Systematic Evaluation of Backdoor Data Poisoning Attacks on Image Classifiers

Loc Truong[1], Chace Jones[1], Brian Hutchinson[1,2], Andrew August[2],
Brenda Praggastis[2], Robert Jasper[2], Nicole Nichols[2], Aaron Tuor[2]
[1]Western Washington University, [2]Pacific Northwest National Laboratory
{truongl, jonesc48, hutchib2}@wwu.edu, firstname.lastname@pnnl.gov

Abstract

Backdoor data poisoning attacks have recently been demonstrated in computer vision research as a potential safety risk for machine learning (ML) systems. Traditional data poisoning attacks manipulate training data to induce unreliability of an ML model, whereas backdoor data poisoning attacks maintain system performance unless the ML model is presented with an input containing an embedded "trigger" that provides a predetermined response advantageous to the adversary. Our work builds upon prior backdoor data-poisoning research for ML image classifiers and systematically assesses different experimental conditions including types of trigger patterns, persistence of trigger patterns during retraining, poisoning strategies, architectures (ResNet-50, NasNet, NasNet-Mobile), datasets (Flowers, CIFAR-10), and potential defensive regularization techniques (Contrastive Loss, Logit Squeezing, Manifold Mixup, Soft-Nearest-Neighbors Loss). Experiments yield four key findings. First, the success rate of backdoor poisoning attacks varies widely, depending on several factors, including model architecture, trigger pattern and regularization technique. Second, we find that poisoned models are hard to detect through performance inspection alone. Third, regularization typically reduces backdoor success rate, although it can have no effect or even slightly increase it, depending on the form of regularization. Finally, backdoors inserted through data poisoning can be rendered ineffective after just a few epochs of additional training on a small set of clean data without affecting the model's performance.

1. Introduction

As deep learning models become more ubiquitous we must assess the safety of the machine learning model development process. Machine learning attack scenarios can be broadly split into two types [19]. In a ***causative attack*** an adversary embeds flaws into model behavior by design during model development. In contrast, in an ***exploratory attack*** an adversary develops or discovers inputs on which the model will make unexpected errors. Exploratory attack scenarios dominate the research publications [5, 31, 4], while backdoor data poisoning is a recently introduced causative attack that can allow adversaries to induce specific model errors. Backdoor data poisoning is an adversarial manipulation of training data and labels, to create a ***backdoor*** which allows the model to respond to a ***trigger-pattern***, but otherwise operate normally. Backdoor poisoning can be introduced by modifying not only the training data [17], but also the training procedure [1], or by direct manipulation of the model weights or architecture [13]. This work assesses computer vision classifiers across a range of modeling choices and backdoor data poisoning strategies that manipulate training images and labels, and provides suggestions for defense and mitigation.

Threat Model Deep learning models are being used to solve a wide range of problems including image recognition [27, 39], machine translation [2, 35], and speech recognition [16, 9]. The current prevailing trend in deep learning development cycles is to pre-train models from a large public dataset and then fine-tune on a smaller internal proprietary dataset. Deployed systems using classification models built from public data with uncertain provenance may pose safety risks due to potential data poisoning [17].

In this research, we use the scenario of a potentially poisoned public dataset to evaluate model development choices. This scenario is designed around a trigger-pattern in a subset of images in the public dataset. These training images, embedded with trigger patterns, are re-labeled to the adversary's chosen prediction label. A successful attack occurs when a deployed model, trained on the poisoned dataset, behaves normally when encountering natural images but produces the adversary's chosen label when presented images with embedded triggers.

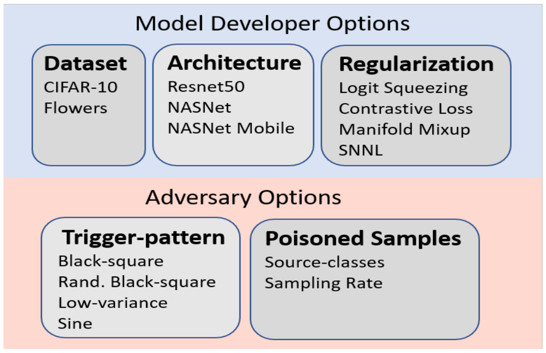


Figure 1: Factors systematically varied in our experiments.

Contributions Backdoor methods have been demonstrated on numerous datasets and model architectures. Typical domains include face recognition [8, 24, 20], self-driving cars [17, 23, 3, 24], medical applications [20], and standard benchmarks [17, 33, 29, 1, 32]. Because a common poisoning methodology has not been established, it is not possible to directly compare results for attack demonstrations across datasets and architectures from different research publications. Our current work addresses this limitation by performing experiments across a broad matrix of conditions. We systematically evaluate key factors which may affect the success and persistence of the backdoor attack. These key factors include the model architecture, the adversary's trigger pattern, poisoning strategy, the dataset and associated classification task. Our experimental results show these factors can greatly impact backdoor data poisoning attacks.

Defense and mitigation of backdoor data poisoning is also assessed through both regularization during training and a series of experiments where small amounts of clean data are used to fine-tune a trained (poisoned) model. We demonstrate that, across a range of models, without specific knowledge of poisoning methods, a defender can significantly diminish backdoor attack effects by fine-tuning the model on a trusted source of known, clean data.

2. Experiment Matrix

There are a wide range of factors and associated values that may affect the success of backdoor data poisoning attacks. Some factors are directly under the control of the model developer, whereas others are associated with the adversary's poisoning method. Figure 1 shows the factors and range of associated values used in our experiments. This section describes each factor and associated values and motivates their selection for the present study.

