What Makes Your Data Unavailable To Deep Learning?

Anonymous Author(s)

ABSTRACT

Availability attacks[1], which poison the training data with imperceptible perturbations, can make the data *not exploitable* by machine learning algorithms so as to prevent unauthorized use of data. In this work, we investigate why these perturbations work in principle. We are the first to unveil an important population property of the perturbations of these attacks: they are almost **linearly separable** when assigned with the target labels of the corresponding samples, which hence can work as *shortcuts* for the learning objective. We further verify that linear separability is indeed the workhorse for availability attacks. We synthesize linearly-separable perturbations as attacks and show that they are as powerful as the deliberately crafted attacks. Moreover, such synthetic perturbations are much easier to generate. For example, previous attacks need dozens of hours to generate perturbations for ImageNet while our algorithm only needs several seconds. Our finding also suggests that the *shortcut learning* is more widely present than previously believed as deep models would rely on shortcuts even if they are of an imperceptible scale and mixed together with the normal features.

CCS CONCEPTS

• **Data Privacy and Ethics**; • **Learning Paradigms** → *Adversarial Learning*;

KEYWORDS

data security, data poisoning, shortcut learning

ACM Reference Format:

Anonymous Author(s). 2022. What Makes Your Data Unavailable To Deep Learning?. *Submitted to ACM Conference (Conference'22).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Sharing personal data online has become an important lifestyle for many people. Despite big datasets crawled from the Internet keep advancing the state-of-the-art deep models [7, 9, 19], there are increasing concerns about the unauthorized use of personal data [6, 23, 36]. For instance, a private company has collected more than three billion face images to build commercial face recognition models without acquiring any user consent [22]. To address those concerns, many data poisoning attacks have been proposed to prevent data from being learned by unauthorized deep models [12–14, 25, 44, 48, 51]. They add imperceptible perturbations to the training data so that the model cannot learn much information from the data and the model accuracy on unseen data is arbitrarily bad. These attacks make the data not *available/exploitable* by machine learning models and are known as *availability attack* [5]. We give an illustration of this type of attack in Figure 1.

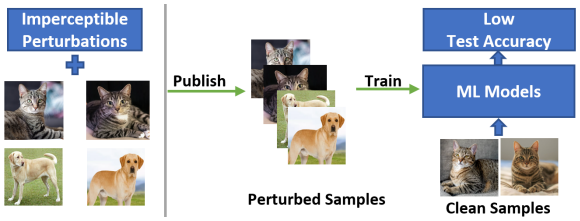

Figure 1: An illustration of clean-label availability attacks.

In literature, there are roughly three methods to construct the availability attack against deep neural networks. The first method generates the perturbations as the solution of a bi-level optimization problem [4, 12, 13, 51]. The bi-level optimization problem updates the perturbations to minimize the loss on perturbed data while maximizing the loss on clean data.

Secondly, Huang et al. [25] conceive a simpler poisoning attack called *error-minimizing noise*, where the perturbation on training data is crafted by minimizing the training loss. The intuition is that if the perturbation can reduce the loss to zero, then there is nothing left for backpropagation in the regular training procedure. Recently, Nakkiran [31] and Fowl et al. [14] point out that *error-maximizing noises*, which are commonly used as adversarial examples, can serve as an availability attack as well. Despite these quite different approaches, all of them are powerful availability attacks. Intrigued by this observation, we ask the following question:

What is the underlying workhorse for availability attacks against deep neural networks?

To answer this question, we first take a close look at the perturbations of existing attacks. We visualize the perturbations of several availability attacks via two-dimensional T-SNEs [49] in Figure 2 and Figure 8 in Appendix A. The experimental setup is depicted in Section 2.2. Surprisingly, the perturbations with the same class label are well clustered, suggesting that the perturbations would be **linearly separable** in the original high-dimensional space. We confirm this by fitting the perturbations with linear models. The perturbations are assigned with the labels of their target examples. It turns out that simple logistic regression models can fit the perturbations of four representative attacks with $> 90\%$ training accuracy. This finding suggests that linearly-separable perturbations may be the key for existing availability attacks to succeed.

To further confirm that the linear separability is a sufficient (not only necessary) condition, we reverse the above procedure: synthesizing some simple linearly-separable perturbations to see if they can

[1] More precisely, we investigate clean-label availability attacks. Some availability attacks inject malicious training samples instead of perturbing existing ones [4].

Conference'22, August 2022, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

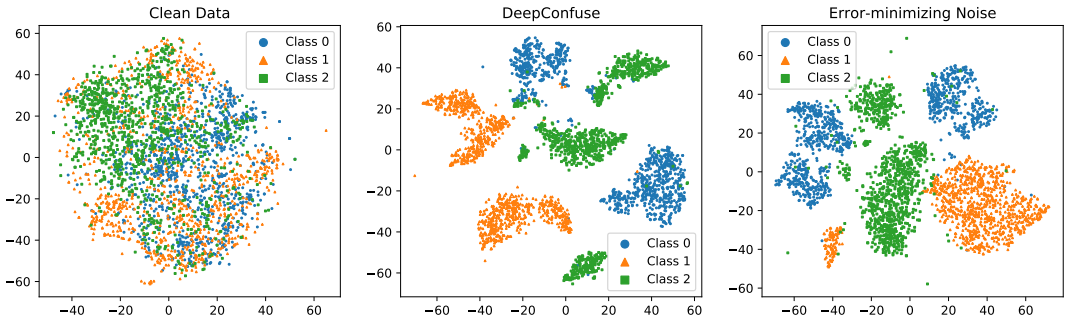

Figure 2: T-SNEs of the first three classes of clean CIFAR-10 data and the perturbations generated via DeepConfuse [12] and error-minimizing noises [25]. The perturbations are flattened and normalized into unit norms.

serve as availability attacks. Specifically, we first generate some initial synthetic perturbations via a method in Guyon [18] and then add a new post-processing procedure so that the synthetic perturbations remain effective when data augmentations are applied. Extensive experiments on benchmark datasets and models demonstrate that synthetic perturbations can be as powerful as existing availability attacks. Notably, generating synthetic perturbations is significantly easier and cheaper than existing attacks as it does not require solving any optimization problems. For example, recent attacks need dozens of hours to generate perturbations for the ImageNet data, while generating synthetic perturbations only needs several seconds. This finding reveals that using linearly-separable perturbations is indeed the workhorse to the success of state-of-the-art availability attacks.

