Machine Learning as an Adversarial Service: Learning Black-Box Adversarial Examples

Jamie Hayes
University College London
j.hayes@cs.ucl.ac.uk

George Danezis
University College London
g.danezis@ucl.ac.uk

***Abstract*—Neural networks are known to be vulnerable to *adversarial examples*, inputs that have been intentionally perturbed to remain visually similar to the source input, but cause a misclassification. Until now, *black-box* attacks against neural networks have relied on *transferability* of adversarial examples. White-box attacks are used to generate adversarial examples on a substitute model and then transferred to the black-box target model. In this paper, we introduce a direct attack against black-box neural networks, that uses another *attacker* neural network to learn to craft adversarial examples. We show that our attack is capable of crafting adversarial examples that are indistinguishable from the source input and are misclassified with overwhelming probability - reducing accuracy of the black-box neural network from 99.4% to 0.77% on the MNIST dataset, and from 91.4% to 6.8% on the CIFAR-10 dataset.**

Our attack can adapt and reduce the effectiveness of proposed defenses against adversarial examples, requires very little training data, and produces adversarial examples that can transfer to different machine learning models such as Random Forest, SVM, and K-Nearest Neighbor. To demonstrate the practicality of our attack, we launch a live attack against a target black-box model hosted online by Amazon: the crafted adversarial examples reduce its accuracy from 91.8% to 61.3%. Additionally, we show attacks proposed in the literature have unique, identifiable distributions. We use this information to train a classifier that is robust against such attacks.

1. Introduction

Machine Learning models are increasingly relied upon for safety and business critical tasks such as in medicine [24], [31], [40], robotics and automotive [29], [33], [38], security [2], [16], [37] and financial [12], [17], [35] applications. Recent research shows that machine learning models trained on entirely uncorrupted data, are still vulnerable to *adversarial examples* [8], [11], [26], [27], [34], [36]: samples that have been maliciously altered so as to be misclassified by a *target* model while appearing unaltered to the human eye.

We focus on the susceptibility of neural networks to adversarial examples, and study neural networks applied to image classification, known to outperform other machine learning approaches. Since Szegedy *et al.* [34] first drew attention to adversarial examples, there has been an arms race within the research community to defend against them or develop attack methods to generate them. However, current attacks have several limitations.

Almost all proposed attacks assume *white-box* access to the models to be attacked. In other words, the attacker is assumed to have complete knowledge and control of the network's weight and architecture. With this insider knowledge, a *white-box* attack normally performs gradient descent to transform a source to an adversarial image. Until now, almost all algorithms that craft adversarial examples are designed by a human expert in machine learning.

In this work, we introduce the first probabilistic, machine learning based *black-box* attack against neural networks. The attack does not require internal model access nor expert knowledge to craft adversarial examples. It operates by constructing a machine learning model that learns how to perturb an image such that it will be misclassified by a neural network while remaining visually similar to the source image. Thus we leverage machine learning itself as an attack tool against machine learning systems.

We show that the training process of the attack can be augmented to produce adversarial examples in a *targeted* or *untargeted* attack: in a targeted attack, the attacker chooses the class that they would like the adversarial example to be misclassified as; while in an untargeted attack it suffices to produce *any* misclassification. Both settings produce excellent adversarial examples.

Other black-box attacks rely on white-box attacks that transfer across models [27], or require full access to the training dataset and are computationally expensive [21]. Our black-box attack requires no such transferability property to hold, does not rely on access to training data, and is efficient to run. We show that direct, efficient black-box attacks are possible, and the attacker only requires a weak signal from which a model can learn to craft adversarial examples.

A summary of our key contributions are:

1) In Section 3, we introduce our machine learning based attack. The attack does not need internal access to the target model, requiring access to only the confidence scores placed on each class, as returned by major ML service providers.
2) In Section 4, we evaluate untargeted and targeted attacks, showing that they succeed in crafting quality adversarial examples.
3) In Section 5, we demonstrate that our attack does not need full access to the training data to successfully craft adversarial examples. An attacker with access to just 1% of the MNIST training set, 600 data samples, can craft adversarial samples that reduce test set accuracy from 99.4% to 12.8%.
4) In Section 6, we evaluate attack efficacy on noisier images. We find that the target model fails to filter out insignificant background information increasing the impact of our attacks.
5) In Section 7, we show that our attack is capable of adapting and weakening the robustness of two popular adversarial example defenses [10], [39].
6) In Section 8, we show that adversarial examples crafted by the attack can successfully transfer to machine learning models other than neural networks such as random forests, support vector machines, and k-nearest neighbor.
7) In Section 9, we compare our attack against other state-of-the-art black-box attacks, showing our attack is more efficient and as successful. We also launch, and evaluate, a successful live attack on the *Amazon Machine Learning Prediction API* tool [1].
8) Finally, in Section 10, we offer some hope in defending against attacks. We discover that there are statistical differences between source and adversarial examples, and it is possible to separate different attacks from each other. We show that a target model trained with auxiliary classes detects and remains robust against all state-of-the-art attacks.

2. Background

We define adversarial examples along with some terminology and notation. We then introduce the threat model considered, and the datasets we use to evaluate the attacks.

2.1. Adversarial Examples

Szegedy *et al.* [34] casts the construction of adversarial examples as an optimization problem. Given a *target model*, f, and a *source* input x, which is sometimes referred to in the literature as a *clean* or *original* input. The input x is classified correctly by f as y. The attacker aims to find an *adversarial* input, $x + c$, which is perceptually identical to x but $f(x + c) \neq y$. Then the adversary attempts to find $\mathrm{argmin}_c\, f(x + c) = l$ such that $x + c$ is a legitimate input and $f(x) \neq l$. The problem space can be extended to find a specific misclassification in a *targeted* attack, or *any* misclassification, referred to as an *untargeted* attack.

In the absence of a distance measure that accurately captures the perceptual differences between a source and adversarial examples, the L_p distance metric is used in [34] as a measure of similarity. The L_0 distance measures the number of pixels that have changed, the L_2 distance measures the Euclidean distance between two images, while the L_∞ distance measures the maximum perturbation between two pixels. We optimize under the L_2 distance metric, however we also report the maximum perturbation values since it is an important measure of similarity.

2.2. Threat Model

We consider an attacker who wishes to craft inputs to be classified incorrectly by a target model: as any incorrect class for untargeted attacks or a class of their choice in targeted attacks. We also constrain the adversary input by the attacker to be visually imperceptible to a source input, evaluated through a distance measure (L_0, L_2 or L_∞) .

Our attacks assume only black-box access to the target model, the attacker feeds in an adversarially crafted sample to the target model and as output, receives confidences scores for each of the possible classes. The black-box setting is the strongest, in that it assumes no attacker knowledge of the type of model, architecture, weights or hyperparameters. If a black-box attack succeeds in fooling a target model, we can be confident that an attack with white-box access will perform equally as well or better. Black-box access is particularly relevant when targeting Machine Learning as a Service (MLaaS) systems, such as Amazon [2], Clarifai [3] and Google [4], that do not provide white-box model information.

We also evaluate our attacks in two settings in which the target model is defended against adversarial inputs by a mechanism, D. In the first setting, we consider an oblivious attacker that is not aware that the target model is defended by D, and moreover, is not aware that any defense is in place. In the second setting, we consider an attacker that has knowledge of the defense D.

In almost all experiments, we consider a *worst-case* scenario with respect to data access, assuming that the attacker has knowledge of, and shares access to, any data samples that the target model was trained with. However, in Section 5 we restrict the attacker to knowing only a few data samples to craft an attack.

2.3. Datasets

We evaluate attacks using two popular datasets in adversarial machine learning research, MNIST [5] and CIFAR-10 [14].

The MNIST dataset consists of $70,000$, 28×28 grayscale images of handwritten numbers ranging from 0 to 9. The dataset is split into $60,000$ training set images and $10,000$ test set images. Our pre-trained model, used as the target model, scores 99.4% accuracy, which is comparable with state-of-the-art classification results on MNIST.

[1] https://aws.amazon.com/machine-learning/

[2] https://aws.amazon.com/machine-learning/

[3] https://www.clarifai.com/

[4] https://cloud.google.com/products/machine-learning/

[5] http://yann.lecun.com/exdb/mnist/

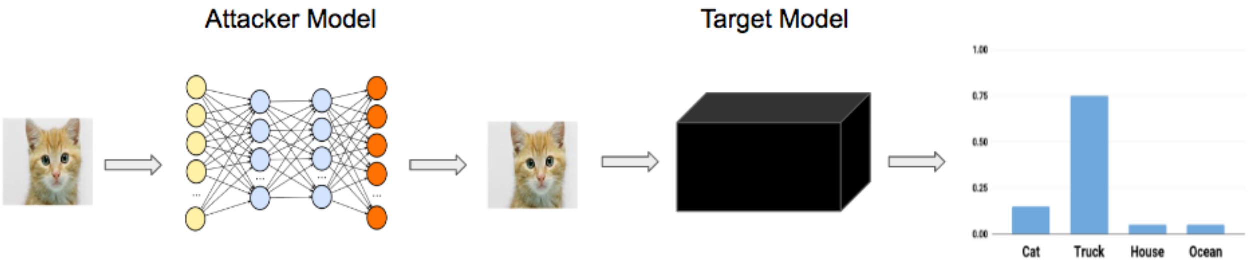


Figure 1: Overview of the attack. An input which is classified correctly by the target model is fed into the attacker model. The attacker model adds perturbations to the source image such that the output image is visually imperceptible to the source, but causes a misclassification when given to the target model.