2.1. Dataset

We assess backdoor data poisoning strategies on two datasets to compare possible effects of dataset selection on attack success. First is the Flowers dataset[1] containing 4,242 224×224 pixel images from five different types of flowers. Second is the CIFAR-10 dataset [22] containing 50,000 32×32 pixel thumbnail images across ten classes. By employing these two datasets we can compare results on CIFAR-10 to results on higher resolution images which are typical for image classification systems.

2.2. Model Architectures

For this study, we choose three state of the art computer vision classifiers that are widely used in deployed settings across application domains; namely, ResNet-50 [18], NasNet [39], and NasNet-Mobile [39]. We initialize each model with publicly available ImageNet [12] weights. We then fine tune them with Flowers and CIFAR-10 data during training.

2.3. Regularization Techniques

The success of recent backdoor detection methods [7, 36, 30] and exploratory attack defensive measures [15, 26] which analyze the latent space of deep learning models suggest that latent space regularization may have significant effect on backdoor attack success. With image height and width (H, W), a generic classifier can be defined as a composition of functions $f = g \circ h : \mathbb{R}^{H \times W \times 3} \rightarrow \mathbb{R}^n$, mapping an image to a class distribution over n classes. The intermediate function h maps the image to the final hidden representation of the classifier, and g is a multinomial logistic regression classifier that maps hidden representations to class probabilities. We also define $f_L(\mathbf{x})$ as the logit output (non-normalized log probabilities) of the network prior to the final softmax activation. Our experiments compare backdoor attack performance on models trained using one of four regularization methods designed to constrain the latent space of the final hidden layer or classification logits of the image classifier.

Logit squeezing [21] introduced logit-squeezing regularization as a method to provide model robustness to adversarial examples. For a training image, $\mathbf{x}$, Logit-squeezing adds $\mathcal{L}_{\text{LS}} = ||f_L(\mathbf{x})||_2$ to the loss function to minimize the l_2 norm of the logit vector.

Manifold Mixup Introduced in [34], Manifold mixup (MIXUP) attempts to fill in gaps in the latent space manifold by interpolating the latent representations and corresponding predictions. Pairs of image hidden representations from the minibatch $(h(\mathbf{x}), h(\mathbf{x}'))$ are averaged according to a randomly sampled mixing weight $\gamma \sim Uniform(0, 1)$. The loss function to train the classifer is then the cross-entropy, $\mathcal{H}$, between the network's prediction for interpolated hidden

[1] https://www.kaggle.com/alxmamaev/flowers-recognition

state pairs and the γ weighted average of true one-hot class label distributions $(\mathbf{y}, \mathbf{y}')$:

$$\begin{aligned} \mathcal{L}_{\texttt{mix}} = & \quad \mathcal{H}(g(\mathbf{h}_{\texttt{mix}}), \mathbf{y}_{\texttt{mix}}) && (1) \\ \mathbf{h}_{\texttt{mix}} = & \quad (1-\gamma)h(\mathbf{x}') + \gamma h(\mathbf{x}) && (2) \\ \mathbf{y}_{\texttt{mix}} = & \quad (1-\gamma)\mathbf{y}' + \gamma \mathbf{y} && (3) \end{aligned}$$

Contrastive Loss Contrastive loss [10] encourages hidden representations from the same object class to be close together, and hidden representations from different object classes to be far apart. Let $\mathbf{x}$ and $\mathbf{x}'$ be two images. The contrastive regularization $L_{\texttt{contrast}}$ is:

$$\frac{1}{n}\|h(\mathbf{x}) - h(\mathbf{x}')\|_2 \tag{4}$$

if $\mathbf{x}$ and $\mathbf{x}'$ are the same class, and otherwise:

$$\frac{n-1}{n}\max(0, c - \|h(\mathbf{x}) - h(\mathbf{x}')\|_2) \tag{5}$$

Soft Nearest Neighbors Loss Soft Nearest Neighbors Loss (SNNL) [14] regularization was introduced to improve hidden space representations in many settings. SNNL weights the contribution of a pair of samples in a batch relative to the probability of being picked randomly as a nearest neighbor. With batch samples $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}), i = 1, ..., b.$ and temperature T, the SNNL regularization term is:

$$\mathcal{L}_{snn} = -\log\left(\frac{\sum_{j\neq i, \mathbf{y}^{(i)} \neq \mathbf{y}^{(j)}} e^{-\frac{\|h(\mathbf{x}^{(i)}) - h(\mathbf{x}^{(j)})\|^2}{T}}}{\sum_{k \neq i} e^{-\frac{\|h(\mathbf{x}^{(i)}) - h(\mathbf{x}^{(k)})\|^2}{T}}}\right) \tag{6}$$

2.4. Trigger Patterns

In this work, the backdoor is embedded in a model via data poisoning with trigger patterns embedded in adversarially re-labeled images. Let $\mathbf{x} \in \mathbb{R}^{H\times W\times 3}$ be a training set image, let $\alpha \in [0, 1]$ be the transparency of the trigger, and let $\mathbf{m} \in \{0, 1\}^{H\times W\times 3}$ be a mask with 1's in pixel positions the trigger will not alter. We introduce a trigger function $\mathcal{T}$ which returns a trigger $\mathbf{t}$. $\mathcal{T}$ may be constant, draw a random sample from a distribution of triggers (*e.g*., augmentation or perturbation of a trigger template), or depend on $\mathbf{x}$ in the case of an adaptive trigger. The general form for constructing a poisoned sample image, $\mathbf{p}$, with an embedded trigger is then:

$$\mathbf{p} = ((1-\alpha)\mathbf{x} + \alpha\mathcal{T}(\mathbf{x})) \odot (\mathbf{1} - \mathbf{m}) + \mathbf{x} \odot \mathbf{m} \tag{7}$$

where $\odot$ is the elementwise multiplication and boldface **1** is an all-ones tensor of the same dimension as the image.

