Supplementary material: Practical Black-box Attacks on Deep Neural Networks using Efficient Query Mechanisms

Arjun Nitin Bhagoji[1] [⋆], Warren He[2], Bo Li[3], and Dawn Song[2]

[1] Princeton University
[2] University of California, Berkeley
[3] University of Illinois at Urbana–Champaign

We provide details of white-box attacks in Section 1, further descriptions of Gradient Estimation attacks in Section 2, the detailed evaluation setup in Section 3, descriptions and results for zero-query black-box attacks in Section 4, details about query reduction based attacks in Section 5.1, more details and results for adversarially trained models in Section 6, evaluation results for countermeasures in Section 7, comparison with previous work in Section 8 and more of the images used to attack Clarifai as well as the method followed in choosing them in Section 9.

1 White-box attacks

In this section, we describe two commonly used white-box attack methods. These attacks are based on either iterative or single-step gradient based minimization of appropriately defined loss functions of neural networks. The single-step Fast Gradient method, first introduced by [8], utilizes a first-order approximation of the loss function in order to construct adversarial examples for the model f. The samples are constructed by performing a single step of gradient ascent for untargeted attacks. Formally, the adversary generates samples $\mathbf{x}_{\text{adv}}$ with L_∞ constraints (known as the Fast Gradient Sign (FGS) method) in the *untargeted* attack setting as

$$\mathbf{x}_{\text{adv}} = \mathbf{x} + \epsilon \cdot \text{sign}(\nabla_{\mathbf{x}} \ell_f(\mathbf{x}, y)), \tag{1}$$

where $\ell_f(\mathbf{x}, y)$ is the loss function with respect to which the gradient is taken. Any loss function can be used and two commonly used ones are the cross-entropy loss [7] and the logit loss [3]. Adversarial samples generated using the *targeted* FGS attack are

$$\mathbf{x}_{\text{adv}} = \mathbf{x} - \epsilon \cdot \text{sign}(\nabla_{\mathbf{x}} \ell_f(\mathbf{x}, T)), \tag{2}$$

where T is the target class.

Iterative Fast Gradient methods are simply multi-step variants of the Fast Gradient method described above [12], where the gradient of the loss is added to

[⋆] Work done while at University of California, Berkeley the sample for $t+1$ iterations, starting from the benign sample, and the updated sample is projected to satisfy the constraints $\mathcal{H}$ in every step:

$$\mathbf{x}_{\text{adv}}^{t+1} = \Pi_{\mathcal{H}}(\mathbf{x}_{\text{adv}}^{t} + \alpha \cdot \text{sign}(\nabla_{\mathbf{x}_{\text{adv}}^{t}} \ell_f(\mathbf{x}_{\text{adv}}^{t}, y))), \tag{3}$$

with $\mathbf{x}_{\text{adv}}^{0} = \mathbf{x}$. Iterative fast gradient methods thus essentially carry out projected gradient descent (PGD) with the goal of maximizing the loss, as pointed out by [16]. *Targeted* adversarial examples generated using iterative FGS are

$$\mathbf{x}_{\text{adv}}^{t+1} = \Pi_{\mathcal{H}}(\mathbf{x}_{\text{adv}}^{t} - \alpha \cdot \text{sign}(\nabla_{\mathbf{x}_{\text{adv}}^{t}} \ell_f(\mathbf{x}_{\text{adv}}^{t}, T))). \tag{4}$$

1.1 Beyond the cross-entropy loss

Prior work by [3] investigates a variety of loss functions for white-box attacks based on the minimization of an appropriately defined loss function. In our experiments with neural networks, for untargeted attacks, we use a loss function based on logits which was found to work well for white-box attacks in [3]. The loss function is given by

$$\ell(\mathbf{x}, y) = \max(\boldsymbol{\phi}(\mathbf{x} + \boldsymbol{\delta})_y - \max\{\boldsymbol{\phi}(\mathbf{x} + \boldsymbol{\delta})_i : i \neq y\}, -\kappa), \tag{5}$$

where y represents the ground truth label for the benign sample $\mathbf{x}$, $\boldsymbol{\phi}(\cdot)$ are the logits. κ is a confidence parameter that can be adjusted to control the strength of the adversarial sample. *Targeted* adversarial examples are generated using the following loss term:

$$\mathbf{x}_{\text{adv}} = \mathbf{x} - \epsilon \cdot \text{sign}(\nabla_{\mathbf{x}}(\max(\boldsymbol{\phi}(\mathbf{x})_i : i \neq T) - \boldsymbol{\phi}(\mathbf{x})_T)). \tag{6}$$

2 Query-based attacks: analysis, expressions and images

In this section, we provide the expressions for attacks using the method of Finite Differences with the cross-entropy loss. We also provide the expressions for targeted attacks using the method of finite differences. Details about Particle Swarm Optimization can be found in Kennedy [10] and details about Simultaneous Perturbation Stochastic Approximation can be found in Spall [19].