The CIFAR-10 dataset consists of $60,000$, 32×32 RGB images of different objects in ten classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck. Due to the complexity of the images over MNIST images, our pre-trained model, a *VGG-19* neural network [32], scored 91.6% accuracy. State-of-the-art results on CIFAR-10 are approximately 95% accurate.

Some works also reports results on ImageNet [30]. However, previous work [3] shows the distortion required to craft adversarial samples on ImageNet is substantially smaller than on MNIST (10× smaller) and on CIFAR-10 (3× smaller). Consequently, crafting adversarial examples is a simpler task on ImageNet and so we elect to report results on more difficult datasets.

3. Machine Learning as an Adversarial Service

3.1. Attack Description

An overview of the attack is given in Figure 1. An attacker is provided black-box access to a target machine learning model, and they attempt to craft a sample input that would lead to a misclassification. Upon feeding the target model an input, the adversary can observe the output label, and the confidence of the categorization across all labels. Our attack is optimized directly through this output by the black-box target model. It proceeds, by training another model, named the *attacker* model, to learn how to transform a source input image into an adversarial image, based on those classification scores output by the target model [6]. We note that the assumption that an attacker has access to confidence scores is not unrealistic; both Amazon and Google, along with numerous other companies, offer black-box machine learning prediction APIs that output prediction probabilities.

To learn to craft adversarial examples the attacker model is trained to minimize the distance between the source and adversarial image while causing a misclassification in the target model. The attacker model accepts a source image as input and outputs an adversarial image. The best method for minimizing distance between an input and output is for the attacker model to learn the identity function, however this will clearly lead to both the source and adversarial image being assigned the same, correct, label by the target model – namely no attack. Therefore we encode a loss function that minimizes the distance subject to a difference in classification between source and adversarial image.

The exact formulation of loss function depends on whether the attack is targeted or untargeted. In an untargeted attack, the attacker model will promote *any* class [7] ahead of the source sample class, whereas in the targeted attack the attacker model will promote the class chosen by the attacker.

More formally, given an attacker model $\mathcal{A}$ and a target model $\mathcal{T}$, we define the loss of the attacker model as a combination of reconstruction success and misclassification success:

$$Loss = \alpha \cdot Loss_{\mathcal{A}} + Loss_{\mathcal{T}_{\mathcal{A}}},$$

where $Loss_{\mathcal{A}}$ is the attacker model loss, $Loss_{\mathcal{T}_{\mathcal{A}}}$ is the black-box target model loss as computed by the attacker, and $\alpha \in \mathbb{R}_{\geq 0}$ is a weight term that places importance on either the attacker loss or the target model loss, referred to as the *attack weight*. $Loss_{\mathcal{A}}$ can also be viewed as the reconstruction term, transforming a source image into an adversarial image. While $Loss_{\mathcal{T}_{\mathcal{A}}}$ can be thought of as a distance measure between the target model classification on source images and adversarial images. The choice of α measures how much weight we place on the reconstruction term. A small α results in a high number of successfully crafted adversarial examples at the expense of large differences from the source images, whereas a large α has the reverse effect.

We define $Loss_{\mathcal{A}}$ as a distance measure between inputs to and outputs from, the attacker model. We saw best results when using the mean squared error (MSE) as the reconstruction measure. The formulation of $Loss_{\mathcal{T}_{\mathcal{A}}}$ is dependent on the type of target. In an untargeted attack, the attacker maximizes the difference between the target model predictions on source images and adversarial images. Maximizing the prediction differences between the two forces the attacker model to

[6] Source code will be made available.

[7] Normally, this is the second most confident class placed on the source image.

TABLE 1: Attacker model architectures for the MNIST and CIFAR-10 dataset experiments.

Layer	Attacker Model	
	MNIST	CIFAR-10
Conv + Batch Norm + ReLU	$10 \times 24 \times 24$	$10 \times 28 \times 28$
Conv + Batch Norm + ReLU	$20 \times 20 \times 20$	$20 \times 24 \times 24$
Conv + Batch Norm + ReLU	$40 \times 18 \times 18$	$40 \times 22 \times 22$
Conv + Batch Norm + ReLU	$80 \times 16 \times 16$	$80 \times 20 \times 20$
DeConv + Batch Norm + ReLU	$40 \times 20 \times 20$	$40 \times 24 \times 24$
DeConv + Batch Norm + ReLU	$20 \times 24 \times 24$	$20 \times 28 \times 28$
DeConv + Batch Norm + ReLU	$10 \times 26 \times 26$	$10 \times 30 \times 30$
DeConv	$1 \times 28 \times 28$	$3 \times 32 \times 32$

TABLE 2: Target model architecture for the MNIST dataset experiment.

Layer	MNIST Target Model
Conv	$10 \times 24 \times 24$
Max Pool + ReLU	$10 \times 12 \times 12$
Conv	$20 \times 8 \times 8$
Dropout	$20 \times 8 \times 8$
Max Pool + ReLU	$20 \times 4 \times 4$
Reshape	320
Linear + ReLU	50
Dropout	50
Linear	10
Log Softmax	10

TABLE 3: Model hyperparameters. Target model hyperparameters for the CIFAR-10 dataset experiment are given in Simonyan & Zisserman [32].

Parameter	Attacker Model		Target Model
	MNIST	CIFAR-10	MNIST
Learning Rate	$2 \cdot 10^{-4}$	$2 \cdot 10^{-4}$	0.01
Momentum	-	-	0.5
Beta 1	0.5	0.5	-
Beta 2	0.999	0.999	-
Dropout	-	-	0.5
Batch Size	64	64	64
Epochs	100	100	50

perturb the source image such that it is a misclassification by the target model. In practice, we minimize:

$$-|p_{adv} - p_{source}|$$

where, p_{adv} are the adversarial image predictions and p_{source} are the source image predictions. This is equivalent to maximizing the prediction differences between the source and adversarial image.

To construct a targeted attack, the attacker minimizes the difference between predictions on adversarial images, and a template prediction. This template prediction is equal to the adversarial example prediction everywhere except the target class index, which is given a value marginally greater than the maximum value in the array of predictions, if the target class index does not already contain the maximum value. This causes the target model prediction of an adversarial image to be maximized around the target class. More specifically, given a target class t, from n classes, and a prediction $p_{adv} = (\gamma_0,\ \gamma_1,\ \cdots,\ \gamma_t,\ \cdots,\ \gamma_{n-1})$ where γ_i is the target model confidence on class i, we compute a new template prediction:

$$p^*_{adv} = \begin{cases} p_{adv} \text{ if } \arg\max_i \gamma_i = t \\ (\gamma_0,\ \gamma_1,\ \cdots,\ \max(p_{adv}) + \delta,\ \cdots,\ \gamma_{n-1}) \text{ otherwise} \end{cases}$$

where $\delta << 1$. We then optimize on the difference between p_{adv} and p^*_{adv}. This has the effect that if our adversarial image has been classified as the target class then the loss is equal to zero and no further optimization is required, otherwise $|\gamma_t - (max(p_{adv}) + \delta)|$ is minimized. This forces the target model to increase its confidence for the target class. We introduce the small value δ to ensure the maximum value is found at the target class index.

3.2. Model Description

Here, we give a description of both the attacker model and target model used throughout the paper.

Architectures for the MNIST and CIFAR-10 attacker models are identical and are given in Table 1. The first four layers of the model is an *encoder*, consisting of convolutions mapping an image into a feature representation, while the final four layers is a *decoder*, deconvolving the feature representation back into the input space. Convolutional autoencoders have shown to produce state-of-the-art results in image feature representation learning [19] and so are a good fit for our use case - learning to reconstruct a source image with some small perturbations. The target model architecture for MNIST experiments is given in Table 2, while we use the *VGG-19* model [32] as a target model for CIFAR-10 experiments. Both target models achieve close to state-of-the-art results on their respective datasets. Table 3 gives hyperparameters for attacker and target models. The MNIST target model is trained using a SGD optimizer while the attacker model is trained using the Adam optimizer [13]. In all experiments the target model is pre-trained before the attack is launched. The attacker model is trained on the MNIST or CIFAR-10 training dataset, and results reported from the test set.

4. Untargeted & Targeted Experiments

We trained target models classifying MNIST and CIFAR-10 with an accuracy of 99.4% and 91.4%, respectively. This is comparable to state-of-the-art classification results on both datasets. For the MNIST dataset, the attacker model was then trained on the MNIST training set and results are reported on the MNIST test set (with equivalent splits for the CIFAR-10 dataset).