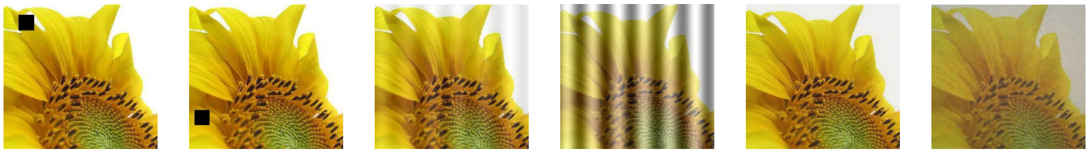
Four trigger types are experimentally evaluated, low-variance (LV), sine-wave (SIN), black square (BS), and random square (RS). Within a single experiment scenario, the same trigger type is applied to all poisoned samples. The black square trigger pattern is a 22 pixel square, located 22 pixels from both the top and left sides of the image. This is similar to the triangle checkerboard trigger used in [17]. The random square trigger is the same as the black square but placed at a random rather than fixed location in the image. The low-variance trigger pattern introduced in [32] is constructed with reference to a particular dataset to be poisoned. First a PCA decomposition is performed on the training data. Then an image not present in the training data is projected onto the last principal components that explains ≥ 0.5 percent of the variance in the dataset. This projection is then mapped back into the original image space to form the trigger pattern. The sine trigger, introduced in [3], consists of gray scale pixel intensities which vary horizontally across the image according to a sine function. In particular the value for all three channels at pixel (i, j) for the sine trigger is $0.4\sin(0.05\pi j)$.

Trigger patterns that overlay the entire image such as sine and low variance in particular are easy to detect if their α values are too high. Considering this, we pay particular attention to a set of experimental runs with α values of 0.5 and 0.1 for the low variance and sine triggers respectively. These α values were selected as the highest alpha value before the image alteration becomes completely apparent. For the black square trigger we use an α value of 1 since it is relatively inconspicuous, covering a small portion of the image. Figure 2 shows an image from the flowers dataset with triggers embedded with these particular α values.

2.5. Poisoned Samples

In addition to choice of trigger pattern, an adversary also has control over which images from the training dataset to poison (embed the trigger pattern). The ***source-class*** is the true class of an image upon which a trigger is embedded, and the ***poison-class*** is the class label given by the adversary. In the poisoning procedure we investigate, samples are drawn from the set of source-classes, embedded with a trigger pattern, and these poisoned samples then *supplant* clean samples from the poison-class. The untampered versions of the poisoned images remain in the source-classes. This method of poisoning ensures the number of images with each class label remains the same after poisoning, thereby eliminating class distribution shift due to data poisoning.

An important factor which may affect the success of data poisoning is the distribution of poisoned images within the poisoned dataset. We define the ***poison-rate*** as the percentage of the poison-class images replaced by poisoned samples. Let $N_1, N_2, ..., N_n$, be the number of images from each class in the training set, and t be the index of the poison-class. Given poison-rate λ, $\lfloor \lambda N_t \rfloor$ is the total number of samples to be replaced in the poison-class. For a set of source-classes $\mathcal{K} \subset \{1, ..., n\} \setminus \{t\}$, the expected number of samples, P_c,



(a) Square $\alpha = 1$ (b) Random Square $\alpha = 1$ (c) Sine $\alpha = 0.1$ (d) Sine $\alpha = 0.5$ (e) Low Variance $\alpha = 0.1$ (f) Low Variance $\alpha = 0.5$

Figure 2: Trigger patterns applied to an image from the Flowers dataset.

		N_1	N_2	N_3	N_4	N_5	
		710	980	734	675	904	
t	$\lfloor \lambda N_t \rfloor$	P_1	P_2	P_3	P_4	P_5	p
1-daisy	71	0	21.1	15.8	14.6	19.5	0.018
2-dandelion	98	23.0	0	23.8	21.9	29.3	0.025
3-rose	73	15.9	21.9	0	15.1	20.2	0.018
4-sunflower	67	14.3	19.7	14.8	0	18.2	0.017
5-tulip	90	20.6	28.5	21.3	19.6	0	0.023

Table 1: Poison class statistics with $\lambda = 0.1$ for many-to-one poisoning on the Flowers dataset.

drawn from each source-class, c, is:

$$P_c = \lfloor \lambda N_t \rfloor \frac{N_c}{\sum_{k\in\mathcal{K}} N_k} \tag{8}$$

The ***effective-poisoning-rate***, p, is defined as the percentage of the total number of training samples which are poisoned:

$$p = \frac{\lfloor \lambda N_t \rfloor}{\sum_{k\in\mathcal{K}}^{n} N_k} \tag{9}$$

The choice of source-classes has a direct effect on the distribution of poisoned images and so in addition to testing the effectiveness over various poison-rates we consider poisoning strategies which draw from a single source-class (***one-to-one***) or multiple source classes (***many-to-one***). In one-to-one poisoning, poisoned images from a single source-class supplant images from a single poison-class. In many-to-one poisoning, all classes excluding the poison-class are source-classes. Table 1 shows class distribution and poison sample distribution statistics for the Flowers dataset with the many-to-one poisoning strategy and a poison-rate $\lambda = 0.1$.