2.1 Approximate FGS with finite differences

In the untargeted FGS method, the gradient is usually taken with respect to the cross-entropy loss between the true label of the input and the softmax probability vector. The cross-entropy loss of a network f at an input $\mathbf{x}$ is then $\ell_f(\mathbf{x}, y) = -\sum_{j=1}^{|\mathcal{Y}|} \mathbf{1}[j = y] \log p_j^f(\mathbf{x}) = -\log p_y^f(\mathbf{x})$, where y is the index of the original class of the input. The gradient of $\ell_f(\mathbf{x}, y)$ is

$$\nabla_{\mathbf{x}} \ell_f(\mathbf{x}, y) = -\frac{\nabla_{\mathbf{x}} p_y^f(\mathbf{x})}{p_y^f(\mathbf{x})}. \tag{7}$$

An adversary with query access to the softmax probabilities then just has to estimate the gradient of $p^f_y(\mathbf{x})$ and plug it into Eq. 7 to get the estimated gradient of the loss. The adversarial sample thus generated is

$$\mathbf{x}_{\text{adv}} = \mathbf{x} + \epsilon \cdot \text{sign}\left(\frac{\text{FD}_{\mathbf{x}}(p^f_y(\mathbf{x}), \delta)}{p^f_y(\mathbf{x})}\right). \tag{8}$$

This method of generating adversarial samples is denoted as FD-xent. *Targeted* black-box adversarial samples generated using the Gradient Estimation method are then

$$\mathbf{x}_{\text{adv}} = \mathbf{x} - \epsilon \cdot \text{sign}\left(\frac{\text{FD}_{\mathbf{x}}(p^f_T(\mathbf{x}), \delta)}{p^f_T(\mathbf{x})}\right). \tag{9}$$

The targeted version of the method is denoted as FD-xent-T.

2.2 Estimating the logit-based loss

We also use the loss function based on logits described in Section 1.1. If the confidence parameter κ is set to 0, the logit loss is $\max(\boldsymbol{\phi}(\mathbf{x}+\boldsymbol{\delta})_y - \max\{\boldsymbol{\phi}(\mathbf{x}+\boldsymbol{\delta})_i : i \neq y\}, 0)$. For an input that is correctly classified, the first term is always greater than 0, and for an incorrectly classified input, an untargeted attack is not meaningful to carry out. Thus, the loss term reduces to $\boldsymbol{\phi}(\mathbf{x}+\boldsymbol{\delta})_y - \max\{\boldsymbol{\phi}(\mathbf{x}+\boldsymbol{\delta})_i : i \neq y\}$ for relevant inputs.

An adversary can compute the logit values up to an additive constant by taking the logarithm of the softmax probabilities, which are assumed to be available in this threat model. Since the loss function is equal to the difference of logits, the additive constant is canceled out. Then, the finite differences method can be used to estimate the difference between the logit values for the original class y, and the second most likely class y', i.e., the one given by $y' = \text{argmax}_{i \neq y}\, \boldsymbol{\phi}(\mathbf{x})_i$. The untargeted adversarial sample generated for this loss in the white-box case is $\mathbf{x}_{\text{adv}} = \mathbf{x} + \epsilon \cdot \text{sign}(\nabla_{\mathbf{x}}(\boldsymbol{\phi}(\mathbf{x})_{y'} - \boldsymbol{\phi}(\mathbf{x})_y))$. Similarly, in the case of a black-box adversary with query-access to the softmax probabilities, the adversarial sample is

$$\mathbf{x}_{\text{adv}} = \mathbf{x} + \epsilon \cdot \text{sign}(\text{FD}_{\mathbf{x}}(\boldsymbol{\phi}(\mathbf{x})_{y'} - \boldsymbol{\phi}(\mathbf{x})_y, \delta)). \tag{10}$$

Similarly, a *targeted* adversarial sample is

$$\mathbf{x}_{\text{adv}} = \mathbf{x} - \epsilon \cdot \text{sign}(\text{FD}_{\mathbf{x}}(\max(\boldsymbol{\phi}(\mathbf{x})_i : i \neq T) - \boldsymbol{\phi}(\mathbf{x})_T, \delta)). \tag{11}$$

The untargeted attack method is denoted as FD-logit while the targeted version is denoted as FD-logit-T.

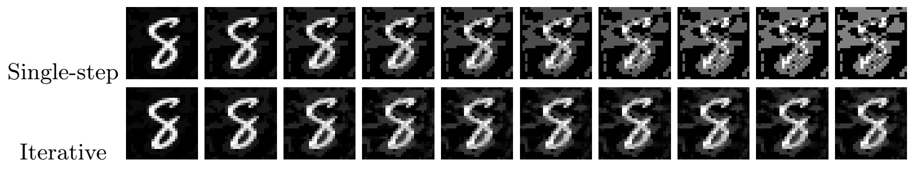
2.3 Effect of δ on attack success

We find that for the logit loss, a large range of values of δ are effective. For example, on the MNIST dataset, δ ranging from 1×10^{-2} to 1×10^{-4} achieves roughly the same success rate. Attack success rate decreases for lower values of δ as the differences in logit values become small, leading to poor estimation. On the other hand, the `xent` loss is more sensitive to the choice of δ and performs best for large values of δ. In fact, it is most effective when set to 1.0 for the MNIST dataset. Similar trends hold for the CIFAR-10 data.

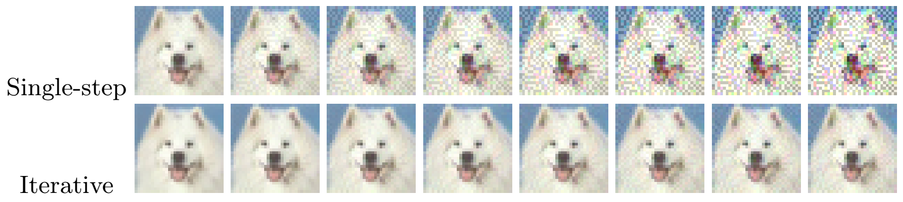
2.4 Visualization of different levels of image distortion:

In Table 1 below, we qualitatively show adversarial samples with different distortion levels. We will add these samples to our updated version as well.

Table 1. Visualization of targeted black-box adversarial examples.



(a) MNIST: Perturbation level ϵ varies from 0.05 to 0.5 in increments of 0.05.



(b) CIFAR-10 Perturbation level ϵ varies from 4 to 32 in increments of 4.

