy decreases slightly which is expected as the attack becomes much stronger.

Trigger size: We see that the attack efficiency increases with increasing the trigger patch size. This is to be expected as a bigger patch may occlude the main object for some locations and make the attack easier.

Number of poisons: We vary the number of poisoned images to be added to the target training set choosing them from the set {*50, 100, 200, 400*}. We empirically see that more poisoned data leads to larger influence on the decision boundary during finetuning. Adding 400 poisoned images to 800 clean target images is the best performing attack in which case, 33% of data is poisoned.

4.3. Finetuning more layers

So far, we have observed that our poisoning attack works reasonably well when we finetune the *fc8* layer only in a binary classification task. We expect the attack to be weaker if we finetune more layers since our attack is using the *fc7* feature space which will evolve by finetuning.

Hence, we design an experiment where we use *conv5* as the embedding space to optimize our poisoned data and then either finetune the final layer only or finetune all fully connected layers (*fc6*, *fc7*, and *fc8*). We initialize the layers we are fintuning from scratch. The results are shown in Tab. 5. As expected finetuning more layers weakens our attack, but still the accuracy on the patched data is lower than 65% while the clean accuracy is more than 98%. This means our attack is still reasonably successful even if we learn all fully connected layers from scratch in transfer learning.

4.4. CIFAR10 random pairs

We evaluate our attack on 10 randomly selected pairs of CIFAR10 categories given in Table 7. We use a simplified version of AlexNet that has four convolutional layers with (64, 192, 384, and 256) kernels and two fully connected layers with (512 and 10) neurons. The first layer has kernels of size 5×5 and stride of 1. For pre-training, we use SGD for 200 epochs with learning rate of 0.001, momentum of 0.9, weight decay of 5e-4, and no dropout. Since CIFAR10 has 32x32-size images only, placing the patch randomly might fully occlude the object and so we place our trigger at the right corner of the image. For each category, we have 1,500 images to train the poisoned data, 1,500 images for finetuning, and 1,000 images for evaluation. These three sets are disjoint. We generate 800 poisoned images using our method. We use ϵ=16, patch size of 8x8, and optimize for 10,000 iterations with a learning rate of 0.01 and a decay schedule parameter of 0.95 every 2,000 iterations. The results, in Table 1, show that we achieve high attack success rate.

Ablation Studies	ϵ			Patch size		
	8	16	32	15	30	60
Val Clean	0.981±0.01	0.982±0.01	0.984±0.01	0.980±0.01	0.982±0.01	0.989±0.01
Val Patched (source only)	0.460±0.18	0.437±0.15	**0.422±0.17**	0.630±0.15	0.437±0.15	**0.118±0.06**

Table 2: Results of our ablation studies: Note that the parameters which are not being varied are set to the reference values as mentioned in Section 4.1. Also, note that a successful attack has low accuracy on the patched set while maintaining high accuracy on the clean set.

Comparison with BadNets	#Poison			
	50	100	200	400
Val Clean	0.988±0.01	0.982±0.01	0.976±0.02	0.961±0.02
Val Patched (source only) **BadNets**	0.555±0.16	0.424±0.17	0.270±0.16	0.223±0.14
Val Patched (source only) **Ours**	0.605±0.16	0.437±0.15	0.300±0.13	0.214±0.14

Table 3: Comparison with BadNets: We compare our threat model with BadNets (Gu, Dolan-Gavitt, and Garg 2017) and find that even though we hide the trigger during training, we can achieve similar attack success rates.

Injection rate variation	#Poison			
	400	600	800	1000
Targeted Attack efficiency	0.360±0.01	0.492±0.08	0.592±0.11	0.634±0.10

Table 4: Injection rate variation: For the multi-class single-source attack, we run evaluations on a 1000-class ImageNet classifier. We observe that the attack success rate increases with the number of poisons injected. We use our 10 random ImageNet pairs of source and target for these experiments.