3. Experimental Setup

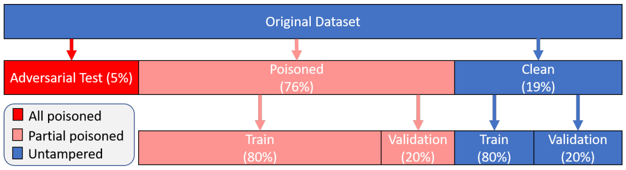


Figure 3: Dataset partitions, where all (adversarial), some (poisoned) or no (clean) images have been poisoned.

Data Partitioning Because the goal of this research is to assess the overall safety of a model, we partition the data to allow performance evaluation from both adversary and model developer perspectives. Adversarial success rate (the fraction of poisoned images predicted to be the poison-class) is used to evaluate the adversary's success, while model accuracy is used to assess the model developer's. The dataset partitioning is shown in Figure 3. The original dataset is partitioned into a 76/19/5 split. In our experiment, the largest partition (76%), which we call the ***poison-set*** plays the role of a larger, publicly available dataset that the adversary has tampered with, and that the model developer uses to train their first-pass computer vision model. The next largest partition (19%), which we call the ***clean-set***, simulates a smaller internal dataset curated by the model developer to fine-tune the first-pass computer vision model. Note that the clean-set is $1/5$th the size of the poison-set. Both the clean-set and poison-set are further split into respective 80/20 train/validation sets. We use the remaining 5% of the original dataset, which we call ***adversarial-test*** to evaluate the success rate of the adversary. Accordingly, all images in the adversarial test set are poisoned.

Poisoning details Preliminary results showed higher adversarial success rate when poison-class samples were not corrupted, thus when constructing the poison-set, the trigger pattern is not embedded onto samples drawn from the poison-class (*i.e.*, the poison-class is never one of the source-classes). The adversarial test set also contains no images from the poison-class, since the purpose of the adversarial test set is to gauge the adversary's ability to *change* a prediction. To eliminate performance effects associated with changes in class distributions, we maintain the same number of samples from each class prior to and post poisoning. To ensure this consistent class size across all experimental runs, poisoned samples are exchanged for samples in the poison-class, but their non-poisoned counterparts are not removed from source-class which they are drawn from.

State-of-the-art accuracy Due to the data splits needed to conduct our study (Fig. 3), our models only have access to around 60% of the original data for training. As expected, these models do not achieve the state-of-the-art of models trained on the full training set. Ultimately, our goal is not state-of-the-art performance, but a systematic comparison of data poisoning; that said, we do tune each model to achieve as competitive of performance as possible. As a sanity check on the correctness of our training process, we successfully replicated publicly reported results for each of our models using the full training set.

Training procedure Our procedure simulates the scenario where a model developer trains a base model on poisoned public data until the early stopping criterion (5 epochs with no improvement on validation accuracy) and then fine-tunes on an internal clean training set for a fixed number of epochs. During training, we monitor the model prediction accuracy on the clean and poison validation sets, and the adversarial success rate on the adversarial test set. For each experimental run we perform independent random splits and poison samples at the specified rate randomly.

4. Experimental Results and Analysis

In this section we analyze experimental results to answer several questions about backdoor attack success rate, backdoor persistence, and backdoor effects on model validation accuracy. Unless otherwise stated, the experiments described below use the "many-to-one" poisoning strategy, set poison-rate $\lambda = 0.1$, trigger pattern transparency $\alpha = 1$ for the Square and RS triggers patterns, $\alpha = 0.1$ for sine and $\alpha = 0.5$ for low variance.

4.1. Effect of Trigger Pattern and Model

We first analyze the effect of trigger patterns on different model architectures for backdoor poisoning. On the Flowers and CIFAR-10 datasets, we range over all trigger patterns, classes as poison-class, and architectures (180 runs total). We report average adversarial success rate and validation accuracy (over all classes) at early stopping after training on the poisoned training set. The average early stopping epoch for ResNet50, NasNet, and NasNet-Mobile was 14.6, 17.35, and 26.5, respectively. The resulting adversarial successes are shown in Table 2 (see "Retrained? No" rows). It reveals that the square and random square triggers are the most effective for the Flowers dataset, while the sine and square triggers are the most effective for CIFAR-10. It also shows that NasNet-Mobile is by far the most robust to poisoning on Flowers, while NasNet-Mobile and NasNet are both slightly more robust on CIFAR-10. Alarmingly, multiple combinations of model and trigger pattern yield adversarial success rates exceeding 60%.

Table 3 shows the model accuracy on the poisoned and clean validation sets (again see "Retrained? No" rows). For the models trained on Flowers, there is a negative correlation between model accuracy and robustness to poisoning, but for CIFAR-10 same models yield top performance on both. It is important to note that while the particular trigger pattern makes a significant difference in adversarial success, it has very little effect on the accuracy of the trained model, regardless of dataset. Lastly, the minimal gap between performance on the poisoned and clean validation sets is an unfortunate finding for the model developer's perspective, because it suggests that poisoned data may be hard to detect by inspection of model performance.