3 Detailed evaluation setup

The empirical evaluation carried out in the paper and the supplementary material is on state-of-the-art neural networks on the MNIST [13] and CIFAR-10 [11] datasets. The details of the datasets and the architecture and training procedure for all models are given below.

3.1 Datasets

MNIST. This is a dataset of images of handwritten digits [13]. There are 60,000 training examples and 10,000 test examples. Each image belongs to a single class from 0 to 9. The images have a dimension d of 28×28 pixels (total of 784) and are grayscale. Each pixel value lies in $[0, 1]$. The digits are size-normalized and centered. This dataset is used commonly as a 'sanity-check' or first-level benchmark for state-of-the-art classifiers. We use this dataset since it has been extensively studied from the attack perspective by previous work.

CIFAR-10. This is a dataset of color images from 10 classes [11]. The images belong to 10 mutually exclusive classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck). There are 50,000 training examples and 10,000 test examples. There are exactly 6,000 examples in each class. The images have a dimension of 32×32 pixels (total of 1024) and have 3 channels (Red, Green, and Blue). Each pixel value lies in $[0, 255]$.

3.2 Model training details

In this section, we present the architectures and training details for both the normally and adversarially trained variants of the models on both the MNIST and CIFAR-10 datasets. The accuracy of each model on benign data is given in Table 5.

MNIST. Each pixel of the MNIST image data is scaled to $[0, 1]$. We trained four different models on the MNIST dataset, denoted Models A to D [21] which represent a good variety of architectures. For the attacks constrained with the L_∞ distance, we vary the adversary's perturbation budget ϵ from 0 to 0.4, since at a perturbation budget of 0.5, any image can be made solid gray. The model details for the 4 models trained on the MNIST dataset are as follows:

1. Model A (3,382,346 parameters): Conv(64, 5, 5) + Relu, Conv(64, 5, 5) + Relu, Dropout(0.25), FC(128) + Relu, Dropout(0.5), FC + Softmax
2. Model B (710,218 parameters) - Dropout(0.2), Conv(64, 8, 8) + Relu, Conv(128, 6, 6) + Relu, Conv(128, 5, 5) + Relu, Dropout(0.5), FC + Softmax
3. Model C (4,795,082 parameters) - Conv(128, 3, 3) + Relu, Conv(64, 3, 3) + Relu, Dropout(0.25), FC(128) + Relu, Dropout(0.5), FC + Softmax
4. Model D (509,410 parameters) - [FC(300) + Relu, Dropout(0.5)] $\times$ 4, FC + Softmax

Models A and C have both convolutional layers as well as fully connected layers. They also have the same order of magnitude of parameters. Model B, on the other hand, does not have fully connected layers and has an order of magnitude fewer parameters. Similarly, Model D has no convolutional layers and has fewer parameters than all the other models. Models A, B, and C all achieve greater than 99% classification accuracy on the test data. Model D achieves 97.2% classification accuracy, due to the lack of convolutional layers. All models trained on benign data are trained for 6 epochs using a training batch size of 64.

CIFAR-10. Each pixel of the CIFAR-10 image data is in $[0, 255]$. We choose two ResNet architectures for this dataset, which we denote as Resnet-32 [9] and Resnet-28-10 [23]. For the attacks constrained with the L_∞ distance, we vary the adversary's perturbation budget ϵ from 0 to 28. Resnet-32 is a standard 32 layer ResNet with no width expansion, and Resnet-28-10 is a wide ResNet with 28 layers with the width set to 10, based on the best performing ResNet from

Table 2. Attack success rates and average distortion for **untargeted black-box attacks**. Gradient Estimation using Finite Differences is our method, which has performance matching white-box attacks. Attacks on use MNIST an L_∞ constraint of $\epsilon = 0.3$ and adversarial examples are transferred from Model B. For CIFAR-10, the L_∞ constraint is $\epsilon = 8$ and adversarial examples are transferred from Resnet-32

	Baseline		**Gradient Estimation using Finite Differences**				**Transferability-based**			
MNIST			Single-step		Iterative		Single-step		Iterative	
Models	D. of M.	Rand.	FD-xent	FD-logit	IFD-xent	IFD-logit	FGS-xent	FGS-logit	IFGS-xent	IFGS-logit
A	44.8 (5.6)	8.5 (6.1)	51.6 (3.3)	92.9 (6.1)	75.0 (3.6)	**100.0** (2.1)	66.3 (6.2)	80.8 (6.3)	89.8 (4.75)	88.5 (4.75)
B	81.5 (5.6)	7.8 (6.1)	69.2 (4.5)	98.9 (6.3)	86.7 (3.9)	**100.0** (1.6)	-	-	-	-
CIFAR-10			Single-step		Iterative		Single-step		Iterative	
Models	D. of M.	Rand.	FD-xent	FD-logit	IFD-xent	IFD-logit	FGS-xent	FGS-logit	IFGS-xent	IFGS-logit
Resnet-32	9.3 (440.5)	19.4 (439.4)	49.1 (217.1)	86.0 (410.3)	62.0 (149.9)	**100.0** (65.7)	74.5 (439.4)	76.6 (439.4)	99.0 (275.4)	98.9 (275.6)
Resnet-28-10	6.7 (440.5)	17.1 (439.4)	50.1 (214.8)	88.2 (421.6)	46.0 (120.4)	**100.0** (74.9)	-	-	-	-

Zagoruyko and Komodakis [23]. The width indicates the multiplicative factor by which the number of filters in each residual layer is increased.

Resnet-32 is trained for 125,000 steps and Resnet-28-10 is trained for 167,000 steps on the benign training data. All models were trained with a batch size of 128. The two ResNets achieve close to state-of-the-art accuracy [1] on the CIFAR-10 test set, with Resnet-32 at 92.4% and Resnet-28-10 at 94.4%.