4.1. Untargeted Attack

Figure 2 shows successful adversarial examples from the MNIST test set, 99.4% of source samples were correctly classified by the target model while all adversarial samples in the figure were incorrectly classified. As expected, as the attack weight is reduced, adversarial sample quality also decreases. At $\alpha = 80$ and $\alpha = 40$, adversarial examples

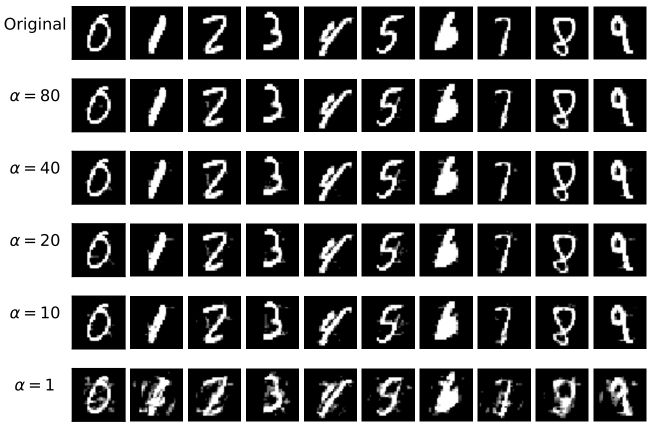


Figure 2: Examples of successful adversarial images constructed under different attack weights for the MNIST dataset. The probability of an adversarial image fooling the target model is 40.2%, 54.7%, 73.2%, 80.9%, and 99.23% for attack weights 80, 40, 20, 10, and 1, respectively.

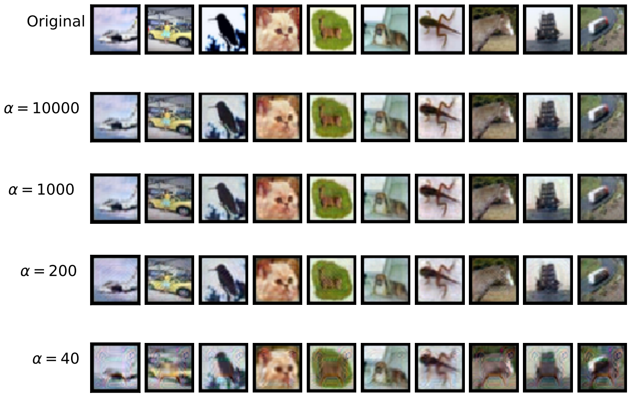


Figure 3: Examples of successful adversarial images constructed under different attack weights for the CIFAR-10 dataset. The probability of an adversarial image fooling the target model is 19.7%, 20.2%, 71.3%, and 83.6% for attack weights 10000, 1000, 200, and 40, respectively.

are almost indistinguishable from their source counterparts, however the attacker model only succeeded in creating a misclassified image with probability 40.2% and 54.7%, respectively. At $\alpha = 10$, the attacker model succeeds with overwhelming probability (80.9%), while image quality only slightly degrades. As we decreased α further, image quality suffered to the extent where the attack utility is negligible since adversarial examples look nothing like their source counterpart.

Figure 3 give examples of successfully crafted adversarial examples by the attacker model for the CIFAR-10 test set. While α values differ from MNIST experiments, we see that the attacker is still able to craft visually similar successful adversarial examples, with probability of success dependent on α. As we lowered α, the misclassification rates increased: at $\alpha = 40$ the target model misclassified at a rate of 83.6%, as we continued to lower α our best results, while keeping adversarial images visually similar, were successful in fooling the target model with a probabiliity of 93.2%. CIFAR-10 experiments required an α value ten times that of MNIST experiments to achieve similar success probabilities. In other words, CIFAR-10 images required less distortion from the source image to fool the target model.

For attack weights reported in Figure 2 and Figure 3, we give the corresponding perturbation levels and success probabilities as the attacker model was trained, in Figure 4. The perturbation value, ϵ, is given by the L_∞ distance metric, defined as the maximum difference between pixels of the source and corresponding adversarial image [8]. Clearly as the attacker model is trained, it learns to craft visually similar images (the perturbation level decreases over time) and learns to fool the target model (the probability of an adversarial image successfully fooling the target model increases over time). However, Figure 4g and Figure 4h show that an extremely large α value completely inhibits the attacker models capacity to learn to craft successful adversarial images. This is to be expected since a large α effectively removes $Loss_{\mathcal{T}_\mathcal{A}}$ from the loss function, resulting in the attacker model optimizing exclusively for visually similar images. We give the attacker model loss, $Loss_\mathcal{A}$, defined by the mean squared error (MSE), in Appendix A.

4.2. Targeted Attack

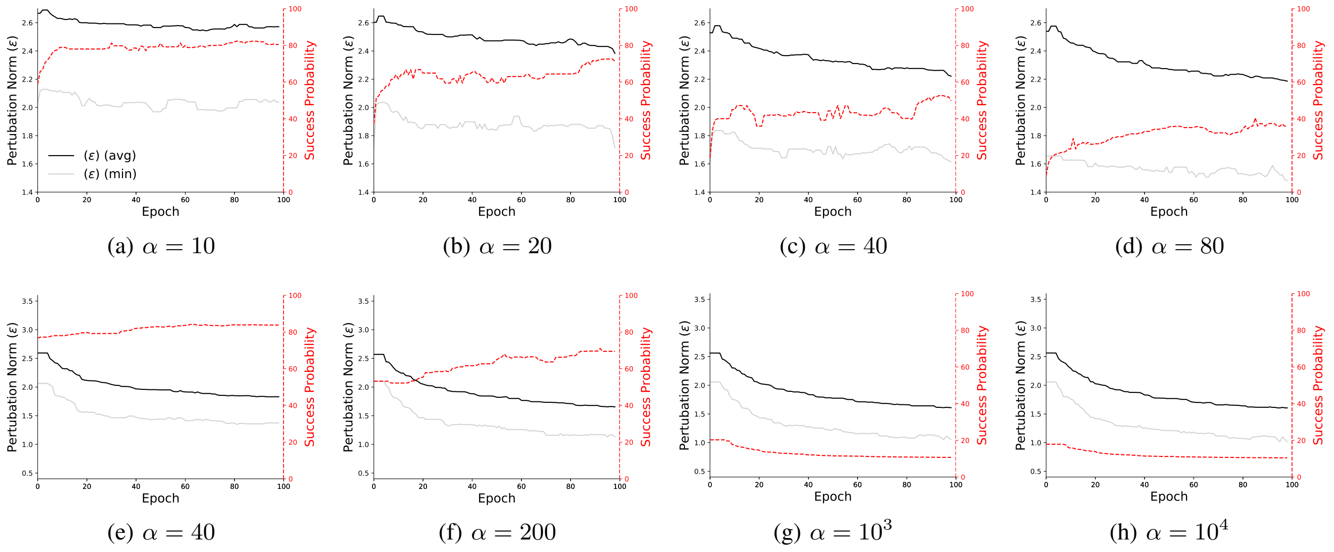
We follow the same experimental set-up as in the untargeted experiments, however now, given an input, the attacker chooses a class, c, they would like the target model to classify an adversarial example as, and success is calculated as the probability that an adversarial example is classified as c. Figure 5 and Figure 6 show all possible source-adversarial target pairs for the MNIST and CIFAR-10 datasets, with an attack weight of $\alpha = 10$ and $\alpha = 40$, respectively. The attack is capable of transforming any source image to be misclassified as any chosen class, while remaining visually similar to the source image, and we achieve similar success probabilities to untargeted experiments.

5. Does the Attacker Need Full Data Access?

So far, we have assumed the attacker shares full access to any data samples that were used to train the target model. However in practice, this may not be the case - a black-box prediction model may only publicly share the type or subsample of the training data. We therefore evaluate our untargeted attack under stronger assumptions of attacker access to training data - ultimately showing that the attack does not suffer from training on fewer data samples.

Figure 7 shows the target model accuracy on the MNIST test set of (a) source images, (b) adversarial images when the attacker model has been trained on the full training set, (c) adversarial images when the attacker model has been trained on a subset of the training set. The target model is trained on subsets of the MNIST training set of various

[8] For example, the ϵ value between a grayscale white image and a grayscale black image would be equal to 255.



(a) $\alpha = 10$ (b) $\alpha = 20$ (c) $\alpha = 40$ (d) $\alpha = 80$

(e) $\alpha = 40$ (f) $\alpha = 200$ (g) $\alpha = 10^3$ (h) $\alpha = 10^4$

Figure 4: Success probability of adversarial images and perturbation levels as the attacker model is trained for various attack weights. MNIST results are given in (a), (b), (c), (d), CIFAR-10 results are given in (e), (f), (g), (h).