4.2. Effect of Retraining on Persistence

We next look at the extent to which different architectures retain the backdoor even after retraining on clean data. We take each of the models described in the previous experiments and fine-tune ("retrain") them on the smaller, untampered-with clean training set. The results are aggregated analogously and reported in the "Retrained? Yes" rows of Tables 2 and 3. These results show that clean retraining is an effective method for unlearning adversarial features. ResNet50, NasNet and NasNet mobile's adversarial test accuracy decrease significantly while model accuracy (on either clean or poisoned) is not affected. However, even after retraining NasNet still has almost 20% adversarial success on square trigger pattern, far above ResNet50 and NasNet-Mobile. Therefore, the model developer's decision on architecture may have significant implications on performance as well as safety of the model.

4.3. Effect of Regularization

For all regularization experiments we use the simple black square trigger pattern and the Flowers dataset (the most effective pattern for the dataset). For each of five regularization strategies and for each of the five possible poison-classes in the Flowers dataset we train 10 ResNet50 models with different random samples of poisoned images, holding all hyperparameter choices constant. The initial weights of the models are pre-trained on the ImageNet image classification task provided from Pytorch model zoo [11]. We use a learning rate of 0.00001, mini-batch size of 32, and Adam optimization to train all models.

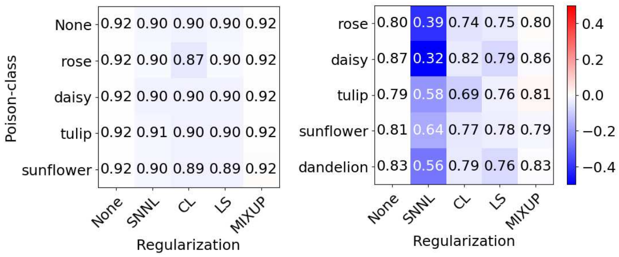
Figure 4 shows two tables of results: (a) accuracy on the clean validation set, which both the developer and adversary would like to maximize and (b) adversarial success rate, which the adversary would like to maximize but the developer would like to minimize. The columns of the tables correspond to the regularization strategy employed and the rows correspond to the poison-class. The color of each cell indicates the difference regularization has relative to no regularization (column 1). Blue indicates that regulariza-

			ResNet50				**NasNet**				**NasNet Mobile**			
Dataset	Split	Retrained?	Square	RS	Sine	LV	Square	RS	Sine	LV	Square	RS	Sine	LV
Flowers	Adversarial Test	No	0.75	0.64	0.24	0.26	0.65	0.58	0.18	0.06	0.33	0.15	0.14	0.12
Flowers	Adversarial Test	Yes	0.08	0.09	0.06	0.05	0.18	0.14	0.06	0.04	0.05	0.05	0.06	0.06
CIFAR-10	Adversarial Test	No	0.74	0.61	0.90	0.55	0.74	0.53	0.63	0.06	0.67	0.43	0.79	0.16
CIFAR-10	Adversarial Test	Yes	0.04	0.04	0.06	0.05	0.09	0.08	0.08	0.02	0.05	0.03	0.08	0.05

Table 2: Adversarial success before and after clean retraining, for Flowers and CIFAR-10.

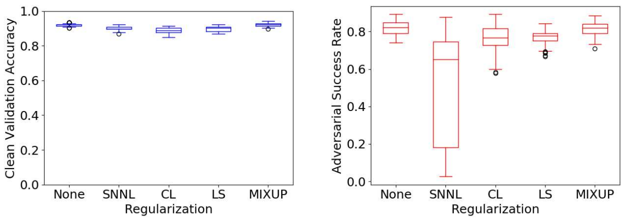
			ResNet50				**NasNet**				**NasNet Mobile**			
Dataset	Split	Retrained?	Square	RS	Sine	LV	Square	RS	Sine	LV	Square	RS	Sine	LV
Flowers	Poisoned	No	0.89	0.87	0.85	0.87	0.87	0.87	0.85	0.85	0.81	0.80	0.79	0.80
Flowers	Clean	No	0.88	0.87	0.87	0.89	0.87	0.87	0.87	0.87	0.83	0.83	0.81	0.83
Flowers	Poisoned	Yes	0.86	0.85	0.86	0.86	0.86	0.87	0.87	0.86	0.80	0.79	0.80	0.80
Flowers	Clean	Yes	0.89	0.89	0.89	0.90	0.87	0.88	0.89	0.89	0.81	0.82	0.82	0.84
CIFAR-10	Poisoned	No	0.73	0.74	0.69	0.74	0.93	0.92	0.92	0.92	0.86	0.85	0.86	0.85
CIFAR-10	Clean	No	0.74	0.74	0.69	0.74	0.93	0.92	0.93	0.93	0.87	0.86	0.87	0.86
CIFAR-10	Poisoned	Yes	0.74	0.73	0.73	0.73	0.91	0.91	0.91	0.91	0.85	0.85	0.85	0.85
CIFAR-10	Clean	Yes	0.74	0.74	0.74	0.74	0.93	0.93	0.93	0.93	0.86	0.86	0.86	0.86

Table 3: Accuracy before and after clean retraining, for Flowers and CIFAR-10.



(a) Clean validation accuracy (b) Adversarial success rate

Figure 4: Average clean validation accuracy and adversarial success rate over 10 experimental runs, with many-to-one poison-class strategy. The color bar shows the difference relative to no regularization (column 1).