The above finding conceptually links the availability attacks with *shortcut learning* [16]. Shortcut learning stands for the behavior that deep models tend to rely on features that do not generalize on realistic test data. Such features are referred to as shortcuts. With this concept, the perturbations of availability attacks are also shortcuts. We note that the shortcuts in previous works are usually part of natural data, which are somehow heuristic, e.g., "grass" is a shortcut for recognizing "cow" in natural images [3, 16]. In this work, we expose a more explicit form of shortcuts and discuss extensively how to construct such shortcuts. We unveil that deep learning models would overwhelmingly rely on spurious shortcuts even though the shortcuts are scaled down to an imperceptible magnitude. This finding exposes a fundamental vulnerability of deep models and hence may be of independent interest to the community.

Our contributions are summarized as follows:

- We reveal that the perturbations of several existing availability attacks are (almost) linearly separable.
- We propose to use synthetic shortcuts to perform availability attack, which is much cheaper and easier to conduct.
- We link availability attacks with shortcut learning and greatly widen the understanding of shortcuts in deep learning.

1.1 Related Work

Data poisoning. In general, data poisoning attacks perturb training data to intentionally cause some malfunctions of the target model [5, 17, 40]. A common class of poisoning attacks aims to cause test-time error on some given samples [8, 15, 43, 54] or on all unseen samples [12–14, 25, 44, 51]. The latter attacks are also known as availability attacks [2]. In this work, we investigate and reveal the workhorse of availability attacks. We show that the perturbations of these availability attacks are (almost) linearly separable. We further confirm that synthesised linearly-separable perturbations can perform strong attacks.

Backdoor attacks are another type of data poisoning attack that perturbs training data so that the attacker can manipulate the target model's output with a designed trigger [11, 33, 34, 39, 41, 47]. The perturbations of backdoor attacks have two major differences compared to those of availability attacks. Firstly, the perturbations of availability attacks are imperceptible. Secondly, advanced availability attacks use a different perturbation for every sample while a backdoor trigger is applied to multiple samples. In the threat model of availability attacks, the data are probably crowdsourced. It is preferred to use different perturbations for different samples in such a setting. Otherwise, the adversarial learner can remove perturbations from all related samples if any poisoned image leaks.

Shortcut learning. Recently, the community has realized that deep models may rely on shortcuts to make decisions [3, 16, 35]. Shortcuts are spurious features that are correlated with target labels but do not generalize on test data. Beery et al. [3] show that a deep model would fail to recognize cows when the grass background is removed, suggesting that the model takes "grass" as a shortcut for "cow". Niven and Kao [35] show that large language models use the strong correlation between some simple words and labels to make decisions, instead of trying to understand the sentence. For instance, the word "not" is directly used to predict negative labels. In this work, we show shortcut learning exists more widely than previously believed. Our experiments in Section 3 demonstrate that deep models only pick shortcuts even if the shortcuts are scaled down to an imperceptible magnitude and mixed together with normal features. These experiments reveal another form of shortcut learning, which has been unconsciously exploited by availability poisoning attacks. There also exist other synthesized datasets that offer a stratification of features [21, 42]. Those synthetic data contain shortcuts that can not be used as perturbations as they are visible and affect the normal data utility. For example, Shah et al. [42] generate synthetic data

by vertically concatenating images from the MNIST and CIFAR-10 datasets.

1.2 Notations

We use bold lowercase letters, e.g., $\boldsymbol{v}$, and bold capital letters, e.g., $\boldsymbol{M}$, to denote vectors and matrices, respectively. The L_p norm of a vector $\boldsymbol{v}$ is denoted by $\|\boldsymbol{v}\|_p$. A sample consists of feature $\boldsymbol{x}$ and label y. We use $\mathbb{D}$ to denote a dataset that is sampled from distribution $\mathcal{D}$. In this paper, we focus on classification tasks. The classification loss of a model f on a given sample is denoted by $\ell(f(\boldsymbol{x}), y)$.

2 AVAILABILITY ATTACKS USE LINEARLY-SEPARABLE PERTURBATIONS

In this section, we investigate the common characteristic of existing availability attacks. First, We briefly introduce three different approaches to construct availability attacks. Then, we visualize the perturbations of advanced attacks with two-dimensional T-SNEs. The plots suggest that the perturbations of all three kinds of attacks are some 'easy' features. Finally, we verify that the perturbations of these attacks are almost linearly separable by fitting them with simple models.

2.1 Three Types Of Availability Attacks

2.1.1 The Alternative Optimization Approach. We first introduce the alternative optimization approach to generate perturbations for availability attacks. It solves the following bi-level objective,

$$\begin{aligned} &\arg\max_{\{\boldsymbol{\delta}\}\in\Delta} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}[\ell(f^*(\boldsymbol{x}), y)], \\ &\text{s.t. } f^* \in \arg\min_{f} \sum_{(\boldsymbol{x},y)\in\mathbb{D}} \ell(f(\boldsymbol{x}+\boldsymbol{\delta}), y), \end{aligned} \tag{1}$$

where $\boldsymbol{\delta}$ is a sample-wise perturbation and Δ is a constraint set for perturbations. The two formulas in Equation 1 directly reflect the goal of availability attacks. Specifically, the optimal solution on perturbed data (specified by the second formula) should have the largest loss on clean data (specified by the first formula). The constraint set Δ is set to make the perturbations imperceptible, e.g., a small L_p norm ball.

Directly solving Equation (1) is intractable for deep neural networks. Recent works have designed multiple approximate solutions [12, 13, 51]. Feng et al. [12] use multiple rounds of optimization to generate perturbations. At each round, they first approximately optimize the second objective by updating a surrogate target model on perturbed data for a few steps. Then they approximately optimize the first objective by updating a generator for a few steps. The outputs of the generator are used as perturbations. Another example is the Neural Tangent Generalization Attacks (NTGAs) in Yuan and Wu [51]. They approximately optimize the bi-level objective based on the recent development of Neural Tangent Kernels [26].

2.1.2 The Error-minimizing Noise. Huang et al. [25] propose another bi-level objective to generate perturbations. Instead of solving Equation (1), they use the following objective,

$$\arg\min_{\{\boldsymbol{\delta}\}\in\Delta} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathbb{D}}[\min_{f} \ell(f(\boldsymbol{x}+\boldsymbol{\delta}), y)]. \tag{2}$$

The perturbations are intentionally optimized to reduce the training loss. The main motivation is that if the training loss is zero, then the target model will have nothing to learn from the data because there is nothing to backpropagate. A randomly initialized model is used as a surrogate of the target model. They also use multiple rounds of optimization to generate perturbations. At each round, they first train the surrogate model for a few steps to minimize the loss on perturbed data. Then they optimize the perturbations to also minimize the loss of the surrogate model. They repeat the above process until the loss on perturbed data is smaller than a pre-defined threshold.