4 Zero-query attacks

In this section, we describe existing methods for generating adversarial examples. In all of these attacks, the adversary's perturbation is constrained using the L_∞ distance.

4.1 Baseline attacks

These baseline black-box attacks which can be carried out without any knowledge of or query access to the target model.

Random perturbations With no knowledge of f or the training set, the simplest manner in which an adversary may seek to carry out an attack is by adding a random perturbation to the input [20,8,6]. These perturbations can be generated by any distribution of the adversary's choice and constrained according to an appropriate norm. If we let P be a distribution over $\mathcal{X}$, and $\mathbf{p}$ is a random variable drawn according to P, then a noisy sample is just $\mathbf{x}_{\text{noise}} = \mathbf{x} + \mathbf{p}$. Since random noise is added, it is not possible to generate targeted adversarial examples in a principled manner. This attack is denoted as Rand. throughout.

Difference of means A perturbation aligned with the difference of means of two classes is likely to be effective for an adversary hoping to cause misclassification for a broad range of classifiers [22]. While these perturbations are far

Table 3. Attack success rates and average distortion for **targeted black-box attacks**

MNIST	**Baseline**	**Gradient Estimation using Finite Differences**				**Transfer from Model B**			
		Single-step		Iterative		Single-step		Iterative	
Model	D. of M.	FD-xent	FD-logit	IFD-xent	IFD-logit	FGS-xent	FGS-logit	IFGS-xent	IFGS-logit
A	15.0 (5.6)	30.0 (6.0)	29.9 (6.1)	**100.0** (4.2)	99.7 (2.7)	18.3 (6.3)	18.1 (6.3)	54.5 (4.6)	46.5 (4.2)
B	35.5 (5.6)	29.5 (6.3)	29.3 (6.3)	**99.9** (4.1)	98.7 (2.4)	-	-	-	-
CIFAR-10	**Baseline**	**Gradient Estimation using Finite Differences**				**Transfer from Resnet-28-10**			
		Single-step		Iterative		Single-step		Iterative	
Model	D. of M.	FD-xent	FD-logit	IFD-xent	IFD-logit	FGS-xent	FGS-logit	IFGS-xent	IFGS-logit
Resnet-32	1.2 (440.3)	23.8 (439.5)	23.0 (437.0)	**100.0** (110.9)	**100.0** (89.5)	15.8 (439.4)	15.5 (439.4)	71.8 (222.5)	80.3 (242.6)
Resnet-28-10	0.9 (440.3)	29.2 (439.4)	28.0 (436.1)	100.0 (123.2)	**100.0** (98.3)	-	-	-	-

from optimal for DNNs, they provide a useful baseline to compare against. Adversaries with at least partial access to the training or test sets can carry out this attack. An adversarial sample generated using this method, and with L_∞ constraints, is $\mathbf{x}_{\text{adv}} = \mathbf{x} + \epsilon \cdot \text{sign}(\boldsymbol{\mu}_t - \boldsymbol{\mu}_o)$, where $\boldsymbol{\mu}_t$ is the mean of the target class and $\boldsymbol{\mu}_o$ is the mean of the original ground truth class. For an untargeted attack, $t = \text{argmin}_i\, d(\boldsymbol{\mu}_i - \boldsymbol{\mu}_o)$, where $d(\cdot,\cdot)$ is an appropriately chosen distance function. In other words, the class whose mean is closest to the original class in terms of the Euclidean distance is chosen to be the target. This attack is denoted as D. of M. throughout.

Effectiveness of baseline attacks In the baseline attacks described above, the choice of distribution for the random perturbation attack and the choice of distance function for the difference of means attack are not fixed. Here, we describe the choices we make for both attacks. The random perturbation $\mathbf{p}$ for each sample (for both MNIST and CIFAR-10) is chosen independently according to a multivariate normal distribution with mean $\mathbf{0}$, i.e. $\mathbf{p} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. Then, depending on the norm constraint, either a signed and scaled version of the random perturbation (L_∞) or a scaled unit vector in the direction of the perturbation (L_2) is added. For an untargeted attack utilizing perturbations aligned with the difference of means, for each sample, the mean of the class closest to the original class in the L_2 distance is determined.

As expected, adversarial examples generated using Rand. do not achieve high adversarial success rates in spite of having similar or larger average distortion than the other black-box attacks for both the MNIST and CIFAR-10 models. However, the D. of M. method is quite effective at higher perturbation values for the MNIST dataset. Also, for Model B, the D. of M. attack is more effective than FD-xent. The D. of M. method is less effective in the targeted attack case. Its success rate is comparable to the targeted transferability based attack for Model A as well.

The relative effectiveness of the two baseline methods is reversed for the CIFAR-10 dataset, however, where Rand. outperforms D. of M. considerably as ϵ is increased. This indicates that the models trained on MNIST have normal vectors to decision boundaries which are more aligned with the vectors along the difference of means as compared to the models on CIFAR-10.

4.2 Transferability based attacks

Here we describe black-box attacks that assume the adversary has access to a representative set of training data in order to train a local model. One of the earliest observations with regards to adversarial examples for neural networks was that they *transfer*; i.e, adversarial attack samples generated for one network are also adversarial for another network. This observation directly led to the proposal of a black-box attack where an adversary would generate samples for a local network and transfer these to the target model, which is referred to as a Transferability based attack. Targeted transferability attacks are carried out using locally generated targeted white-box adversarial examples.

Single local model. These attacks use a *surrogate local model* f^s to craft adversarial examples, which are then submitted to f in order to cause misclassification. Most existing black-box attacks are based on *transferability from a single local model* [17,15]. The different attack strategies to generate adversarial instances introduced in Section 1 can be used here to generate adversarial instances against f^s, so as to attack f.