(a) Clean validation accuracy (b) Adversarial success rate

Figure 5: Validation accuracy and adversarial success rate as a function of regularization strategy.

tion decreases the value. We see in Fig. 4a a marginal drop in clean validation accuracy for all regularization strategies except for Manifold Mixup which does not affect performance on the validation set. The largest drop in validation accuracy comes from using the contrastive loss with Rose as the poison-class. Fig. 4b shows that SNNL, Contrastive, and Logit Squeezing regularization strategies all have the effect of lowering average adversarial success rates. However, SNNL has a more dramatic effect, dropping the overall average adversarial success rate across all poison-classes by 31% absolute (from 82% to 51%). Note also that the poison-class has little effect on accuracy, but significantly affects adversarial success.

To get a sense of the consistency of these findings, Fig. 5 shows the spread of validation accuracy and adversarial success rate across the 50 experimental runs for each regularization strategy, as a box-and-whiskers plot. We see that all regularization strategies besides Manifold Mixup have a more dramatic affect on adversarial success rate than validation accuracy. The variance for adversarial success rate with SNNL loss is quite a bit larger compared to the other regularization methods. We conclude that regularization can be used to defend a model without significantly degrading the baseline performance on the validation set.

4.4. Effect of Trigger Pattern Transparency

Here we address effect of the trigger pattern transparency parameter, α. Because square and random square use $\alpha = 1$, we limit this analysis to the sine and low variance triggers. We concentrate the range of tested α values on the lower range, since higher α's are less realistic. We also only target

Accuracy		Poisoned		Clean	
Model	Dataset	1-to-1	M-to-1	1-to-1	M-to-1
ResNet50	Flowers	0.87 ± 0.01	0.89	0.90 ± 0.01	0.88
NasNet	Flowers	0.86 ± 0.02	0.89	0.85 ± 0.01	0.86
NasNet-M	Flowers	0.78 ± 0.02	0.82	0.81 ± 0.03	0.84
ResNet50	CIFAR-10	0.71 ± 0.03	0.70	0.71 ± 0.02	0.69
NasNet	CIFAR-10	0.92 ± 0.01	0.92	0.93 ± 0.00	0.93
NasNet-M	CIFAR-10	0.85 ± 0.01	0.85	0.85 ± 0.01	0.86

Table 4: Accuracy for one-to-one vs many-to-one.

the most robust poison-classes, truck and rose, for CIFAR-10 and Flowers, respectively. These experiments compare poison-rates of $\lambda = 0.05$ and $\lambda = 0.1$, with a total of 432 runs. Our results are shown in Fig. 6, the top row using Flowers and the bottom row CIFAR-10.

We find that higher α values can increase the trigger's effectiveness significantly, although the most effective performance comes when the trigger pattern is clearly perceptually detectable to humans. However, safety concerns remain because high α but low poison-rate attacks may be feasible in a big dataset where manual inspection of even a fraction of the samples is impractical. Figs. 6c and 6f show performance after retraining with clean data, finding that retraining is not always effective against full image trigger patterns at sufficiently high α. A comprehensive defensive strategy should include a mechanism to detect "obvious" samples perturbed with high alpha triggers. We attribute the 0% adversarial success at $\alpha = 1.0$ in Fig. 6d to two factors: 1) at $\alpha = 1$, all poisoned samples are identical and thus 0% and 100% are the only valid outcomes, and 2) noise in the training process at the particular early stopping point.

4.5. Effect of Poison-rate

We next study the effect of poison-rate, λ. We used CIFAR-10 as it has more samples per class than the Flowers dataset, providing us finer granularity for the poison-rate. Once again, we only target truck (the most robust poison-class on CIFAR-10) and focus primarily on small λ values because they are more practical. Here we use only NasNet, since it has the highest clean and poisoned validation accuracy on CIFAR-10 using our standard hyper-parameters with a total of 44 runs. Unsurprisingly, Fig. 7a shows that accuracy on poisoned validation steadily decreases as the poison-rate increases (as the poison-rate increases, the number of actual training samples in the target class decreases). Fig. 7b plots the adversarial success rate as a function of poison-rate for different trigger patterns. Sine requires the least poisoning, as it is extremely effective even with 2% poisoning. Random square requires the most poisoning, only finding middling success with impractically high poisoning rates.

Adversarial Success		Adversarial Test	
Model	Dataset	1-to-1	M-to-1
ResNet50	Flowers	0.54 ± 0.02	0.72
NasNet	Flowers	0.37 ± 0.02	0.71
NasNet-M	Flowers	0.13 ± 0.14	0.35
ResNet50	CIFAR-10	0.58 ± 0.18	0.97
NasNet	CIFAR-10	0.27 ± 0.07	0.73
NasNet-M	CIFAR-10	0.40 ± 0.11	0.85

Table 5: Adversarial success for one-to-one vs many-to-one.