2.1.3 Adversarial Examples. Instead of using bi-level objectives, Fowl et al. [14] show that the common objectives of adversarial examples are sufficient to generate powerful data poisoning perturbations. They use both untargeted (the first objective) and targeted adversarial examples (the second objective),

$$\begin{aligned} &\arg\max_{\{\boldsymbol{\delta}\}\in\Delta} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathbb{D}}\left[\ell\left(f\left(\boldsymbol{x}+\boldsymbol{\delta}\right), y\right)\right], \\ &\arg\min_{\{\boldsymbol{\delta}\}\in\Delta} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathbb{D}}\left[\ell\left(f\left(\boldsymbol{x}+\boldsymbol{\delta}\right), y'\right)\right], \end{aligned} \tag{3}$$

where $y' \neq y$ is an incorrect label and f is a trained model. Surprisingly, Fowl et al. [14] demonstrate that these simple objectives can generate perturbations that achieve state-of-the-art attack performance.

2.2 Visualizing The Perturbations

Although the three approaches in Section 2.1 have different objectives, they all manage to perform powerful attacks. Intrigued by this observation, we try to find out whether there is a common pattern among different types of availability attacks. If so, the common pattern may be the underlying workhorse for availability attacks.

To find such a common pattern, we first visualize different types of perturbations, including DeepConfuse [12], NTGA [51], error-minimizing noises [25], and adversarial examples [14]. We compute their two-dimensional t-SNEs [49]. These four attacks achieve advanced attack performance and cover all the three approaches in Section 2.1. We use their official implementations to generate perturbations. Detailed configurations are in Appendix B.

The two-dimensional t-SNEs of DeepConfuse and error-minimizing noises are shown in Figure 2. The plots of NTGA and adversarial examples are presented in Figure 8 of Appendix A due to the space limit. Surprisingly, for all the attacks considered, the perturbations for the same class are well clustered, suggesting that even linear models can classify them well. For comparison purposes, we also compute the t-SNEs of the clean data. As shown in Figure 2, in contrast with the t-SNEs, the projections of different classes of the clean data are mixed together, which indicates that they require a complex neural network to be correctly classified. This observation suggests that using linearly-separable perturbations may be the common pattern among availability attacks.

2.3 Availability Attacks Use Linearly-Separable Perturbations

To quantify the 'linear separability' of the perturbations, we fit the perturbations with simple models and report the training accuracy.

Table 1: Training accuracy (in %) of simple models on clean data and the perturbations of different attacks.

Algorithm	Linear Model	Two-layer NN
Clean Data	49.9	70.1
DeepConfuse [12]	100.0	100.0
NTGA [51]	100.0	100.0
Error-minimizing [25]	100.0	100.0
Adv. Examples (Untargeted) [14]	91.5	99.9
Adv. Examples (Targeted) [14]	100.0	100.0

The perturbations are labeled with the labels of the corresponding target examples. The simple models include linear models and two-layer neural networks. Details can be found in Appendix B.

The results are presented in Table 1. Compared to the results on clean data, simple models can easily fit the perturbations. On all attacks considered, linear models achieve more than 90% training accuracy and two-layer neural networks achieve nearly 100% training accuracy. These results confirm that the perturbations of advanced availability attacks are all (almost) linearly separable.

Existing attacks against deep neural networks all use ReLU activation functions in their crafting models. Deep models with ReLU activation functions are known to learn piecewise linear functions in input space [1]. Therefore, it is natural to wonder whether the linear separability is stemming from the property of ReLU. In Section 5.1, we replace the ReLU layers with Tanh layers in the crafting models of adversarial examples and error-minimizing noises. It turns out that simple models still can easily fit the new perturbations, which demonstrates that the linear separability is an intrinsic property of these availability attacks rather than something associated with a specific network structure.

2.4 Connecting To Shortcut Learning

The fact that the perturbations can be easily fitted by linear models naturally connects to a recent concept named shortcut learning [16]. Shortcut learning summarizes a general phenomenon when any learning system makes decisions based on spurious features that do not generalize on realistic test data[2]. Shortcut features have been found in different fields. For vision tasks, Beery et al. [3] show deep models fail to recognize cows when the grass background is removed from images, suggesting the grass background is a shortcut for predicting cows. In the field of natural language processing, Niven and Kao [35] show language models use the strong correlation between some simple words and labels to make decisions, instead of really understanding the data.

With the presence of shortcut learning, it seems reasonable to postulate that the perturbations of existing attacks succeed by creating shortcuts to the target model. We give an illustration in Figure 3. A major difference between the perturbations of poisoning attacks and existing shortcut features is that the perturbations are of an imperceptible scale and mixed together with useful features. Since there is no direct evidence to show deep models will take this kind of shortcuts, in the next section, we design experiments to confirm the postulated explanation. We synthesize imperceptible and linearly-separable perturbations and show deep models are very vulnerable to such synthetic shortcuts.

[2]Geirhos et al. [16] use a more specific definition of shortcuts. They denote shortcuts as those features that do not generalize on out-of-distribution (OOD) data. We note that poisoning attacks would change the distribution of training data and hence make the clean test data 'OOD' with respect to the trained model.

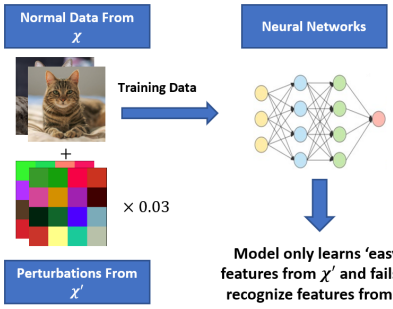

Figure 3: An illustration of how the perturbations of availability attacks work as shortcuts.

3 LINEAR SEPARABILITY IS A SUFFICIENT CONDITION FOR AVAILABILITY ATTACKS TO SUCCEED

Although we have demonstrated that the perturbations of four advanced attacks are all almost linearly separable, it is a bit early to claim that 'linear separability' is the underlying working principle of availability poisoning attacks. Perturbations are linearly separable may only be a necessary but not sufficient condition for poisoning attacks to succeed. In order to verify this postulated explanation, we use simple synthetic data to serve as perturbations and compare their effectiveness with existing poisoning attacks. It turns out that the synthetic perturbations are as powerful as advanced attacks.