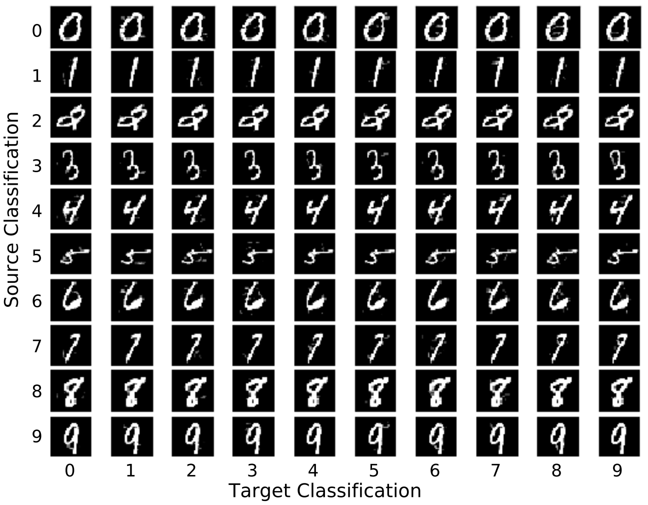


Figure 5: Matrix of targeted attack on every source-target pair. The source images were randomly selected from the MNIST test set.

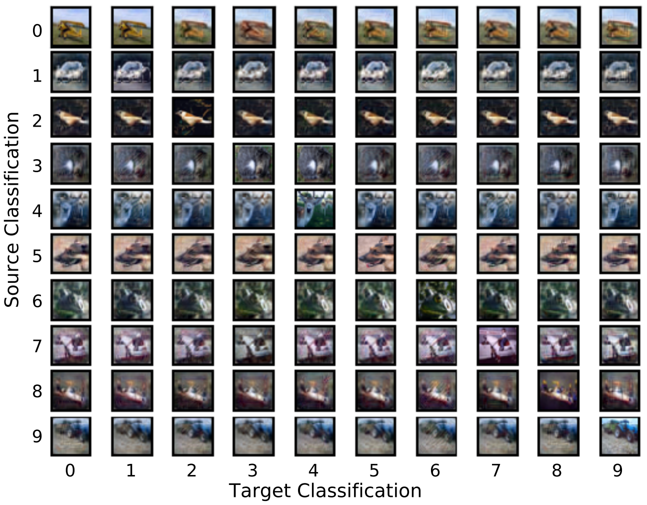


Figure 6: Matrix of targeted attack on every source-target pair. The source images were randomly selected from the CIFAR-10 test set.

sizes. As expected, training on more data samples improves classification accuracy; the target model is 73.6% accurate when trained on 0.1% of the MNIST training set, while it is 99.2% accurate when trained on 80% of the MNIST training set - in other words, there is virtually no difference in test accuracy when training on between 80-100% of the training set. There is also no significant difference in adversarial success between an attacker model that has been trained on many data samples and few data samples. For example, attacking a target model (that was trained on 99% of the MNIST training set) by training the attacker model on the full MNIST training set and on just 1% of the MNIST training set produces adversarial examples that reduce target model accuracy to 12.2% and 12.8%, respectively. Similarly, attacking a target model (that was trained on 80% of the MNIST training set) by training the attacker model on the full MNIST training set and on just 20% of the MNIST training set yields target model accuracies of 10.2% and 6.5%, respectively. The amount of data samples provided to the attacker model does not significantly impact its ability to learn to craft adversarial examples, all that must be known is the structure of the dataset on which the target model was trained. We note that this is in agreement with Papernot *et al.'s* [27] findings on the number of source samples required to launch a practical black-box attack.

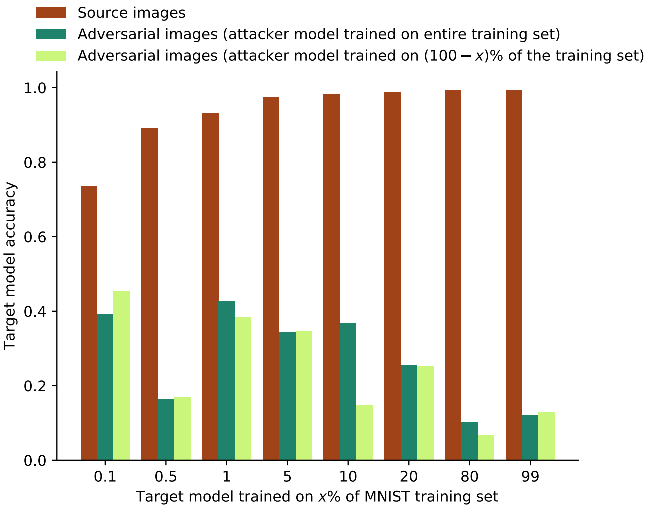


Figure 7: A comparison of target model accuracy on source and adversarial images when the target model is trained on subsets of the MNIST training set, the attacker model is trained on the full MNIST training set, and the attacker model is trained on subsets of the MNIST training set.

6. Perturbed Backgrounds

An adversarial image that is not distinguishable from its source counterpart must not significantly alter background pixels in the MNIST dataset, due to the clean composition of the images. We expect that as the background is perturbed more in source images, the ability to craft adversarial examples will also improve. In other words, adversarial example crafting is easier on noisier, more complex images. Empirically, this is supported by our superior results on CIFAR-10 over MNIST. To verify this hypothesis we randomly flipped background pixels (from black to white) in the MNIST dataset at rates of 5%, 10%, 25% and 50%. Figure 8 shows a sample of background perturbed MNIST datasets, clearly for low rates of background perturbations, it is still practical for a human to visually classify the correct digit. Furthermore, the target model does not dramatically suffer in classification accuracy as more background pixels are flipped; at a background perturbation rate of 5% the test set accuracy is 99.1%, while at a 50% perturbation rate the test set accuracy is 94.1% (see Table 4). We also note that a *perfect* target model would not use any information in the background from which to separate classes, since background pixels were flipped at random, and so have no underlying structure a model could exploit.

Figure 9 gives an example of successful adversarial examples crafted from noisy source images, for various attack weights at background perturbation rates of 5% and 10%, in an untargeted attack setting. Firstly, we found that different attack weights effect the final classification. Smaller weights resulted in 3's being classified as 8's while larger attack weights resulted in predictions of 0's, 2's or 8's.

We define *adversarial pixels* as the pixels that differed from the perturbed source image and corresponding adversarial images - pixels that caused the misclassification. Figure 9 shows that nearly all adversarial pixels are located adjacent to the digit. This make sense, because the structure that was available to the target model to learn distinct classes are the pixels making up the digit, and so these pixels have the largest impact on final classification.

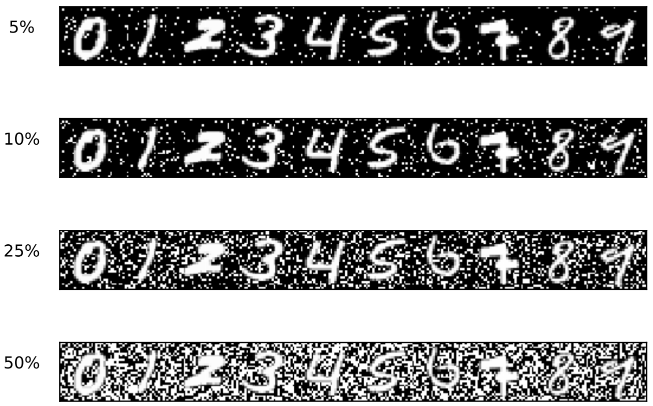


Figure 8: MNIST with different levels of background flipped pixels.

Table 4 shows, for different background perturbation rates: (1) the target model classification accuracy of the source and adversarial images, (2) the number of adversarial pixels (as a percentage of the total number of pixels), and (3) the number of adversarial pixels that are adjacent to foreground digit pixels (as a percentage of the total number of pixels). Firstly, the target model accuracy on source images is high. At 5% background perturbation, the reduction in accuracy is 0.3% (from 99.4% to 99.1%), while at 50% background perturbation, the reduction in accuracy is 5.3% (from 99.4% to 94.1%). The target model accuracy for adversarial examples crafted from perturbed source images is lower (for all weights) than for all adversarial examples crafted from non-perturbed source images. The percentage of required pixels to cause a misclassification decreases as we increase the rate of background perturbation and the attack weight. In other words, as expected, the attacker model can more easily exploit the target model when images are noisier, and when images are more visually similar to the source image. The number of adversarial pixels that are adjacent to foreground source digit pixels decreases as we increase the background perturbation rate. For example, with an attack weight of ten and 50% background pixels flipped, only 57.8% of adversarial pixels are adjacent to source digit - the attacker model has learned to exploit background information that has no relevance to the foreground digit, to cause a misclassification.

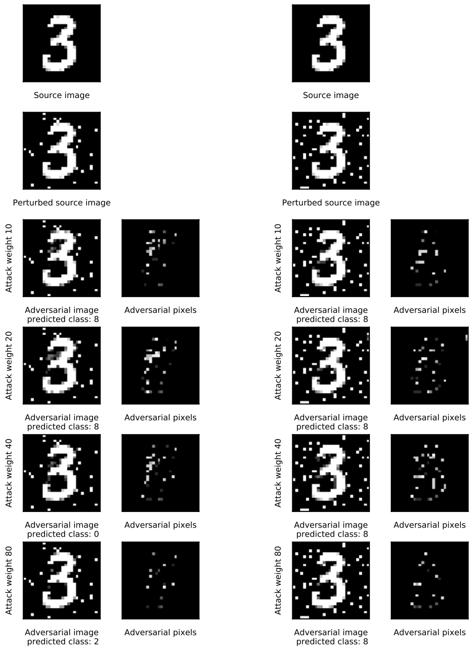
7. Attack on Defenses

We now turn our attention to attacking defenses against adversarial examples. We experiment by attacking two popular defenses: *PCA whitening* [10] and *Feature Squeezing* [39], on the MNIST dataset.