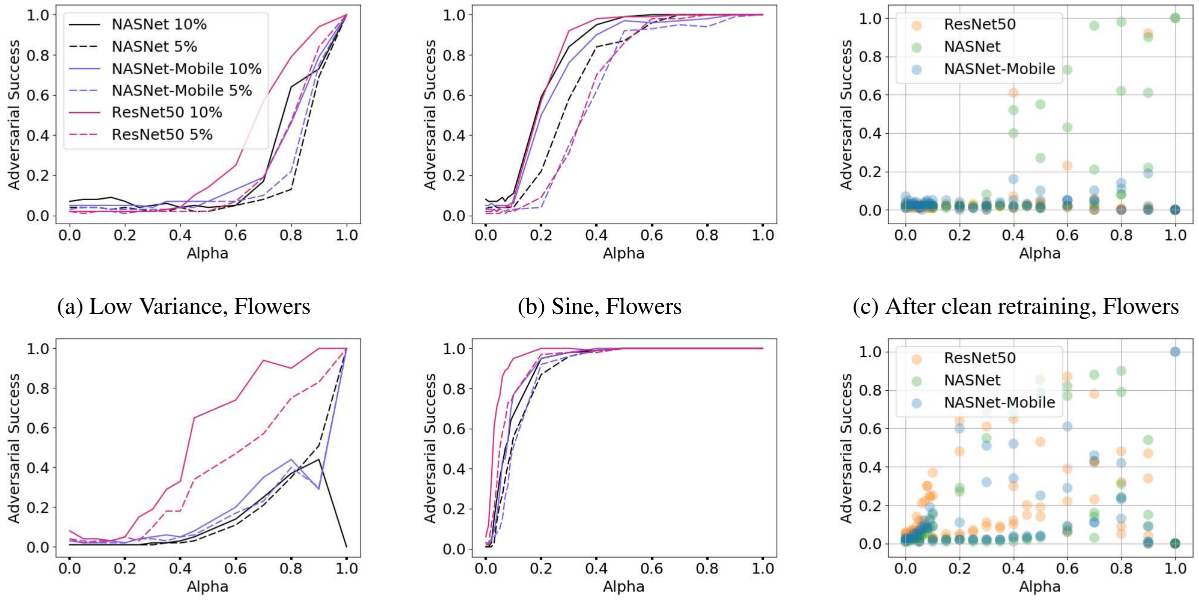
4.6. One-to-one vs Many-to-one

Lastly, we evaluate whether the one-to-one ("1-to-1") or many-to-one ("M-to-1") poisoning strategy is more effective. Table 4 compares the accuracies of these two strategies for all models on both datasets. Square is used to poison Flowers while Sine is used to poison CIFAR-10 (the most effective patterns for them, respectively). Recall that one-to-one and many-to-one use the same number of poisoned samples for a given poison-rate; the only difference is the source of the poisoned samples. The table reveals that these poisoning strategies do not have a significant impact on either the poisoned or clean validation set accuracies. In contrast, Table 5 shows that many-to-one is significantly more effective than one-to-one in terms of adversarial success. We hypothesize this is because the model incorporates the adversarial features better when the trigger pattern is spread across many classes, all pointing to the same target class.

5. Conclusions and Future Work

This paper presents a systematic study of backdoor poisoning attacks on image classifiers. We evaluate the effect of design decisions within the model developer's control, including model architecture, regularization scheme, and any additional fine-tuning on a smaller, clean dataset, as well as those within the control of an adversary, including the trigger pattern and the rate and strength of the poisoning. We evaluate these on two datasets, Flowers and CIFAR-10, to assess the sensitivity to the particular training task. We report four key findings:

1. Adversarial success rate varies widely depending on several factors, including model architecture, trigger pattern and regularization technique.
2. While one would expect model performance and adversarial success to be negatively correlated, we find this rarely to be the case, suggesting poisoned models are not detectable through performance inspection alone.
3. Regularization typically reduces backdoor success rate, although it can have no effect or even slightly increase it, depending on the form of regularization.
4. Backdoors inserted through data poisoning can be ren-



(a) Low Variance, Flowers

(b) Sine, Flowers

(c) After clean retraining, Flowers

(d) Low Variance, CIFAR-10

(e) Sine, CIFAR-10

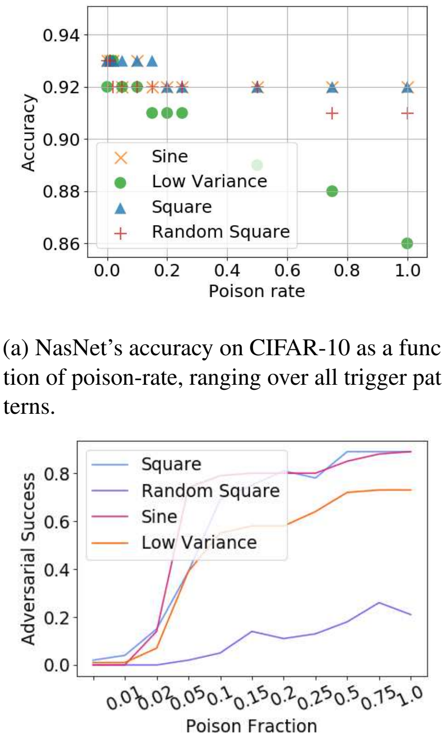
(f) After clean retraining, CIFAR-10

Figure 6: Effect of α.

dered ineffective after just a few epochs of additional training on a small set of clean data without affecting the model's performance.

We intend our current assessment to serve as a resource for safe and effective model development practices in face of adversity. However, adversarial machine learning is a rapidly evolving field of research. Backdoor data poisoning assessment can be characterized as the analysis of a two player zero-sum game with emerging innovative actions for the roles of adversary and developer, and so a complete analysis is beyond the scope of any single research study.

For future work, one could extend our assessment along three complementary dimensions. First, one could explore a greater range of values for studied factors (Fig. 1); *e.g.*, assessing with a larger dataset such as ImageNet. Recent work also motivates additional regularization methods, such as Gaussian mixture loss [37] and ℓ_2 regularization [6], which can also partially mitigate data poisoning attacks. Second, there are further factors of model developer decisions influencing model behavior which should be explored. To our knowledge, the choice of optimizer (*e.g.*, SGD, Adam, AdamW [25]) has not been evaluated in the context of backdoor data poisoning. Lastly, one could extend our assessment of adversarial exploits. For instance, in this work we assess attacks which falsely label images, but *clean-label* backdoor attacks without label alteration have recently been demonstrated [3, 33, 28, 38].