The rest of this section is organized as follows. In Section 3.1, we first give an algorithm for generating synthetic data as perturbations. In Section 3.2, we verify the effectiveness of synthetic perturbations on different models and datasets.

3.1 Generating Synthetic Perturbations as Shortcuts

The synthetic perturbations in this section are generated via two building blocks. In the first block, we use a method in Guyon [18] to generate samples from some normal distributions. In the second block, we transfer the samples into the image format so that they can be applied to benchmark vision tasks. We give the pseudocode in Algorithm 1.

The first building block proceeds as follows. We first generate some points that are normally distributed around the vertices of a hypercube. The points around the same vertex are assigned with the same label. Then for each class, we introduce different covariance. Any two classes of the generated points can be easily classified by a hyperplane as long as the side length of the hypercube is reasonably large.

In the second building block, we pad each dimension of the sampled points and reshape them into two-directional images. The padding operation introduces local correlation into the synthetic images. Local correlation is an inherent property of natural images.

In Section 5.2, we show the padding operation is necessary to make the synthetic perturbations remain effective when data augmentation methods are applied.

Algorithm 1: Generating Perturbations for Vision Datasets

1: **Input:** number of classes k, number of examples in each class $\{n_i\}_{i=1}^{k}$, image size (w, h), patch size p, norm bound ϵ.
2: Compute $w' = round(w/p) + 1$ and $h' = round(h/p) + 1$.
// The first block: using the method in Guyon [18] to generate some initial data points.
3: Create a $w'h'$-dimensional hypercube.
4: **for** $i = 1$ **to** k **do**
5: Generate $\boldsymbol{D}^{(i)} \in \mathbb{R}^{n_i \times w'h'}$, where each row of $\boldsymbol{D}^{(i)}$ is sampled from $\mathcal{N}(0, \boldsymbol{I}_{w'h' \times w'h'})$.
6: *// Introduce random covariance among columns.*
7: Uniformly sample the elements of $\boldsymbol{A} \in \mathbb{R}^{w'h' \times w'h'}$ from $[-1, 1]$.
8: Compute $\boldsymbol{D}^{(i)} = \boldsymbol{D}^{(i)}\boldsymbol{A}$.
9: Randomly choose an unused vertex and let $\boldsymbol{c}^{(i)} \in \mathbb{R}^{w'h'}$ be its coordinates.
10: *// Move the sampled points to the chosen vertex.*
11: Compute $\boldsymbol{D}^{(i)} = \boldsymbol{D}^{(i)} + \boldsymbol{c}^{(i)}$, i.e., $\boldsymbol{c}^{(i)}$ is added to each row of $\boldsymbol{D}^{(i)}$.
12: Assign the rows of $\boldsymbol{D}^{(i)}$ with label i.
13: **end for**
// The second block: duplicating each dimension to introduce local correlation.
14: Duplicate each dimension of the initial data for p^2 times and reshape the results into two-dimensional patches with size $p \times p$.
15: Put the patches together and take crops to generate synthetic noises with size (w, h).
// Scale down the magnitude of synthetic data and harvesting perturbations.
16: Normalize each synthetic sample with L_2 norm bound ϵ.
17: Add synthetic perturbations to corresponding clean images.

In Algorithm 1, the synthetic images are scaled down before being used as perturbations. We visualize the synthetic perturbations and corresponding perturbed images in Figure 4. We also visualize the perturbations in Huang et al. [25] for a comparison. The details of perturbations can be found in Section 3.2. As shown in Figure 4, the synthetic perturbations do not affect data utility.

3.2 Synthetic Perturbations Are Highly Effective as Availability Attacks

Now we verify the effectiveness of synthetic perturbations and make comparisons with existing availability attacks. We perturb the entire training set following the main setup in previous works [12, 14, 25, 51]. That is, we synthesize a perturbation for every training example. In Section 4.1 and 4.2, we show synthetic perturbations are still effective when only partial training data are poisoned.

We use L_2-norm for synthetic perturbations to keep the sample-wise variation in the same class. We normalize the synthetic noises into a L_2-norm ball with radius $\sqrt{d}\epsilon'$, where d is the dimension of the input. We evaluate synthetic perturbations on three benchmark datasets: SVHN [32], CIFAR-10, CIFAR-100 [27], and a subset of ImageNet [38]. Following the setup in Huang et al. [25], we use the first 100 classes of the full dataset as the ImageNet subset. The target model architectures include VGG [45], ResNet [20], and DenseNet [24]. We adopt standard random cropping and flipping as data augmentation. The hyperparameters for training are standard and can be found in Appendix B. We use $\epsilon' = 6/255$ for synthetic perturbations. The patch size in Algorithm 1 is set as 8.

Table 2: Accuracy on clean test data of CIFAR-10. The target model is ResNet-18. The training data are poisoned with different attacks. The closer the accuracy to random guessing, the better the attack efficiency.

Algorithm	Test Accuracy (in %)
No Perturbation	94.69
TensorClog [44]	48.07
Alignment [13]	56.65
DeepConfuse [12]	28.77
NTGA [51]	33.29
Error-minimizing [25]	19.93
Adversarial Examples [14]	6.25
Synthetic Perturbations	13.54

We first compare synthetic perturbations with existing poisoning attacks. The comparisons are made on the CIFAR-10 dataset with ResNet-18 as the target model. We use the best-performing setup in their official implementations to generate perturbations. We present the comparison in Table 2. Synthetic perturbations are as powerful as advanced poisoning attacks despite they are much easier to generate.

Then we evaluate synthetic perturbations on different models and datasets. The test accuracy of target models is in Table 3. We also plot the training curves of target models on both clean and perturbed data in Figure 5. The results in Table 3 and Figure 5 further confirm the effectiveness of synthetic perturbations.

We note that generating synthetic perturbations is data irrelevant and only takes several seconds using a single CPU core. In contrast, existing attacks often need hours or even days to generate perturbations using GPUs. We compare the computational complexities of synthetic perturbations and recent attacks in Section 3.3.

In summary, our experiments demonstrate that using linearly-separable perturbations is indeed a sufficient condition for availability poisoning attacks to succeed. Moreover, these results also expose that deep models are very vulnerable to obscured shortcuts. This finding has two meanings to the community. First, it confirms that advanced availability poisoning attacks do succeed by providing shortcuts. Second, it further exposes the shortcut learning problem, which is a fundamental vulnerability of deep models.