Hendrycks & Gimpel's [10] PCA whitening method

TABLE 4: MNIST background perturbation experiments

Experiment Type	**Image Type**	**Attack weight (α)**	**Background Pertubation (%)**			
			5	10	25	50
Target Model Accuracy (%)	**Source Images**	-	99.1	98.6	96.9	94.1
	Adversarial Images	10	18.1	16.5	15.3	12.2
		20	36.5	22.5	21.8	19.9
		40	46.8	29.8	23.1	23.5
		80	54.2	42.3	29.4	22.1
Adversarial Pixels (%)	**Adversarial Images**	10	4.29	3.74	4.20	3.58
		20	3.72	3.59	2.97	3.11
		40	3.23	3.57	2.86	2.77
		80	2.91	2.90	2.74	2.66
Adjacent Adversarial Pixels (%)	**Adversarial Images**	10	89.6	88.8	64.1	57.8
		20	90.1	86.2	79.1	65.1
		40	92.8	89.6	78.5	66.3
		80	93.0	90.7	79.9	69.4



(a) 5% (b) 10%

Figure 9: Examples of adversarial images and the corresponding pixels that caused misclassification (adversarial pixels) on different levels of background perturbations.

takes image training data as input, centers the data around zero, and finds the covariance matrix C. They then find the singular value decomposition (SVD) of C by finding $C = U\Sigma V^{*}$, and perform 'PCA whitening' by finding $\Sigma^{\frac{-1}{2}} U^{T} x$, given an input image x. PCA coefficients for adversarial examples were found to have consistently greater variance in comparison to non-adversarial images, and so is used as a detection method. Using the Fast Gradient Sign Method [8] to create adversarial images from both the MNIST and CIFAR-10 datasets, they were able to separate adversarial and source images with 100% success. The authors remark that PCA whitened adversarial images are often visibly distinct from whitened source images, however we show that adversarial images can still fool a target model defended by PCA whitening.

Feature Squeezing [39] is a model hardening technique, attempting to convert adversarial images to images that will be correctly classified. Feature Squeezing reduces the complexity of an input image such that adversarial noise is removed, or rendered ineffective. This is performed in two steps: (1) by reducing the color depth of each pixel in the image, and (2) smoothing the image with a median filter. The authors report that using this method to defend against adversarial images created using the JSMA attack [28] on MNIST, resulted in a reduction in accuracy of only 1.5% - from 99.1% accuracy on source images to 97.6% accuracy on defended adversarial images.

We measure the success of each defense under two scenarios: (1) where the type of defense is known to the attacker, and (2) where the defense is not known. While we anticipate that a service providing black-box access to a prediction model would not advertise a defense if one was in use, it is important to understand the advantage an attacker may have if this information were to leak. In both (1) and (2) we allow a pre-processing step, performed prior to input into the target model but not visible to the attacker, that transforms the image to either a PCA whitened image or a 'Feature Squeezed' image. In (1) we assume the attacker knows the method of defense, and so allow this pre-processing step to also be performed by the attacker before the source image is fed into the attacker model, while in (2) we assume no such knowledge and so no pre- or post-processing step is performed by the attacker.

Table 5 shows results for both scenarios. Firstly the target model performs well on defended source images: PCA whitening reduces test set accuracy on source images by 1.6% (from 99.4% to 97.8%), while Feature Squeezing reduces test set accuracy on source images by 0.4% (from 99.4% to 99.0%). When the defense is known, it is rendered ineffective, with the target model scoring similarly to when there is no defense in place, while the defenses do well when the defense is unknown. However, we note that the target model

TABLE 5: Attack on two popular adversarial example defenses, PCA Whitening (*PCA-W*) and Feature Squeezing (*FS*), on the MNIST dataset. We report the target model accuracies on source images and adversarial images on the MNIST test set.

Image Type		**Target Model Accuracy (%)**				
		Defense				
		No Defense	*PCA-W*		*FS*	
Source		99.4	97.8		99.0	
	Attack Weight (α)		**Defense Unknown**	**Defense Known**	**Defense Unknown**	**Defense Known**
Adversarial	1	0.77	76.9	18.7	81.2	8.1
	10	19.1	83.9	22.4	87.0	11.6
	20	26.8	89.1	26.9	91.7	20.0
	40	45.3	92.8	29.9	93.4	30.4
	80	59.8	96.4	44.8	98.6	40.1

still suffers in accuracy for low attack weights, adversarial examples reduce accuracy by approximately 20% for both defenses.

8. Attack Transferability

An adversarial image is *transferable* if it successfully fools a model that was not its original target. This could be another deep neural network with a differing architecture or a different model entirely. Transferability is a yardstick of the robustness of adversarial examples, and is the main property used by Papernot *et al.* [26], [27] to construct black-box adversarial examples. They construct a white-box attack on a local target model that has been trained to replicate the intended target models decision boundaries, and show that the adversarial examples can successfully transfer to fool the black-box target model.

We conduct experiments into transferability under two settings; *direct* and *non-direct*. Direct transferability refers to the traditional setting: the target model and attacker model is as described in Section 3.2, we then create 10,000 adversarial images - one for each image in the MNIST test set - and apply them to a *different* target model. In non-direct transferability, we refer to a target model that is different from reported in Section 3.2, however we allow the attacker model to query the target model during training. Table 6 gives results for transferability experiments on three target models - a Random Forest (RF), Support Vector Machine (SVM), and K-Nearest Neighbour (KNN) [9]. The target model accuracies on source images for RF, SVM and KNN are 95.2%, 91.8% and 95.4%, respectively. We find that adversarial samples crafted using our attack do transfer to other models. For a direct attack with an attack weight of ten, adversarial examples reduce RF accuracy from 95.2% to 69.2%, SVM accuracy from 91.8% to 71.1%, and KNN accuracy from 95.4% to 85.9%. While a non-direct attack with an attack weight of ten reduces RF accuracy from 95.2% to 29.1%, SVM accuracy from 91.8% to 47.5%, and KNN accuracy from 95.4% to 71.7%.

KNN faired best at resisting adversarial examples while RF suffered the most. Non-direct transfered samples were more successful than direct adversarial examples at fooling the target model, as expected, since we allow the attacker model to query the target model during training. Finally, we note that it has been hypothesized [26] that the respective target models are somewhat robust to adversarial examples due to: the constraint on decision boundaries in the hyperplane during training (SVM), non-differentiability of the model (RF, KNN), and that no model is learned during training (KNN).

9. Amazon Attack and Comparison with Other Attacks

We now evaluate our attack on a real-world black-box prediction model - the *Amazon Machine Learning prediction API* - and against another popular attack on black-box models - the Cleverhans [25] implementation of the FGSM black-box attack [27].

9.1. Amazon Attack.

Amazon Machine Learning [10] provides customers a tool that trains a black-box model on a dataset (of the customers choice) and then allows the model to be queried for predictions publicly. To evaluate our attack on *Amazon Machine Learning*, we uploaded the MNIST dataset to an S3 bucket [11], and trained a model using the default settings. The model took 17 minutes to train and reported 91.8% accuracy on the test set.

We trained our attacker model on the class confidence scores output by the Amazon target model. In total, we made 280,000 queries to the target model; at a cost of $0.0001 per query, the total monetary cost of the attack was $28. After training the attacker model with an attack weight of 20, we created 10,000 adversarial images from the test set, and queried the target model with each adversarial image. The Amazon target model reported an accuracy of 61.3% on the adversarial test set - a reduction of 30%. In other words, the created adversarial images only succeeded in fooling the target model with probability 38.7%. In comparison, the same attack set-up on a local black-box target model was 73.2% successful (see Figure 3). However, Amazon uses

[9] We use the `scikit-learn` Python library to implement RF, SVM and KNN. The parameters of each model are as follows: RF is made up of ten trees, SVM uses the RBF kernel with $C = 5$ and $\gamma = 0.05$, and KNN uses five neighbors for classification.

[10] https://aws.amazon.com/machine-learning/

[11] https://aws.amazon.com/s3/

TABLE 6: Transferability experiments on MNIST.

Model	**Accuracy of Target Model (%)**											
	Source Images	**Adversarial Images**										
			Direct Transfer					**Non-direct Transfer**				
		Attack Weight (α):	1	10	20	40	80	1	10	20	40	80
RF	95.2		38.5	69.2	72.1	73.6	80.3	10.2	29.1	48.2	66.0	77.1
SVM	91.8		44.7	71.1	78.0	76.6	82.5	23.1	47.5	57.7	71.8	81.5
KNN	95.4		79.0	85.9	88.3	90.2	92.9	49.9	71.7	81.4	80.9	83.5

TABLE 7: Comparison with other attacks on MNIST.