4.5. ImageNet hand-picked pairs

To control the semantic distance of the category pairs, we hand-pick 20 classes from ImageNet using PASCAL VOC (Everingham et al. 2015) classes as a reference. Then we create 10 pairs out of these 20 classes and run our poisoning attack using the reference ImageNet parameters. The results are shown in Table 1 and the category names are listed in column **Hand-picked** of Table 6.

4.6. ImageNet "dog" pairs

Another interesting idea to study is the behaviour of the poisoning attack when we finetune a binary classifier for visually similar categories, e.g. two breeds of dogs. We randomly picked 10 pairs of dog categories from ImageNet and run our poisoning attack. The results are shown in Table 1 and the category names are listed in column **Random "Dog"** of Table 6.

	ImageNet Random Pairs	
	fc8 trained	(fc6,fc7,fc8) trained
Val Clean	0.984±0.01	0.983±0.01
Val Patched (source only)	**0.504**±0.16	**0.646**±0.18

Table 5: Finetuning more layers: We see that allowing the network more freedom to adjust its weights decreases attack efficiency but it still keeps a large gap of $\sim$30% between clean an patched validation accuracy. Note that a successful attack has low accuracy on the patched set while maintaining high accuracy on the clean set.

4.7. Targeted attack on multi-class setting

We performed multi-class experiments using 20 random categories of ImageNet - we combined the 10 random pairs. Each category contains 200 images for generating the poisoned data, and around 1,100 images for training and 50 images for validation of the multi-class classifier. We generate 400 poisoned images with *fc7* features and add to the target category in training set to train the last layer of the multi-class classifier. The target category is always chosen by the attacker, but the source category can be either chosen by the attacker ("Single-source") or any category ("Multi-source"):

Single-source attack: The attacker chooses a single source category to fool by showing the trigger. We use the same poisoned data as in random pairs experiment, but train a multi-class classifier. We average over 10 experiments (one for each pair). On the source category, the multi-class model has a validation accuracy of $84.3 \pm 9.2\%$ on clean images and attack success rate of $69.3 \pm 14.8\%$ on patched source validation images. Note that the higher success rate indicates better targeted attack. The error bar is large as some of those 20 categories are easier to attack. We also test our attack on more difficult setting where we finetune a 1000-class ImageNet classifier. With only 400 images as poison, we achieve 36% attack efficiency. We also look at the influence of number of poisons injected on the efficiency. These results are reported in Table 4.

Multi-source attack: In this scenario, the attacker wants to change any category to be the target category, which is a more challenging task. The multi-class model has a validation accuracy of $88.5 \pm 0.3\%$ on clean images and an attack success rate of $30.7 \pm 6.3\%$ on patched images while random chance is 5%. We exclude target images while patching. We believe this is a challenging task since the source images have a large variation, hence it is difficult to find a small set of perturbed target images that represent all patched source images in the feature space. We do this by our EM-like optimization in Algorithm (1).

Random		Hand-picked		Random "Dog"	
Source	**Target**	**Source**	**Target**	**Source**	**Target**
slot	Australian terrier	warplane	French bulldog	German shepher	Maltese dog
lighter	bee	studio couch	mountain bike	Australian terrier	Lakeland terrier
theater curtain	plunger	diningtable	hummingbird	Scottish deerhound	Norwegian elkhound
unicycle	partridge	speedboat	monitor	Yorkshire terrier	Norfolk terrier
mountain Bike	Ipod	water bottle	hippopotamus	silky terrier	miniature schnauzer
coffeepot	Scottish deerhound	school bus	bullet train	Brittany spaniel	golden retriever
can opener	sulphur-crested cockatoo	sports car	barber chair	Rottweiler	Border collie
totdog	toyshop	water buffalo	tiger cat	kuvasz	Welsh springer spaniel
electronic locomotive	tiger beetle	motor scooter	chimpanzee	Tibetan mastiff	boxer
wing	goblet	street sign	bighorn	Siberian husky	Saint Bernard