3.3 Complexity Analysis

Here we give an analysis of the computational complexity of Algorithm 1. The complexity of generating synthetic perturbations is $O(nd/p^2)$, where n is the size of the dataset, d is the dimension of clean data, and p is the patch size. This complexity is mainly from introducing covariance into synthetic data (Line 6 in Algorithm 1).

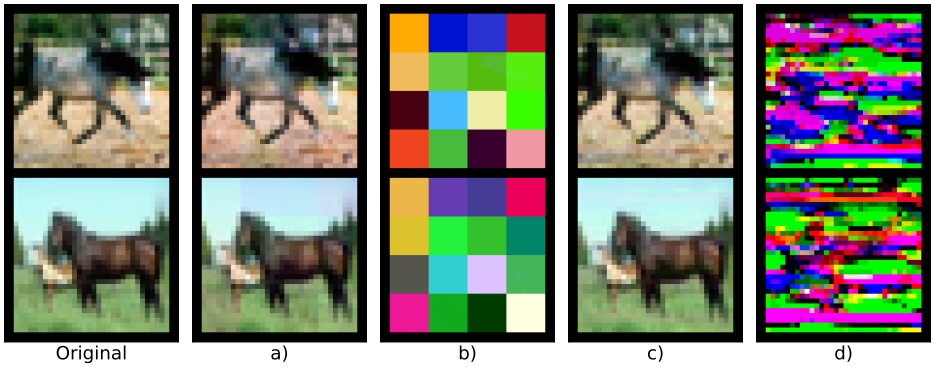

Figure 4: Visualization of perturbed images and normalized perturbations. Columns a) and b) use synthetic perturbations. Columns c) and d) use the attack in Huang et al. [25].

Table 3: Accuracy (in %) on clean test data. The target models are trained on clean data ($\mathbb{D}_c$) and data perturbed by synthetic perturbations ($\mathbb{D}_{syn}$).

Target Model	SVHN		CIFAR-10		CIFAR-100		ImageNet Subset	
	$\mathbb{D}_c$	$\mathbb{D}_{syn}$	$\mathbb{D}_c$	$\mathbb{D}_{syn}$	$\mathbb{D}_c$	$\mathbb{D}_{syn}$	$\mathbb{D}_c$	$\mathbb{D}_{syn}$
VGG-11	95.4	18.1	91.3	28.3	67.5	10.9	79.1	10.7
ResNet-18	96.2	8.0	94.7	13.5	74.8	9.0	79.7	11.0
ResNet-50	96.4	7.8	94.8	14.9	75.2	8.4	82.4	10.8
DenseNet-121	96.7	9.7	95.0	10.6	76.5	7.6	82.9	14.7

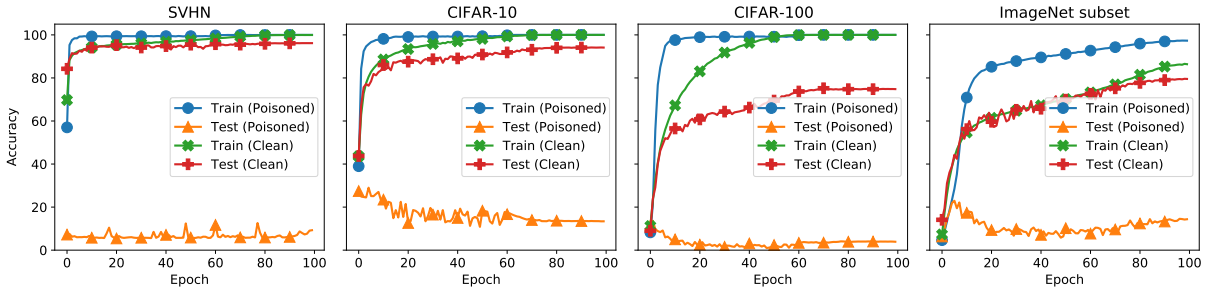

Figure 5: Training curves of ResNet-18 models on perturbed and clean data. The word 'poisoned' denotes the model is trained on perturbed data. The test performance is evaluated on clean data. The test accuracy is low throughout training when synthetic perturbations are added.

The complexity of generating synthetic perturbations is significantly smaller than those of recent attacks. The complexity of running the algorithms in recent attacks is $O(nTL(dw + w^2))$, where T is the number of iterations generating the poisons, L is the network depth, and w is the network width (for simplicity, we assume the network is of equal width). The term $nL(dw + w^2)$ is the cost of one forward/backward pass. This complexity is strictly worse than that of generating synthetic perturbations.

In recent attacks, the number of iterations generating the poisons is usually large. For example, Huang et al. [25] use multiple gradient descent steps to solve the optimization problems in Eq (2). On the ImageNet dataset, they first run 100 SGD updates for the outer problem. Then they loop over every target example to optimize the inner problem. They run 20 SGD updates for each example. The above process is repeated until the training accuracy is larger than a pre-defined threshold. Another example is the attack in Fowl et al. [14]. For each target example, they use 250 Projected Gradient Descent (PGD) steps to generate perturbations.

We now give an empirical comparison. We measure the time costs of generating error-minimizing noises, adversarial examples, and synthetic perturbations. The device is a server with a single Tesla V100 GPU and an Intel Xeon E5-2667 CPU. We note that running Algorithm 1 does not require a GPU. The time costs are tested on SVHN, CIFAR-10, and the ImageNet subset. The target

Table 4: Time costs (in seconds) of generating error-minimizing noises, adversarial examples, and synthetic perturbations.

Method	SVHN	CIFAR-10	ImageNet
Error-min. Noises	~2.7k	~3.5k	>28k
Adv. Examples	~3.3k	~4.1k	>30k
Algorithm 1	<3	<3	<3

Table 5: Test accuracy (in %) on clean ($\mathbb{C}$) and poisoned ($\mathbb{P}$) classes. Numbers of poisoned classes are 1, 3, and 5.

Method	1		3		5	
	$\mathbb{C}$	$\mathbb{P}$	$\mathbb{C}$	$\mathbb{P}$	$\mathbb{C}$	$\mathbb{P}$
Error-min. Noises [25]	94.6	2.4	93.9	1.1	93.8	3.1
Synthetic Perturbations	94.7	2.7	94.0	0.6	93.2	2.9

model is ResNet18. We use the configurations described in Huang et al. [25] and Fowl et al. [14] to generate error-minimizing noises and adversarial examples. For synthetic perturbations, we use the same setting as that in Section 3.2. The time costs are reported in Table 4. Generating synthetic perturbations is significantly cheaper than existing availability attacks.