Attack	**Successful Adversarial Examples (%)**	L_∞ **(avg.)**	**Time (s)**
BB-FGSM	63.6	0.3	$101,348$
BB-CL2	59.9	0.51	$113,476$
ZOO-Adam (100 iter.)	70.7	0.21	$20,057$
ZOO-Adam (500 iter.)	100	0.26	$72,677$
JH ($\alpha = 1$)	99.3	2.39	296
JH ($\alpha = 5$)	91.7	1.69	305
JH ($\alpha = 25$)	71.7	1.23	288

multinomial logistic regression as a prediction model - a non-differentiable model - and our results are in line with results from transferability experiments on other non-differentiable models.

9.2. Comparison with Other Attacks.

We compare our attack, referred to as JH, against two state-of-the-art black-box attacks. A full description of each attack is given in Section 12:

1) the Cleverhans [25] black-box attack implementation using (1) Fast Gradient Sign Method [8] (BB-FGSM) and (2) Carlini & Wagner's L_2 based attack [4] (BB-CL2). In this attack, Papernot *et al.* [27] trained a substitute model that approximates the black-box target model. Adversarial images are generated from the substitute model and transferred to the target model.
2) Chen *et al.'s* [5] black-box attack (ZOO-Adam). This attack is based on Carlini & Wagner's L_2 based attack but uses gradient free optimization methods to learn to craft adversarial examples. We experimented with two attack settings of 100 and 500 iterations. An iteration corresponds to updating the adversarial image after computing the loss. More iterations equates to a longer period for loss convergence and a greater chance of finding an adversarial image.

Table 1 shows attack comparison results. Attacks are trained on the MNIST training set (when applicable), and results reported on the full MNIST test set, we also report the total time that the attack took to run. JH, BB-FGSM and BB-CL2 attacks were trained for ten epochs [12], our attack finished training in approximately 300 seconds. BB-FGSM and BB-CL2 took approximately 375× longer than our attack, while ZOO-Adam took nearly 240× longer, due to the timely optimization process that is performed for *each* image. In terms of misclassification success, our attack is substantially stronger than BB-FGSM, BB-CL2, and ZOO-Adam (100 iter.), and is comparable to ZOO-Adam (500 iter.). Our attack has marginally larger average L_∞ distances from source images, but is comparable to reconstruction error in state-of-the-art autoencoders. Not only was our attack faster to train, but can be optimized to increase attack success or perturbation levels, depending on the context of attack. Furthermore, altering the attack weight does not impact attacker model training time.

10. Attacks are Identifiable

There are a variety of methods for crafting adversarial examples, from detecting and manipulating pixels that contribute the most to classification [4], [23] , to perturbing all pixels by a small amount [4], [8]. It is no surprise then, that a classifier can recognize and differentiate between these attacks. Figure 10 shows a two dimensional visualization of six popular attacks, including our own, referred to as JH (see Section 12 for a description of other attacks). From the MNIST test set, we created 5,000 successful adversarial images for each of the six attacks - using the target model described in Table 2 -and visualize via the dimensionality reduction technique, *t-SNE* [18]. There is clearly structure among each of the attacks which can be used as a detection mechanism. Interestingly, the FGSM adversarial images are particularly identifiable, creating ten distinct clusters - one for each digit, while CL0 and CLi are the least identifiable. However, one should not use this visualization as a benchmark of adversarial image detectability; the differences in visual detectability could be due to inherent weakness in the dimensionality reduction method.

To establish if adversarial images created under different attacks are separable, we train a classifier that aims to recognize one attack from another. We use the same model as described in Table 2, but now compress the Log Softmax output to a six dimensional vector instead of ten - one for each type of attack. The model was trained on 2,500 samples from each of the six attacks (15,000 training images) and tested on the remaining 2,500 from each attack (15,000 test images). Figure 11 shows the confusion matrix of the model; each attack is easy to separate from one another. As suggested by Figure 10, the FGSM attack is easily identifiable, with 99% of FGSM test images being correctly classified, and the CL0 is the most difficult to identify, with a 76% test set accuracy. Interestingly, when an attack is confused with

[12] All experiments used the same computational set-up: an NVIDIA TITAN X Graphics Card and 32GB RAM.

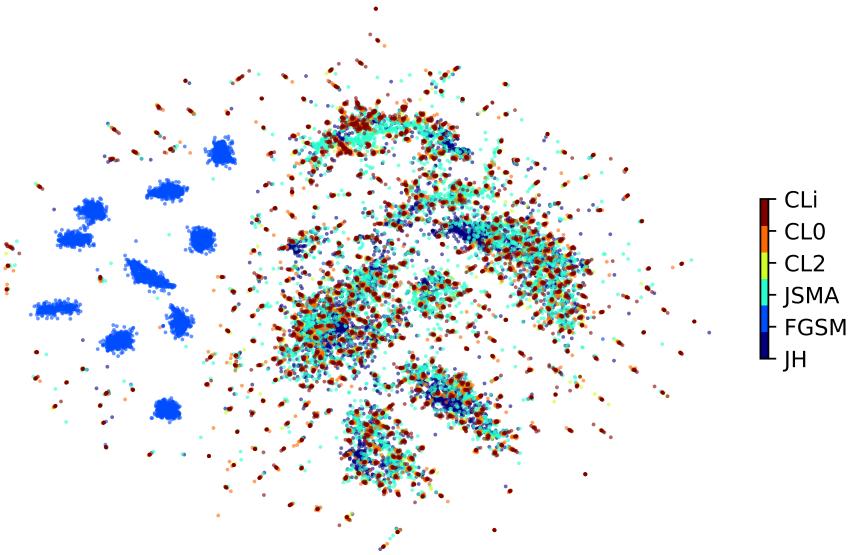
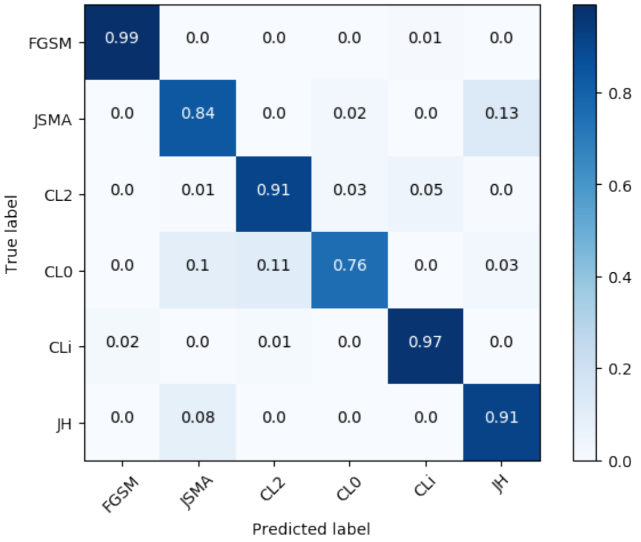


Figure 10: Visualization of different attacks using t-SNE.

another, it is predominantly confused with just one other attack. For example, the CL0 attack is confused with CL2 attack in 11% of test images, while the JSMA attack (which has an accuracy of 84%) is confused with the JH attack in 13% of test images. However the converse does not always hold, for example the CL2 attack is not confused with the CL0 attack at the same rate.

Clearly attacks are identifiable and separable from one another, a natural question then is, can this insight be used by a target model to detect and defend against attacks? We answer this by extending the previous experiment to source images from the MNIST dataset. The task then is to recognize attacks from one another and from MNIST digits. Again, the model used is as described in Table 2, but now we compress the Log Softmax output to a 16 dimensional vector - one for each type of attack and each digit. The model was trained on 10% of the MNIST dataset (7,000 images) and 10% of the adversarial images (3,000 images), with results reported on the remaining images. The overall test set accuracy is 92.7%, and more importantly adversarial images are almost never identified as a source image; either being assigned the correct attack label or, on rare occasions, is confused with another attack. This method can therefore be used an effective method for detecting adversarial images.



True label	FGSM	JSMA	CL2	CL0	CLi	JH
FGSM	0.99	0.0	0.0	0.0	0.01	0.0
JSMA	0.0	0.84	0.0	0.02	0.0	0.13
CL2	0.0	0.01	0.91	0.03	0.05	0.0
CL0	0.0	0.1	0.11	0.76	0.0	0.03
CLi	0.02	0.0	0.01	0.0	0.97	0.0
JH	0.0	0.08	0.0	0.0	0.0	0.91

Figure 11: Confusion matrix of a classifier that tries to recognize different attacks. The classifier is trained on only 10% of the MNIST dataset.