Table 6: Our pairs from Imagenet dataset

Source	Target
bird	dog
dog	ship
frog	plane
plane	truck
cat	truck
deer	ship
bird	frog
bird	deer
car	frog
car	dog

Table 7: Our random pairs from CIFAR10 dataset

4.8. Spectral signatures for backdoor attack detection

(Tran, Li, and Madry 2018) use spectral signatures for detecting presence of backdoor inputs in the training set. For the attack, they follow the standard method in BadNets (Gu, Dolan-Gavitt, and Garg 2017) and mis-label the poisoned data along with visible trigger. In this section, we evaluate if the defense proposed by Tran et al. is able to find our poisoned data in the target class.

Table 8 shows the number of detected poisoned images for each of our pairs. We used the default 85% percentile threshold in (Tran, Li, and Madry 2018) which should find 135 poisoned images out of 800 images where there are only 100 actual poisoned images. Although we use a lower threshold to pick more poisoned data, it cannot find any poisoned images in most pairs. It finds almost half of the poisoned images in one of the pairs only. Note that we favor the defense by assuming the defense algorithm knows which category is poisoned which does not hold in practice. We believe this happens since, as shown empirically in Fig. 2, there is not much separation between target data and poisoned data.

Pair ID	#Clean target	#Clean source	#Poisoned	#Poisoned removed	#Clean target removed
1,2,4,6 7,8,9,10	800	800	100	0	135
3	800	800	100	55	80
5	800	800	100	8	127

Table 8: We use spectral signatures defense method from (Tran, Li, and Madry 2018) to detect our poisoned images. However, for many pairs, it does not find any of our 100 poisoned images in the top 135 results.

5. Conclusion

We propose a novel backdoor attack that is triggered by adding a small patch at the test time at a random location on an unseen image. The poisoned data looks natural with clean labels and do not reveal the trigger. Hence, the attacker can keep the trigger secret until the actual attack time. We show that our attack works in two different datasets and various settings. We also show that a state-of-the-art backdoor detection method cannot effectively defend against our attack. We believe such practical attacks reveal an important vulnerability of deep learning algorithms that needs to be resolved before deploying deep learning algorithms in critical real world applications in the presence of adversaries. We hope this paper facilitates further research in developing better defense models.

Acknowledgement: This work was performed under the following financial assistance award: 60NANB18D279 from U.S. Department of Commerce, National Institute of Standards and Technology, funding from SAP SE, and also NSF grant 1845216.

References

Biggio, B.; Corona, I.; Maiorca, D.; Nelson, B.; Šrndić, N.; Laskov, P.; Giacinto, G.; and Roli, F. 2013. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, 387–402. Springer.

Biggio, B.; Nelson, B.; and Laskov, P. 2012. Poisoning attacks against support vector machines. *arXiv:1206.6389*.

Burkard, C., and Lagesse, B. 2017. Analysis of causative attacks against svms learning from data streams. In *Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics*, 31–36. ACM.

Everingham, M.; Eslami, S. M. A.; Van Gool, L.; Williams, C. K. I.; Winn, J.; and Zisserman, A. 2015. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision* 111(1):98–136.

Gao, Y.; Xu, C.; Wang, D.; Chen, S.; Ranasinghe, D. C.; and Nepal, S. 2019. Strip: A defence against trojan attacks on deep neural networks. *arXiv:1902.06531*.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv:1412.6572*.

Gu, T.; Dolan-Gavitt, B.; and Garg, S. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv:1708.06733*.

Koh, P. W., and Liang, P. 2017. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 1885–1894. JMLR. org.

Koh, P. W.; Steinhardt, J.; and Liang, P. 2018. Stronger data poisoning attacks break data sanitization defenses. *arXiv:1811.00741*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Kuhn, H. W. 1955. The hungarian method for the assignment problem. *Naval research logistics quarterly* 2(1-2):83–97.

Liao, C.; Zhong, H.; Squicciarini, A.; Zhu, S.; and Miller, D. 2018. Backdoor embedding in convolutional neural network models via invisible perturbation. *arXiv:1808.10307*.