Ensemble of local models. Since it is not clear which local model f^s is best suited for generating adversarial examples that transfer well to the target model f, Liu et al. [14] propose the generation of adversarial examples for an ensemble of local models. This method modifies each of the existing transferability attacks by substituting a sum over the loss functions in place of the loss from a single local model.

Concretely, let the ensemble of m local models to be used to generate the local loss be $\{f^{s_1}, \dots, f^{s_m}\}$. The ensemble loss is then computed as $\ell_{\text{ens}}(\mathbf{x}, y) = \sum_{i=1}^{m} \alpha_i \ell_{f^{s_i}}(\mathbf{x}, y)$, where α_i is the weight given to each model in the ensemble. The FGS attack in the ensemble setting then becomes $\mathbf{x}_{\text{adv}} = \mathbf{x} + \epsilon \cdot \text{sign}(\nabla_{\mathbf{x}} \ell_{\text{ens}}(\mathbf{x}, y))$. The Iterative FGS attack is modified similarly. In Table 4, we can see that Transferability attack (local model ensemble) performs well even in the targeted attack case, while Transferability attack (single local model) is usually only effective for untargeted attacks. The intuition is that while one model's gradient may not be adversarial for a target model, it is likely that at least one of the gradient directions from the ensemble represents a direction that is somewhat adversarial for the target model.

Transferability attack results. For the transferability experiments, we choose to transfer from Model B for MNIST dataset and from Resnet-28-10 for CIFAR-10 dataset. Adversarial examples generated using single-step methods and transferred from Model B to the other models have higher success rates for untargeted attacks when they are generated using the logit loss as compared to the cross entropy loss as can be seen in Table 2. For iterative adversarial examples, however, the untargeted attack success rates are roughly the same for both loss functions. As has been observed before, the adversarial success rate for targeted attacks

Table 4. Adversarial success rates for **transferability-based attacks** on Model A (MNIST) at $\epsilon = 0.3$. Numbers in parentheses beside each entry give the average distortion $\Delta(\mathbf{X}, \mathbf{X}_{\text{adv}})$ over the test set. This table compares the effectiveness of using a single local model to generate adversarial examples versus the use of a local ensemble.

	Untargeted Transferability to Model A			
	Single-step		Iterative	
Source	FGS-xent	FGS-logit	IFGS-xent	IFGS-logit
B	66.3 (6.2)	80.8 (6.3)	89.8 (4.75)	88.5 (4.75)
B,C	68.1 (6.2)	89.8 (6.3)	95.0 (4.8)	97.1 (4.9)
B,C,D	56.0 (6.3)	88.7 (6.4)	73.5 (5.3)	94.4 (5.3)
	Targeted Transferability to Model A			
	Single-step		Iterative	
Source	FGS-T (xent)	FGS-T (logit)	IFGS-T (xent)	IFGS-T (logit)
B	18.3 (6.3)	18.1 (6.3)	54.5 (4.6)	46.5 (4.2)
B,C	23.0 (6.3)	23.0 (6.3)	76.7 (4.8)	72.3 (4.5)
B,C,D	25.2 (6.4)	25.1 (6.4)	74.6 (4.9)	66.1 (4.7)

with transferability is much lower than the untargeted case, even when iteratively generated samples are used. On the MNIST dataset, the highest targeted transferability rate is 54.5% (Table 3) as compared to 89.8% in the untargeted case (Table 2).

One attempt to improve the transferability rate is to use an ensemble of local models, instead of a single one. The results for this on the MNIST data are presented in Table 4. In general, both untargeted and targeted transferability increase when an ensemble is used. However, the increase is not monotonic in the number of models used in the ensemble, and we can see that the transferability rate for IFGS-xent samples falls sharply when Model D is added to the ensemble. This may be due to it having a very different architecture as compared to the models, and thus also having very different gradient directions. This highlights one of the pitfalls of transferability, where it is important to use a local surrogate model similar to the target model for achieving high attack success rates.

5 Query reduction techniques: analysis and results

5.1 Query reduction using PCA components

A more principled way to reduce the number of queries the adversary has to make to estimate the gradient is to compute directional derivatives along the principal components as determined by principal component analysis (PCA) [18], which requires the adversary to have access to a set of data which is representative of the training data. PCA minimizes reconstruction error in terms of the L_2 norm; i.e., it provides a basis in which the Euclidean distance to the original sample from a sample reconstructed using a subset of the basis vectors is the smallest.

Algorithm 1 Gradient estimation with query reduction using PCA components

```
Input: $\mathbf{x}$, $k$, $\mathbf{U}$, $\delta$, $g(\cdot)$
Output: Estimated gradient $\hat{\nabla}_{\mathbf{x}} g(\mathbf{x})$ of $g(\cdot)$ at $\mathbf{x}$
1: for $i \leftarrow 1$ to $k$ do
2:     Initialize $\mathbf{v}$ such that $\mathbf{v} = \frac{\mathbf{u}_i}{\|\mathbf{u}_i\|}$, where $\mathbf{u}_i$ is the $i^{\text{th}}$ column of $\mathbf{U}$
3:     Compute
           $$\alpha^i(\mathbf{v}) = \frac{g(\mathbf{x}+\delta\mathbf{v}) - g(\mathbf{x}-\delta\mathbf{v})}{2\delta},$$
       which is the two-sided approximation of the directional derivative along $\mathbf{v}$
4:     Update $\hat{\nabla}_{\mathbf{x}} g(\mathbf{x})^i = \hat{\nabla}_{\mathbf{x}} g(\mathbf{x})^{i-1} + \alpha^i(\mathbf{v})\mathbf{v}$
5: end for
6: Set $\hat{\nabla}_{\mathbf{x}} g(\mathbf{x}) = \hat{\nabla}_{\mathbf{x}} g(\mathbf{x})^k$
```