11. Discussion

11.1. Attacker Model.

Deciding which neural network architecture to use given a problem is more of a dark art than a science. We experimented with hundreds of architectures and hyperparameters before settling on the attacker model reported in Table 1. We selected a convolutional autoencoder over a fully-connected neural network due to its superior ability to learn an accurate reconstruction of the source image, while maintaining perturbations that cause an intentional misclassification in the target model. We found that optimizing the attacker model loss under the L_2 distance metric produced better results than optimizing for the absolute difference or maximum perturbation, and produced similar results when using the

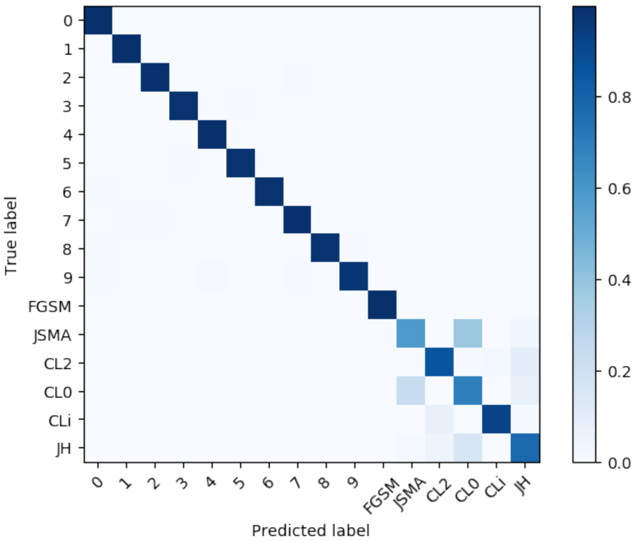


Figure 12: Confusion matrix of a classifier that tries to recognize different attacks and MNIST digits. The classifier is trained on only 10% of the MNIST dataset.

mean binary cross-entropy.

11.2. Datasets.

Machine learning research has started to move away from datasets such as MNIST, favoring more complex, larger datasets such as ImageNet on which to benchmark research. This should be applauded, as applying research to more complex datasets provides a clearer intuition of generalizability and impact. However, we claim the converse should be a standard for benchmarking adversarial example attacks - adversarial examples should be primarily tested on datasets like MNIST and CIFAR-10. An attack that has shown to be effective against a dataset that normally requires more distortion to misclassify an input, and that has an associated target model with near perfect test set accuracy, is more useful to the research community than an attack that has been tested on a large, complex dataset with a weak target model.

11.3. How Much Data Does the Attack Need?

In Section 5, we demonstrated that an attacker with access to 600 or 60,000 data samples can achieve approximately the same attack success rate. The attacker model is able to learn and generalize while training on few data samples. It would be an interesting direction for future work to quantify the amount of dataset knowledge an attacker needs to launch an attack. We did not seek to answer questions such as this in this work, however, an attacker would clearly need knowledge of at least one image from each class, and realistically, more than one, in order to learn some information about the structure of each class. We believe it is important to evaluate an attack in all possible circumstances. That is, in both *best* and *worst case scenarios*, where an attacker has knowledge of *very few* or *all* data samples used to train the target model (*cf.* Figure 7).

11.4. A Target Model Oblivious Attack.

Our attack queries the target model directly, from which the attacker model is trained. As such, it is agnostic to the type of model under attack. We have shown that we can attack a neural network, RF, SVM, and KNN, all with high adversarial example success. Furthermore, we show we do not have to query the target model directly for adversarial examples to work: for small attack weights, adversarial examples crafted by an attacker model that was originally trained against a target neural network, transfer and fool a RF target model. For example, an attack weight of ten reduced the RF test accuracy on MNIST from 95.2% to 69.2%, and because Figure 2 showed that adversarial examples crafted with an attack weight of ten are visually similar to the source samples, we claim that our attack can craft adversarial examples that transfer between models.

11.5. Insights into Adversarial Examples.

Our work offers important insights into how attacks are realized. In Section 6 we showed that a target model is inhibited by noise in its goal to separate classes, and that an attack can exploit this inherent weakness to craft adversarial examples that perturb fewer pixels in the foreground image. Perturbing fewer pixels in the foreground of an image such as a digit in MNIST is advantageous for an attack, since this is what distinguishes classes. Thus an attack that perturbs background noise is more likely to be stealthy and visually imperceptible from the source image. We also highlight the distinguishability among different types of attacks. Previous work [7], [9], [20] concentrated on a binary classification of adversarial and non-adversarial images. While we do not claim any significant advantage over this approach for defending against adversarial examples, we consider the insight that different attacks have unique distributions to be both interesting and potentially useful for the construction of more advanced detection methods in future work.

11.6. An Adaptive Attack.

The attack weight, a unique property of our attack, allows an attacker to tune their model depending on the setting. For example, when attacking the Feature Squeezing defense, we were able to decrease test set accuracy on the adversarial examples as we decreased the attack weight, at the expense of more visually dissimilar examples. The capacity to tailor an attack based on properties that an attacker would like to achieve (for example, transferability or an attack on a proposed defense) is useful and important in practical settings. Ultimately, the attack weight trades overall success of fooling the target model for visual similarity between the source and adversarial image, and we leave an evaluation of attack weight choice strategies for future work.

12. Related Work

We now survey the landscape of attacks and defenses in adversarial example research.

12.1. Attacks

In a departure from other attacks that assume white-box access to the target model [1], [4], [8], [15], [22], [28] or construct their black-box attack through a transfer of a white-box attack [27], our attack is optimized directly by information output by the black-box target model. Here we give an overview of a number of aforementioned state-of-the-art white-box and black-box attacks:

- The *Fast Gradient Sign Method* (FGSM) [8] is an attack, optimizing the L_∞ distance metric, that is designed to compute adversarial perturbations efficiently. Given an source image, x, FGSM iteratively updates x to produce an adversarial image by setting:

  $$x^{adv} = x + \epsilon \cdot \text{sign}(\nabla_x J(\theta, x, y)),$$

  where θ are the model parameters, y is the target of the model, $J(\theta, x, y))$ is the objective function of the model, and ϵ is the step-size of the perturbation. FGSM computes the direction the pixels of the image need to move towards to cause a misclassification, and then shifts them all by a factor of ϵ. This provides a coarse approximation of the optimal solution, resulting in visually noisy adversarial images; Kurakin *et al.* [15] show an improvement on FGSM by taking smaller step sizes.
- The *Jacobian-based Saliency Map* (JSMA) [28] is optimized under the L_0 distance metric, searching for pixels that have the most impact on classification, and perturbing them, resulting in a prediction with higher chosen target class confidence. More specifically, given a neural network F, a clean image x, and a target class t, to achieve the intended misclassification, $F_t(x)$ must be increased while prediction probabilities of $F_i(x)$ for classes $i \neq t$ must decrease until $t = \arg\max_i F_i(x)$. This is accomplished using the saliency map:

  $$S(x,t)[i] = \begin{cases} 0 \text{ if } \frac{\partial F_t(x)}{\partial x_j} < 0 \text{ or } \sum_{i \neq t} \frac{\partial F_i(x)}{\partial x_j} > 0 \\ \left(\frac{\partial F_t(x)}{\partial x_j}\right) \cdot \left|\sum_{i \neq t} \frac{\partial F_i(x)}{\partial x_j}\right| \text{ otherwise} \end{cases},$$

  where i is an input feature. JSMA finds features i and j in an input image x such that the pair of features maximizes the summation of their respective saliency maps. Each feature is then perturbed by ϵ. The process repeats until a threshold number of pixels have been modified, or $F(x) = t$.
- Carlini and Wagner [4] recently proposed three attacks under an L_0, L_2 and L_∞ distance metric (referred to as CL0, CL2 and CLi, respectively). All three attacks generate high quality adversarial images, and have shown to be successful in attacking a number of recently proposed defenses [3]. We encourage interested readers to refer to the original paper for a full technical explanation of the attacks. In short, the CL2 attack uses gradient descent to solve

  $$\text{minimize } ||\tfrac{1}{2}(\tanh(w) + 1) - x||_2^2 + c \cdot l(\tfrac{1}{2}(\tanh(w) + 1))||$$

  where l is the loss function defined as

  $$l(x') = \max(\max Z(x')_i : i \neq t - Z(x')_t, -\kappa)$$

  Z is the logits of the target neural network, t is the attacker chosen target class, w is a change of variable such that $\delta = \frac{1}{2}(\tanh(w) + 1) - x$, where δ is the adversarial perturbation applied to the image, c is a weight coefficient, and κ controls the confidence of misclassification.
- Moosavi-Dezfooli *et al.* [22] developed an attack that crafts adversarial examples by approximating a non-linear model as a linear model, and finding the closest decision boundary - according to an L_p distance metric. The iterative attack updates an image by repeatedly finding the closest boundary and taking steps in that direction. The attack terminates once the image fools the non-linear model. Moosavi-Dezfooli *et al.* [21] also show that there is a *Universal Adversarial Perturbation* that can cause a misclassification of an image from any dataset with high probability. However, this method is computationally expensive and requires access to the target model's training data, somewhat limiting its practicality.
- Papernot *et al.* [27] develop a black-box attack by exploiting the transferability property of adversarial examples. By constructing a *substitute* model on similar data samples, crafting an adversarial image through a white-box attack, and transferring this to the target model they show practical black-box attacks are feasible.
- Papernot *et al.* [26] have recently shown that the susceptibility to adversarial examples is not model specific, it is possible to construct adversarial inputs on one model and t ransfer them to another model while still preserving intentional misclassifications. Thus, an attacker with only limited information can still launch powerful attacks.
- In concurrent work, Baluja & Fischer [1] develop a machine learning approach to crafting white-box adversarial examples. They train a neural network to produce adversarial examples for a target model, using the target model's gradient as a reward signal. While Chen *et al.* [5], have recently developed a black-box attack that relies on gradient free optimization methods. The attack is based on Carlini and Wagner's [4] CL2 attack, but optimizes under a loss function using F, the output of the target model, instead of the logits Z:

  $$l(x') = \max(\max F(x')_i : i \neq t - F(x')_t, -\kappa)$$

12.2. Defenses

An arms race has developed between offensive and defensive adversarial research. Attacks have succeeded in developing high quality images that fool target models. However, progress has also continued on the defensive side, we give an overview of some of the most promising defenses and detection methods:

- Gong *at al.* [7] and Metzen *et al.* [20] both train an additional neural network on top of the target model, that makes a binary prediction of if an input was adversarial or not. Similarly, Grosse *et al.* [9] adds an additional "adversarial class" to a target model to detect adversarial examples. All works report high detection rates on datasets such as MNIST, CIFAR-10 and ImageNet.
- Feinman *et al.* [6] devise a detection method by showing adversarial examples deviate from the true data manifold. The method uses a density estimate between a sample and a set of known non-adversarial samples and shows that adversarial examples are drawn from a different distribution.
- Hendrycks & Gimpel [10] perform PCA on an image, arguing adversarial examples have PCA coefficients with greater variance, which is used as a detection method.
- Xu *et al.* [39] claim reducing color depth and smoothing an image with a filter effectively scrubs adversarial perturbations from an adversarial image while maintaining high classification accuracy among non-adversarial images.

13. Conclusion

We presented a first-of-its-kind adversarial example attack on black-box models that uses machine learning at the heart of its construction. We comprehensively evaluated the attack under many different settings, showing that it produces quality adversarial examples capable of fooling a target model in both targeted and untargeted attacks. The attack is adaptive and transfers to many different target models, reduces the effectiveness of adversarial example defenses, and improves on other state-of-the-art black-box attacks.

References

[1] S. Baluja and I. Fischer. Adversarial transformation networks: Learning to generate adversarial examples. *arXiv preprint arXiv:1703.09387*, 2017.

[2] A. L. Buczak and E. Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2):1153–1176, 2016.

[3] N. Carlini and D. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. *arXiv preprint arXiv:1705.07263*, 2017.

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

[5] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh. ZOO: Zeroth Order Optimization based Black-box Attacks to Deep Neural Networks without Training Substitute Models. *ArXiv e-prints*, Aug. 2017.

[6] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

[7] Z. Gong, W. Wang, and W.-S. Ku. Adversarial and clean data are not twins. *arXiv preprint arXiv:1704.04960*, 2017.

[8] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[9] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.

[10] D. Hendrycks and K. Gimpel. Early methods for detecting adversarial images. 2017.

[11] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

[12] S.-J. Kim and S. Boyd. A minimax theorem with applications to machine learning, signal processing, and finance. *SIAM Journal on Optimization*, 19(3):1344–1367, 2008.

[13] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[15] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[16] T. D. Lane. Machine learning techniques for the computer security domain of anomaly detection. 2000.

[17] W.-Y. Lin, Y.-H. Hu, and C.-F. Tsai. Machine learning in financial crisis prediction: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):421–436, 2012.

[18] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[19] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. *Artificial Neural Networks and Machine Learning–ICANN 2011*, pages 52–59, 2011.

[20] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff. On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*, 2017.

[21] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. *arXiv preprint arXiv:1610.08401*, 2016.

[22] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.

[23] N. Narodytska and S. P. Kasiviswanathan. Simple black-box adversarial perturbations for deep networks. *arXiv preprint arXiv:1612.06299*, 2016.

[24] Z. Obermeyer and E. J. Emanuel. Predicting the futurebig data, machine learning, and clinical medicine. *The New England journal of medicine*, 375(13):1216, 2016.

[25] N. Papernot, I. Goodfellow, R. Sheatsley, R. Feinman, and P. McDaniel. cleverhans v1.0.0: an adversarial machine learning library. *arXiv preprint arXiv:1610.00768*, 2016.

[26] N. Papernot, P. McDaniel, and I. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[27] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against deep learning systems using adversarial examples. *arXiv preprint arXiv:1602.02697*, 2016.

[28] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 372–387. IEEE, 2016.

[29] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. *Computer Vision–ECCV 2006*, pages 430–443, 2006.

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

[31] M. A. Shipp, K. N. Ross, P. Tamayo, A. P. Weng, J. L. Kutok, R. C. Aguiar, M. Gaasenbeek, M. Angelo, M. Reich, G. S. Pinkus, et al. Diffuse large b-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning. *Nature medicine*, 8(1):68–74, 2002.

[32] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[33] S. Sivaraman and M. M. Trivedi. Active learning for on-road vehicle detection: A comparative study. *Machine vision and applications*, pages 1–13, 2014.

[34] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[35] T. B. Trafalis and H. Ince. Support vector machine for regression and applications to financial forecasting. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, volume 6, pages 348–353. IEEE, 2000.

[36] F. Tramèr, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*, 2017.

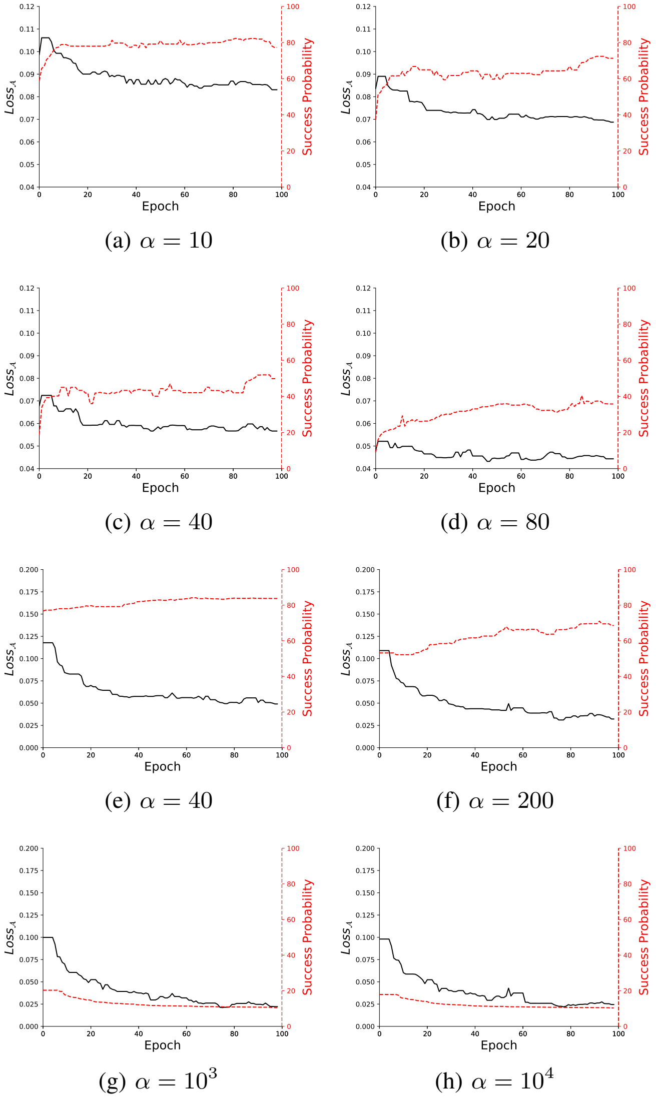
[37] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart. Stealing machine learning models via prediction apis. In *USENIX Security Symposium*, pages 601–618, 2016.

[38] X. Wen, L. Shao, Y. Xue, and W. Fang. A rapid learning algorithm for vehicle classification. *Information Sciences*, 295:395–406, 2015.

[39] W. Xu, D. Evans, and Y. Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

[40] Q.-H. Ye, L.-X. Qin, M. Forgues, P. He, J. W. Kim, A. C. Peng, R. Simon, Y. Li, A. I. Robles, Y. Chen, et al. Predicting hepatitis b virus-positive metastatic hepatocellular carcinomas using gene expression profiling and supervised machine learning. *Nature medicine*, 9(4):416, 2003.

Appendix A.
Attacker Model Loss



(a) $\alpha = 10$

(b) $\alpha = 20$

(c) $\alpha = 40$

(d) $\alpha = 80$

(e) $\alpha = 40$

(f) $\alpha = 200$

(g) $\alpha = 10^3$

(h) $\alpha = 10^4$

Figure 13: Success probability of adversarial images and $Loss_{\mathcal{A}}$ loss as the attacker model is trained for various attack weights. MNIST results are given in (a), (b), (c), (d), CIFAR-10 results are given in (e), (f), (g), (h).