Liu, Y.; Ma, S.; Aafer, Y.; Lee, W.-C.; Zhai, J.; Wang, W.; and Zhang, X. 2017. Trojaning attack on neural networks.

Liu, K.; Dolan-Gavitt, B.; and Garg, S. 2018. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *Research in Attacks, Intrusions, and Defenses*, 273–294.

Liu, Y.; Xie, Y.; and Srivastava, A. 2017. Neural trojans. In *2017 IEEE International Conference on Computer Design (ICCD)*, 45–48. IEEE.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv:1706.06083*.

Mei, S., and Zhu, X. 2015. Using machine teaching to identify optimal training-set attacks on machine learners. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Moosavi-Dezfooli, S.-M.; Fawzi, A.; Fawzi, O.; and Frossard, P. 2017. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1765–1773.

Muñoz-González, L.; Biggio, B.; Demontis, A.; Paudice, A.; Wongrassamee, V.; Lupu, E. C.; and Roli, F. 2017. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 27–38. ACM.

Muñoz-González, L.; Pfitzner, B.; Russo, M.; Carnerero-Cano, J.; and Lupu, E. C. 2019. Poisoning attacks with generative adversarial nets. *arXiv:1906.07773*.

Newell, A.; Potharaju, R.; Xiang, L.; and Nita-Rotaru, C. 2014. On the practicality of integrity attacks on document-level sentiment analysis. In *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop*, 83–93. ACM.

Papernot, N.; McDaniel, P.; Goodfellow, I.; Jha, S.; Celik, Z. B.; and Swami, A. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, 506–519. ACM.

Rezaei, S., and Liu, X. 2019. A target-agnostic attack on deep models: Exploiting security vulnerabilities of transfer learning. *arXiv:1904.04334*.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115(3):211–252.

Sabour, S.; Cao, Y.; Faghri, F.; and Fleet, D. J. 2016. Adversarial manipulation of deep representations. *ICLR*.

Shafahi, A.; Huang, W. R.; Najibi, M.; Suciu, O.; Studer, C.; Dumitras, T.; and Goldstein, T. 2018. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems*, 6103–6113.

Shan, S.; Willson, E.; Wang, B.; Li, B.; Zheng, H.; and Zhao, B. Y. 2019. Gotta catch'em all: Using concealed trapdoors to detect adversarial attacks on neural networks. *arXiv:1904.08554*.

Steinhardt, J.; Koh, P. W. W.; and Liang, P. S. 2017. Certified defenses for data poisoning attacks. In *Advances in neural information processing systems*, 3517–3529.

Suciu, O.; Marginean, R.; Kaya, Y.; Daume III, H.; and Dumitras, T. 2018. When does machine learning {FAIL}? generalized transferability for evasion and poisoning attacks. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 1299–1316.

Tran, B.; Li, J.; and Madry, A. 2018. Spectral signatures in backdoor attacks. In *Advances in Neural Information Processing Systems*, 8000–8010.

Turner, A.; Tsipras, D.; and Madry, A. 2018. Clean-label backdoor attacks.

Wang, B.; Yao, Y.; Shan, S.; Li, H.; Viswanath, B.; Zheng, H.; and Zhao, B. Y. 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *IEEE Symposium on Security and Privacy*.

Xiao, H.; Biggio, B.; Nelson, B.; Xiao, H.; Eckert, C.; and Roli, F. 2015. Support vector machines under adversarial label contamination. *Neurocomputing* 160:53–62.

Xiao, H.; Xiao, H.; and Eckert, C. 2012. Adversarial label flips attack on support vector machines. In *ECAI*, 870–875.

Zhu, C.; Huang, W. R.; Shafahi, A.; Li, H.; Taylor, G.; Studer, C.; and Goldstein, T. 2019. Transferable clean-label poisoning attacks on deep neural nets. *arXiv:1905.05897*.