(a) NasNet's accuracy on CIFAR-10 as a function of poison-rate, ranging over all trigger patterns.

(b) NasNet's adversarial success on CIFAR-10 as a function of poison-rate.

Figure 7: Effect of poison-rate.

References

[1] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. *CoRR*, abs/1807.00459, 2018.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014.

[3] Mauro Barni, Kassem Kallas, and Benedetta Tondi. A new backdoor attack in cnns by training set corruption without label poisoning. *CoRR*, abs/1902.11237, 2019.

[4] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.

[5] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[6] Javier Carnerero-Cano, Luis Muñoz-González, Phillippa Spencer, and Emil C Lupu. Regularisation can mitigate poisoning attacks: A novel analysis based on multiobjective bilevel optimisation. *arXiv preprint arXiv:2003.00040*, 2020.

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

[8] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[9] Chung-Cheng Chiu, Tara N. Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J. Weiss, Kanishka Rao, Katya Gonina, Navdeep Jaitly, Bo Li, Jan Chorowski, and Michiel Bacchiani. State-of-the-art speech recognition with sequence-to-sequence models. *CoRR*, abs/1712.01769, 2017.

[10] Sumit Chopra, Raia Hadsell, Yann LeCun, et al. Learning a similarity metric discriminatively, with application to face verification. In *CVPR (1)*, pages 539–546, 2005.

[11] Torch Contributors. *Pytorch Model Zoo*, 2019 (accessed July 1, 2019).

[12] 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.

[13] Jacob Dumford and Walter J. Scheirer. Backdooring convolutional neural networks via targeted weight perturbations. *CoRR*, abs/1812.03128, 2018.

[14] Nicholas Frosst, Nicolas Papernot, and Geoffrey Hinton. Analyzing and improving representations with the soft nearest neighbor loss. *arXiv preprint arXiv:1902.01889*, 2019.

[15] Matt Gorbett and Nathaniel Blanchard. Utilizing network properties to detect erroneous inputs. *arXiv preprint arXiv:2002.12520*, 2020.

[16] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. Speech recognition with deep recurrent neural networks. *CoRR*, abs/1303.5778, 2013.

[17] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *CoRR*, abs/1708.06733, 2017.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[19] Ling Huang, Anthony D. Joseph, Blaine Nelson, Quoc V. Le, Benjamin I. P. Rubinstein, and J. D. Tygar. Adversarial Machine Learning. *In Proceedings of 4th ACM Workshop on Artificial Intelligence and Security*, pages 43–58, 2011.

[20] Yujie Ji, Xinyang Zhang, Shouling Ji, Xiapu Luo, and Ting Wang. Model-reuse attacks on deep learning systems. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 349–363. ACM, 2018.

[21] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. Adversarial logit pairing. *arXiv preprint arXiv:1803.06373*, 2018.

[22] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55, 2014.

[23] Cong Liao, Haoti Zhong, Anna Squicciarini, Sencun Zhu, and David Miller. Backdoor embedding in convolutional neural network models via invisible perturbation. *arXiv preprint arXiv:1808.10307*, 2018.

[24] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. 2017.

[25] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[26] David J Miller, Zhen Xiang, and George Kesidis. Adversarial learning targeting deep neural network classification: A comprehensive review of defenses against attacks this article provides a contemporary survey of adversarial learning (al), focused particularly on defenses against attacks on deep neural network classifiers. *Proceedings of the IEEE*, 2020.

[27] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[28] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. Hidden trigger backdoor attacks. *arXiv preprint arXiv:1910.00033*, 2019.

[29] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. *arXiv preprint arXiv:1804.00792*, 2018.

[30] Ezekiel Soremekun, Sakshi Udeshi, Sudipta Chattopadhyay, and Andreas Zeller. Exposing backdoors in robust machine learning models. *arXiv preprint arXiv:2003.00865*, 2020.

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

[32] Thomas Tanay, Jerone T. A. Andrews, and Lewis D. Griffin. Built-in vulnerabilities to imperceptible adversarial perturbations. *CoRR*, abs/1806.07409, 2018.

[33] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Clean-label backdoor attacks. 2018.

[34] Vikas Verma, Alex Lamb, Christopher Beckham, Aaron Courville, Ioannis Mitliagkis, and Yoshua Bengio. Manifold mixup: Encouraging meaningful on-manifold interpolation as a regularizer. *arXiv preprint arXiv:1806.05236*, 2018.

[35] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.

[36] Zhen Xiang, David J Miller, and George Kesidis. A benchmark study of backdoor data poisoning defenses for deep neural network classifiers and a novel defense. In *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2019.

[37] Muhammad Yaseen, Muneeb Aadil, and Maria Sargsyan. Preventing clean label poisoning using gaussian mixture loss. *arXiv preprint arXiv:2003.00798*, 2020.

[38] Chen Zhu, W Ronny Huang, Ali Shafahi, Hengduo Li, Gavin Taylor, Christoph Studer, and Tom Goldstein. Transferable clean-label poisoning attacks on deep neural nets. *arXiv preprint arXiv:1905.05897*, 2019.

[39] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *CoRR*, abs/1707.07012, 2017.