Concretely, let the samples the adversary wants to misclassify be column vectors $\mathbf{x}^i \in \mathbb{R}^d$ for $i \in \{1, \ldots, n\}$ and let $\mathbf{X}$ be the $d \times n$ matrix of centered data samples (i.e. $\mathbf{X} = [\tilde{\mathbf{x}}^1 \tilde{\mathbf{x}}^2 \ldots \tilde{\mathbf{x}}^n]$, where $\tilde{\mathbf{x}}^i = \mathbf{x} - \frac{1}{n}\sum_{j=1}^{n} \mathbf{x}^j$). The principal components of $\mathbf{X}$ are the normalized eigenvectors of its sample covariance matrix $\mathbf{C} = \mathbf{X}\mathbf{X}^{\mathsf{T}}$. Since $\mathbf{C}$ is a positive semidefinite matrix, there is a decomposition $\mathbf{C} = \mathbf{U}\Lambda\mathbf{U}^{\mathsf{T}}$ where $\mathbf{U}$ is an orthogonal matrix, $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_d)$, and $\lambda_1 \geq \ldots \geq \lambda_d \geq 0$. Thus, $\mathbf{U}$ in Algorithm 1 is the $d \times d$ matrix whose columns are unit eigenvectors of $\mathbf{C}$. The eigenvalue λ_i is the variance of $\mathbf{X}$ along the i^{th} component.

In Algorithm 1, $\mathbf{U}$ is the $d \times d$ matrix whose columns are the principal components $\mathbf{u}_i$, where $i \in [d]$. The quantity being estimated in Algorithm 1 is an approximation of the gradient in the PCA basis:

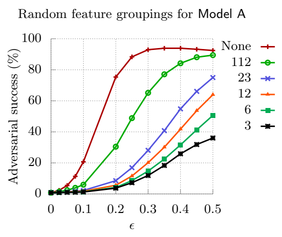
$$(\nabla_{\mathbf{x}} g(\mathbf{x}))^k = \sum_{i=1}^{k} \left(\nabla_{\mathbf{x}} g(\mathbf{x})^{\mathsf{T}} \frac{\mathbf{u}_i}{\|\mathbf{u}_i\|} \right) \frac{\mathbf{u}_i}{\|\mathbf{u}_i\|},$$

where the term on the left represents an approximation of the true gradient by the sum of its projection along the top k principal components. In Algorithm 1, the weights of the representation in the PCA basis are approximated using the approximate directional derivatives along the principal components.

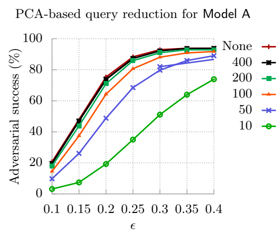
5.2 Effect of dimension on Gradient Estimation attacks with query reduction

We consider the effectiveness of Gradient Estimation with random grouping based query reduction and the logit loss (GE-QR (RG-k, logit)) on Model A on MNIST data in Figure 1a, where k is the number of indices chosen in each iteration of Algorithm 1 of the main paper. Thus, as k increases and the number of groups decreases, we expect adversarial success to decrease as gradients over larger groups of features are averaged. This is the effect we see in Figure 1a, where the adversarial success rate drops from 93% to 63% at $\epsilon = 0.3$ as k increases from 1 to 7. Grouping with $k = 7$ translates to 112 queries per MNIST image, down from 784. Thus, in order to achieve high adversarial success rates with the random grouping method, larger perturbation magnitudes are needed.

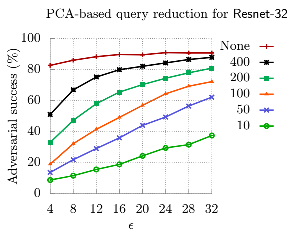
On the other hand, the PCA-based approach GE-QR (PCA-k, logit) is much more effective, as can be seen in Figure 1b. Using 100 principal components to estimate the gradient for Model A on MNIST as in Algorithm 1, the adversarial success rate at $\epsilon = 0.3$ is 88.09%, as compared to 92.9% without any query reduction. Similarly, using 400 principal components for Resnet-32 on CIFAR-10 (Figure 1c), an adversarial success rate of 66.9% can be achieved at $\epsilon = 8$. At $\epsilon = 16$, the adversarial success rate rises to 80.1%.



(a) Gradient Estimation attack with query reduction using random grouping and the logit loss (GE-QR (RG-k, logit)) on Model A (MNIST, $d = 784$). The adversarial success rate decreases as the number of groups $\lceil \frac{d}{k} \rceil$ is decreased, where k is the size of the group and d is the dimension of the input.



(b) Gradient Estimation attack with query reduction using PCA components and the logit loss (GE-QR (PCA-k, logit)) on Model A (MNIST, $d = 784$). The adversarial success rates decrease as the number of principal components k used for estimation is decreased. Relatively high success rates are maintained even for $k = 50$.



(c) Gradient Estimation attack with query reduction using PCA components and the logit loss (GE-QR (PCA-k, logit)) on Resnet-32 (CIFAR-10). Relatively high success rates are maintained even for $k = 400$.

Fig. 1. Adversarial success rates for Gradient Estimation attacks with query reduction (FD-QR (*Technique*, logit)) on Model A (MNIST) and Resnet-32 (CIFAR-10), where *Technique* is either PCA or RG. 'None' refers to FD-logit, the case where the number of queries is $2d$, where d is the dimension of the input.