4 EXPERIMENTS UNDER DIFFERENT SETTINGS

Here we test synthetic perturbations under different settings. We first consider two cases where not all the training data are poisoned. Although the main setting in previous works is to perturb the full training set, in practice we may only need to perturb part of the data. In the first case, we only apply Algorithm 1 to some of the classes. In the second case, we perturb partial data that are randomly sampled from all the classes. Finally, we run experiments on a face dataset following the application scenario in Huang et al. [25].

4.1 Poisoning Some Classes of The Training Data

In many practical datasets, some classes are more sensitive than others, e.g., in medical datasets, the patients that are diagnosed with a certain disease may be more concerned about their data than healthy people. We randomly sample some classes of the CIFAR-10 dataset and apply Algorithm 1 on all the examples of the sampled classes. After training, we report the test accuracy on clean classes and poisoned classes separately.

The synthetic perturbations are generated in the same way as that in Section 3.2. For comparison, we also run experiments with error-minimizing noises using those generated in Section 3.2. The experiments are run on ResNet-18. The numbers of poisoned classes are 1, 3, and 5. The poisoned classes are randomly chosen and are the same for two types of noises. We report the results in Table 5. Algorithm 1 is still highly effective when only some of the classes are poisoned.

Table 6: Test accuracy (in %) with different poisoning percentages p. Training with the poisoned subset does not improve the test accuracy much compared to training with clean data only.

Method	p =90%	p =80%	p =50%	p =20%
Clean Data ($1-p$)	82.6	86.5	92.4	93.9
Error-min. Noises [25]	85.2	86.8	92.8	94.1
Adv. Examples [14]	85.3	88.2	92.2	93.7
Synthetic Perturbations	85.7	86.3	92.9	94.0

4.2 Poisoning Different Percentages of The Training Data

Here we show synthetic perturbations remain effective when only a given percentage of the training data is poisoned. We follow the experimental setup in Fowl et al. [14], Huang et al. [25]. Specifically, for each poisoning percentage, we train two models. One model uses both the clean subset and the poisoned subset as its training data and the other one only uses the clean subset. The difference between the performances of those two models represents how much information the former model gains from the poisoned data. A small performance gap indicates the former model gains little information from the poisoned data.

We test four different poisoning percentages (from 20% to 90%) on the CIFAR-10 dataset. The experiments are run on ResNet-18 models. We compare the performance of synthetic perturbations with that of adversarial examples and error-minimizing noises [14, 25]. The results are presented in Table 6. The performance gain of using the poisoned subset is small for all three attacks. This suggests that synthetic perturbations are still effective in this setting.

4.3 Experiments on Face Data

Here we apply Algorithm 1 on face datasets. We follow the application scenario in Huang et al. [25] (see Figure 4 in Huang et al. [25] for an illustration). The task is to use face images to predict biological identities. We train an Inception-ResNet-v1 model [46] on the WebFace Dataset [50]. A random subset with 20% samples is used for testing and the remaining samples are used for training. The WebFace dataset has 10575 identities and 50 of them are poisoned. We run Algorithm 1 with the configuration in Section 3.2 ($\epsilon' = 6/255$) to process the training images of the poisoned identities.

When using Algorithm 1, the test accuracy of the poisoned identities is only 13.6% which is much lower than the test accuracy of the clean identities (>80%). The training curves are plotted in Figure 7. When using error-minimizing noises, the test accuracy of the poisoned identities reported in Huang et al. [25] is ~16%. These results confirm that Algorithm 1 is also highly effective on face data. We note that generating error-minimizing noises requires some auxiliary data. For example, Huang et al. [25] use 100 identities from the CelebA dataset [30] to generate error-minimizing noises for the 50 identities from the WebFace dataset. On the contrary, running Algorithm 1 does not require any auxiliary data.

5 ABLATION STUDY

In this section, we run some experiments to better understand our findings and the proposed algorithm. We first test whether the linear

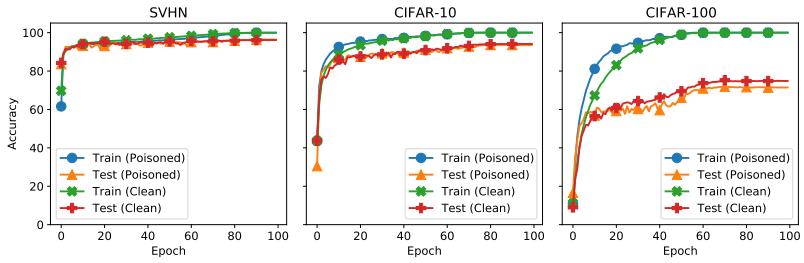

Figure 6: Training curves of ResNet18 models trained on SVHN, CIFAR-10, and CIFAR-100 datasets. The perturbations are NOT processed by the padding opeartion.

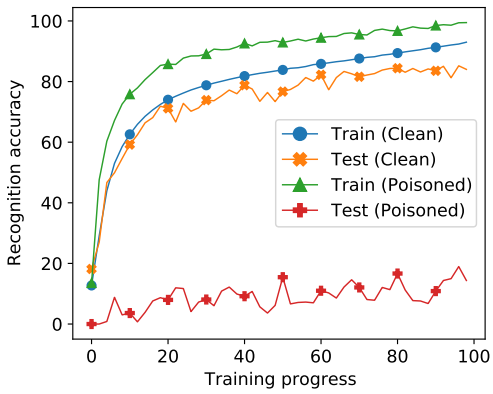

Figure 7: Training curves of an Inception-ResNet model on clean/poisoned WebFace.

separability is stemming from the property of the ReLU activation function. Then we demonstrate the padding operation in Algorithm 1 is necessary to make the perturbations robust against data augmentation methods.

5.1 Linear Separability Is Not Stemming from ReLU

It is well known to the community that a ReLU-DNN learns a piecewise linear function in input space [1]. The crafting models of advanced availability attacks all use the ReLU activation by default. Therefore, the linear separability may be stemming from the property of ReLU. To verify this, we replace the ReLU layers with Tanh layers in the crafting models of error-minimizing noises [25] and adversarial examples [14]. We fit the new perturbations with the same simple models as those in Section 2. The results are presented in Table 7. The new perturbations are still almost linearly separable: linear models achieve more than 90% training accuracy and two-layer neural networks achieve 100% training accuracy. This suggests that the linear separability is not associated with the ReLU activation function.

Table 7: Training accuracy (in %) of simple models on the perturbations of different attacks. The perturbations are generated with Tanh-DNNs.

Algorithm	Linear Model	Two-layer NN
Error-min. Noises [25]	100.0	100.0
Adv. Examples (Untargeted) [14]	92.7	100.0
Adv. Examples (Targeted) [14]	100.0	100.0

5.2 The Effect Of The Padding Operation in Algorithm 1

Here we explain why we duplicate each dimension of the initial data points into two-dimensional patches in Algorithm 1. Intuitively, it is more convenient to directly generate synthetic perturbations that have the same dimension as the original images. We will show this straightforward approach can not be used as a powerful attack.

We directly use the output of the method in Guyon [18] as the straightforward approach, i.e., the dimension of synthetic data is the same as the dimension of flattened images and we simply reshape the synthetic data into the image format. Other configurations are the same as those in Section 3. The models are trained with standard augmentation methods including random crop and flipping. The training curves of the target models are plotted in Figure 6. The test accuracy is still high when the data is poisoned, which does not meet the requirement of availability attacks.

The fact that the padding operation makes the perturbations remain effective may be because it introduces local correlation into the perturbations, which is an inherent property of natural images. In Appendix C, we show synthetic perturbations are still highly effective when more powerful data augmentation methods are applied.

6 CONCLUSION

This work gives an explanation of the working principle of availability poisoning attacks. We show advanced attacks coincidentally generate linearly-separable perturbations. We further synthesize linearly separable perturbations to demonstrate that using linearly separable perturbations is sufficient for an availability attack to succeed. The proposed algorithm is an order of magnitude faster than existing attacks. Our findings also suggest deep models are more prone to shortcuts than previously believed. They will find and heavily rely on shortcuts even when the shortcuts are scaled down to an imperceptible magnitude.

REFERENCES

[1] Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. 2016. Understanding deep neural networks with rectified linear units. *arXiv preprint arXiv:1611.01491* (2016).

[2] Marco Barreno, Blaine Nelson, Anthony D Joseph, and J Doug Tygar. 2010. The security of machine learning. *Machine Learning* (2010).

[3] Sara Beery, Grant Van Horn, and Pietro Perona. 2018. Recognition in terra incognita. In *Proceedings of the European conference on computer vision (ECCV)*.

[4] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2012. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389* (2012).

[5] Battista Biggio and Fabio Roli. 2018. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition* (2018).

[6] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2020. Extracting Training Data from Large Language Models. *arXiv preprint arXiv:2012.07805* (2020).

[7] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. 2020. Big Self-Supervised Models are Strong Semi-Supervised Learners. *arXiv preprint arXiv:2006.10029* (2020).

[8] Valeriia Cherepanova, Micah Goldblum, Harrison Foley, Shiyuan Duan, John Dickerson, Gavin Taylor, and Tom Goldstein. 2021. LowKey: leveraging adversarial attacks to protect social media users from facial recognition. *arXiv preprint arXiv:2101.07922* (2021).

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[10] Terrance DeVries and Graham W Taylor. 2017. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552* (2017).

[11] Khoa Doan, Yingjie Lao, and Ping Li. 2021. Backdoor Attack with Imperceptible Input and Latent Modification. *Advances in Neural Information Processing Systems* (2021).

[12] Ji Feng, Qi-Zhi Cai, and Zhi-Hua Zhou. 2019. Learning to confuse: generating training time adversarial data with auto-encoder. *arXiv preprint arXiv:1905.09027* (2019).

[13] Liam Fowl, Ping-yeh Chiang, Micah Goldblum, Jonas Geiping, Arpit Bansal, Wojtek Czaja, and Tom Goldstein. 2021. Preventing unauthorized use of proprietary data: Poisoning for secure dataset release. *arXiv preprint arXiv:2103.02683* (2021).

[14] Liam Fowl, Micah Goldblum, Ping-yeh Chiang, Jonas Geiping, Wojtek Czaja, and Tom Goldstein. 2021. Adversarial Examples Make Strong Poisons. *arXiv preprint arXiv:2106.10807* (2021).

[15] Jonas Geiping, Liam Fowl, W Ronny Huang, Wojciech Czaja, Gavin Taylor, Michael Moeller, and Tom Goldstein. 2020. Witches' brew: Industrial scale data poisoning via gradient matching. *arXiv preprint arXiv:2009.02276* (2020).

[16] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. 2020. Shortcut learning in deep neural networks. *Nature Machine Intelligence* (2020).

[17] Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Madry, Bo Li, and Tom Goldstein. 2020. Dataset Security for Machine Learning: Data Poisoning, Backdoor Attacks, and Defenses. *arXiv preprint arXiv:2012.10544* (2020).

[18] Isabelle Guyon. 2003. Design of experiments of the NIPS 2003 variable selection benchmark. In *NIPS 2003 workshop on feature extraction and feature selection*.

[19] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Conference on Computer Vision and Pattern Recognition*.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

[21] Katherine L Hermann and Andrew K Lampinen. 2020. What shapes feature representations? exploring datasets, architectures, and training. *arXiv preprint arXiv:2006.12433* (2020).

[22] Kashmir Hill. 2020. The secretive company that might end privacy as we know it. *The New York Times* (2020).

[23] Kashmir Hill and Aaron Krolik. 2019. How photos of your kids are powering surveillance technology. *The New York Times* (2019).

[24] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

[25] Hanxun Huang, Xingjun Ma, Sarah Monazam Erfani, James Bailey, and Yisen Wang. 2021. Unlearnable Examples: Making Personal Data Unexploitable. *International Conference on Learning Representations* (2021).

[26] Arthur Jacot, Franck Gabriel, and Clément Hongler. 2018. Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572* (2018).

[27] Alex Krizhevsky and Geoffrey Hinton. 2009. Learning multiple layers of features from tiny images. (2009).

[28] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. 2019. Fast autoaugment. *arXiv preprint arXiv:1905.00397* (2019).

[29] Dong C Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical programming* (1989).

[30] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild. In *Proceedings of International Conference on Computer Vision*.

[31] Preetum Nakkiran. 2019. A Discussion of 'Adversarial Examples Are Not Bugs, They Are Features': Adversarial Examples are Just Bugs, Too. *Distill* (2019).

[32] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. 2011. Reading digits in natural images with unsupervised feature learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning* (2011).

[33] Anh Nguyen and Anh Tran. 2020. Input-aware dynamic backdoor attack. *arXiv preprint arXiv:2010.08138* (2020).

[34] Anh Nguyen and Anh Tran. 2021. WaNet–Imperceptible Warping-based Backdoor Attack. *arXiv preprint arXiv:2102.10369* (2021).