6 Adversarial training: description and further results

Adversarial training [20,8] defends against adversarial examples by generating them during the training phase and using them during training. The standard loss function for a neural network f is modified as follows:

$$\tilde{\ell}(\mathbf{x}, y) = \alpha \ell_f(\mathbf{x}, y) + (1 - \alpha)\ell_f(\mathbf{x}_{\text{adv}}, y), \quad (12)$$

where y is the true label of the sample $\mathbf{x}$. The underlying objective of this modification is to make the neural networks more robust by penalizing it during training to count for adversarial examples. During training, the adversarial examples are computed with respect to the current state of the network using an appropriate method such as FGSM.

Ensemble adversarial training. [21] proposed an extension of the adversarial training paradigm which is called *ensemble adversarial training*. As the name suggests, in ensemble adversarial training, the network is trained with adversarial examples from multiple networks.

Iterative adversarial training. A further modification of the adversarial training paradigm proposes training with adversarial examples generated using iterative methods such as the iterative FGSM attack described earlier [16].

Dataset (Model)	Benign	Adv	Adv-Ens	Adv-Iter
MNIST (A)	99.2	99.4	99.2	99.3
CIFAR-10 (Resnet-32)	92.4	92.1	91.7	79.1

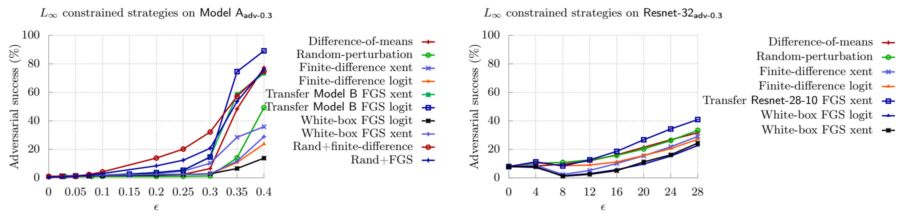
Table 5. Accuracy of models on the benign test data

For all adversarially trained models, each training batch contains 128 samples of which 64 are benign and 64 are adversarial examples (either FGSM or iterative FGSM).This implies that the loss for each is weighted equally during training; i.e., in Eq. 12, α is set to 0.5. Networks using standard and ensemble adversarial training are trained for 12 epochs, while those using iterative adversarial training are trained for 64 epochs.

We train variants of Resnet-32 using adversarial examples with an L_∞ constraint of 8. Resnet-32 $_{\text{adv-8}}$ is trained with FGS samples with the same constraint, and Resnet-32 $_{\text{ens-adv-8}}$ is trained with pre-generated FGS samples from Resnet-32 and Std.-CNN as well as FGS samples. Resnet-32 $_{\text{adv-iter-8}}$ is trained with iterative FGS samples using $t = 10$ and $\alpha = 1.0$. The adversarial variants of Resnet-32 are trained for 80,000 steps. Table 5 shows the accuracy of these models with various defenses on benign test data.

6.1 Single-step attacks on defenses

In Figure 2a, we can see that both single-step black-box and white-box attacks have much lower adversarial success rates on Model A$_{\text{adv-0.3}}$ as compared to Model A. The success rate of the Gradient Estimation attacks matches that of white-box attacks on these adversarially trained networks as well. To overcome this, we add an initial random perturbation to samples before using the Gradient Estimation attack with Finite Differences and the logit loss (FD-logit). These are then the most effective single step black-box attacks on Model A$_{\text{adv-0.3}}$ at $\epsilon = 0.3$ with an adversarial success rate of 32.2%, surpassing the Transferability attack (single local model) from B.



(a) Model $\mathsf{A}_{\mathsf{adv\text{-}0.3}}$ (MNIST) (b) Resnet-32 $_{\mathsf{adv\text{-}8}}$ (CIFAR-10)

Fig. 2. Effectiveness of various single step black-box attacks against adversarially trained models. On the MNIST model, Model $\mathsf{A}_{\mathsf{adv\text{-}0.3}}$ the attack with the highest performance up till $\epsilon = 0.3$ is the Gradient Estimation attack using Finite Differences with initially added randomness. Beyond this, the Transferability attack (single local model) using samples from Model B performs better. On the CIFAR-10 model Resnet-32 $_{\mathsf{adv\text{-}8}}$, the best performing attack is the Transferability attack (single local model) using samples from Resnet-28-10.

In Figure 2b, we again see that the Gradient Estimation attacks using Finite Differences (FD-xent and FD-logit) and white-box FGS attacks (FGS-xent and FGS-logit) against Resnet-32. As ϵ is increased, the attacks that perform the best are Random Perturbations (Rand.), Difference-of-means (D. of M.), and Transferability attack (single local model) from Resnet-28-10 with the latter performing slightly better than the baseline attacks. This is due to the 'gradient masking' phenomenon and can be overcome by adding random perturbations as for MNIST. An interesting effect is observed at $\epsilon = 4$, where the adversarial success rate is *higher* than at $\epsilon = 8$. The likely explanation for this effect is that the model has overfitted to adversarial examples at $\epsilon = 8$. Our Gradient Estimation attack closely tracks the adversarial success rate of white-box attacks in this setting as well.

Increasing effectiveness of single-step attacks using initial random perturbation. Since the Gradient Estimation attack with Finite Differences (FD-xent and FD-logit) were not performing well due the masking of gradients at the benign sample $\mathbf{x}$, we added an initial random perturbation to escape this low-gradient region as in the RAND-FGSM attack [21]. Figure 3 shows the effect of adding an initial L_∞-constrained perturbation of magnitude 0.05. With the addition of a random perturbation, FD-logit has a much improved adversarial success rate on Model $\mathsf{A}_{\mathsf{adv\text{-}0.3}}$, going up to 32.2% from 2.8% without the perturbation at a total perturbation value of 0.3. It even outperforms the white-box FGS (FGS-logit) with the same random perturbation added. This effect is also observed for Model $\mathsf{A}_{\mathsf{adv\text{-}ens\text{-}0.3}}$, but Model $\mathsf{A}_{\mathsf{adv\text{-}iter\text{-}0.3}}$ appears to be resistant to single-step gradient based attacks. Thus, our attacks work well for single-step attacks on DNNs with standard and ensemble adversarial training, and achieve performance levels close to that of white-box attacks.