[35] Timothy Niven and Hung-Yu Kao. 2019. Probing neural network comprehension of natural language arguments. *arXiv preprint arXiv:1907.07355* (2019).

[36] Vinay Uday Prabhu and Abeba Birhane. 2020. Large image datasets: A pyrrhic win for computer vision? *arXiv preprint arXiv:2006.16923* (2020).

[37] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*.

[38] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision* (2015).

[39] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. 2020. Hidden trigger backdoor attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[40] Avi Schwarzschild, Micah Goldblum, Arjun Gupta, John P Dickerson, and Tom Goldstein. 2021. Just how toxic is data poisoning? a unified benchmark for backdoor and data poisoning attacks. In *International Conference on Machine Learning*.

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

[42] Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. 2020. The pitfalls of simplicity bias in neural networks. *arXiv preprint arXiv:2006.07710* (2020).

[43] Shawn Shan, Emily Wenger, Jiayun Zhang, Huiying Li, Haitao Zheng, and Ben Y Zhao. 2020. Fawkes: Protecting privacy against unauthorized deep learning models. In *USENIX Security Symposium*.

[44] Juncheng Shen, Xiaolei Zhu, and De Ma. 2019. TensorClog: An imperceptible poisoning attack on deep neural network applications. *IEEE Access* (2019).

[45] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[46] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[47] Ruixiang Tang, Mengnan Du, Ninghao Liu, Fan Yang, and Xia Hu. 2020. An embarrassingly simple approach for trojan attack in deep neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

[48] Lue Tao, Lei Feng, Jinfeng Yi, Sheng-Jun Huang, and Songcan Chen. 2021. Provable Defense Against Delusive Poisoning. *arXiv preprint arXiv:2102.04716* (2021).

[49] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* (2008).

[50] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. 2014. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923* (2014).

[51] Chia-Hung Yuan and Shan-Hung Wu. 2021. Neural Tangent Generalization Attacks. In *International Conference on Machine Learning*. PMLR.

[52] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

[53] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2017. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412* (2017).

[54] Hengtong Zhang, Jing Gao, and Lu Su. 2021. Data Poisoning Attacks Against Outcome Interpretations of Predictive Models. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*.

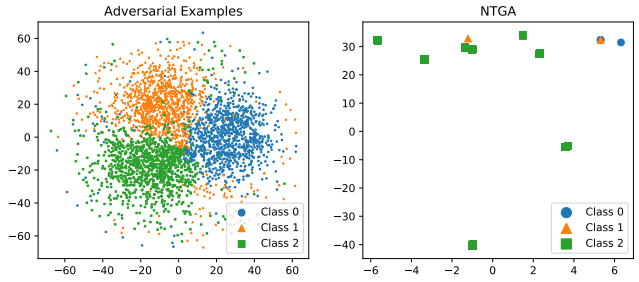

Figure 8: T-SNEs of targeted adversarial examples [14] and NTGA [51]. Perturbations from the same class are well clustered. Notably, many embeddings of NTGA are overlapped, suggesting that it uses very similar perturbations for some examples.

Appendix A ADDITIONAL T-SNE PLOTS

Here we plot the t-SNEs of two other attacks in Table 1, i.e., adversarial examples [14] and NTGA [51]. We use their official implementations to generate the perturbations (see Appendix B for details). The t-SNEs are plotted in Figure 8. The perturbations for the same class are well clustered. This observation is similar to that from Figure 2.

Appendix B IMPLEMENTATION DETAILS OF EXPERIMENTS

Implementation details of the experiments in Section 2. We generate perturbations of baseline algorithms using their official implementations: DeepConfuse[3], NTGA[4], error-minimizing noise[5], and adversarial examples[6]. The configuration is set to be the one that achieves the best attack performance on CIFAR-10. Specifically, DeepConfuse uses an 8-layer U-Net [37] as the crafting model. NTGA uses a 3-layer convolutional network. Error-minimizing noises and adversarial examples use standard ResNet-18 models.

The experimental setup for training the simple models is as follows. We train the simple models with standard cross-entropy loss. Before training, all perturbations are flattened into 1-dimensional vectors and normalized to unit norm. The two-layer neural networks have a width of 30. All models are trained with the L-BFGS optimizer [29] for 50 steps.

Implementation details of the experiments in Section 3. We use the Stochastic Gradient Descent (SGD) optimizer with a momentum coefficient 0.9 for all experiments. For all datasets, we use a batchsize of 128. The learning rates of all models are set to follow the choices in the original papers [20, 24, 45]. The learning rate for ResNet and DenseNet models is 0.1. The learning rate for VGG models is 0.01. All models are trained for 100 epochs. The learning rate is divided by 10 at epoch 50 and 75.

[3]https://github.com/kingfengji/DeepConfuse
[4]https://github.com/lionelmessi6410/ntga
[5]https://github.com/HanxunH/Unlearnable-Examples
[6]https://github.com/lhfowl/adversarial_poisons

Table 8: Test accuracy (in %) of ResNet18 models on the CIFAR-10 dataset.

	Cutout	Mixup	CutMix	FA
Error-min. Noises	18.9	57.4	32.3	41.6
Synthetic Perturbations	**10.6**	**39.5**	**17.7**	**24.4**

Appendix C TRAINING WITH MORE POWERFUL DATA AUGMENTATION METHODS

Here we demonstrate that synthetic noises can not be filtered out by state-of-the-art data augmentation methods. We test four advanced data augmentation methods including Cutout [10], Mixup [53], CutMix [52], and Fast Autoaugment (FA) [28]. Experimental results suggest that synthetic perturbations are still highly effective when those augmentation methods are applied.

We train ResNet18 models on the CIFAR-10 dataset. For all augmentation methods, we use the default configurations for CIFAR-10 from the original papers to set their parameters. Other experimental settings such as the noise strength and training recipe are the same as those in Section 3.2. The results are presented in Table 8. In Table 8, we also include the test accuracy of using error-minimizing noises for a comparison. The results suggest that synthetic perturbations are more effective when advanced augmentation methods are applied. For example, when Fast Autoaugment (FA) is applied, the test accuracy of using synthetic perturbations is 24.4% while the test accuracy of using error-minimizing noises is 41.6%.

Appendix D REPRODUCIBILITY STATEMENT

We upload our source code with README files in the supplement. The README files provide example commands to verify our findings. The code will also be published after the reviewing process. Our implementation is based on Pytorch[7], which is a popular open-source machine learning framework.

[7]https://pytorch.org/