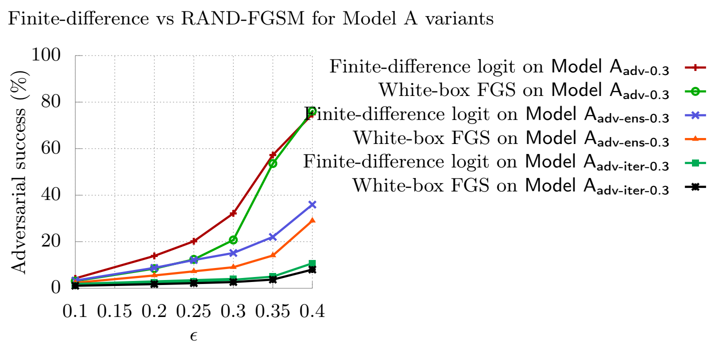


Fig. 3. Increasing the effectiveness of FD-logit attacks on Models $A_{\text{adv-0.3}}$, $A_{\text{adv-ens-0.3}}$ and $A_{\text{adv-iter-0.3}}$ (MNIST) by adding an initial L_∞ constrained random perturbation of magnitude 0.01.

6.2 Targeted attack results

In Table 6, the attack success rate and distortion levels for targeted attacks on adversarially trained models are given. We see that Gradient Estimation match the attack success rates and distortion levels of white-box attacks.

Table 6. Targeted black-box attacks for models with **adversarial training**: attack success rates and average distortion $\Delta(\mathbf{X}, \mathbf{X}_{\text{adv}})$. **Top**: MNIST, $\epsilon = 0.3$. **Bottom**: CIFAR-10, $\epsilon = 8$.

Dataset	**White-box**		**Gradient Estimation (FD)**		**Gradient Estimation (Query Reduction)**			
MNIST Models	Single-step FGS (logit)	Iterative IFGS (logit)	Single-step [1568] FD-logit	Iterative [62720] IFD-logit	Single-step [$\sim$ 200] PCA-100	RG-8	Iterative [8000] PCA-100	RG-8
$A_{\text{adv-0.3}}$	1.6 (6.1)	48.4 (3.9)	1.7 (6.0)	47.8 (3.6)	0.8 (5.9)	0.7 (5.4)	13.9 (4.8)	4.3 (2.7)
$A_{\text{adv-ens-0.3}}$	0.9 (6.3)	61.0 (3.7)	0.9 (6.3)	**63.0 (3.7)**	0.6 (6.0)	0.4 (6.4)	17.0 (4.4)	10.0 (2.3)
$A_{\text{adv-iter-0.3}}$	1.3 (7.5)	2.0 (3.8)	1.3 (6.5)	2.0 (3.4)	0.4 (6.0)	0.1 (7.1)	2.0 (4.4)	1.0 (2.2)
CIFAR-10 Models	Single-step FGS (logit)	Iterative IFGS (logit)	Single-step [6144] FD-logit	Iterative [61440] IFD-logit	Single-step [$\sim$ 800] PCA-400	RG-8	Iterative [$\sim$ 8000] PCA-400	RG-8
Resnet-32 $_{\text{adv-8}}$	1.0 (435.2)	99.0 (122.2)	1.2 (441.5)	**99.5 (123.4)**	0.8 (462.2)	0.9 (465.5)	78.2 (147.3)	75.0 (143.1)
Resnet-32 $_{\text{adv-ens-8}}$	1.4 (435.2)	95.0 (131.5)	1.6 (437.8)	**95.7 (134.3)**	0.9 (438.3)	1.0 (439.4)	61.3 (153.8)	62.2 (150.7)
Resnet-32 $_{\text{adv-iter-8}}$	12.4 (430.4)	18.1 (378.5)	12.5 (431.2)	**18.3 (379.4)**	3.1 (436.7)	2.9 (439.3)	4.1 (216.4)	4.0 (215.3)

7 Countermeasures results

The main countermeasure to our attack that we consider is rounding the probabilities output by a classifier to a smaller number of decimal places. We note that no MLaaS provider we are aware of uses this countermeasure, and that this method could severely inhibit usability for legitimate users wishing to carry out any post-processing of the output probabilities.

When the output probabilities are rounded down to *2 decimal places*, the Gradient Estimation attack with Finite Differences on MNIST, using the logit loss (FD-logit) is unable to generate adversarial examples for $\delta = 0.01$ and achieves an attack success rate of 7% with a distortion of 0.55 with $\delta = 1.0$. In comparison, FD-xent achieves 11% attack success with a distortion of 0.85 with $\delta = 1.0$. Using the Gradient Estimation attack with query reduction (GE-QR (RG-k, xent)), an attack success rate of 19% can be achieved, using a group size of 100 and $\delta = 1.0$. Similarly, for IGE-QR (RG-k, xent), an attack success rate of 28% can be achieved with a group size of 100. For CIFAR-10 as well, we find the cross-entropy loss based attack to perform better with this countermeasure.

The overall takeaway is that due to the countermeasure, if the images used to estimate the loss are too close to one another, as is the case with a small value of δ, it is not possible to find adversarial examples reliably. However, non-trivial attack success rates can be achieved with larger δ and larger groups.

8 Comparison with previous work

As already stated in the main paper, compared to the ZOO attack of Chen et al. [4], our attack uses 192× fewer queries for MNIST and 67× fewer for CIFAR-10 to achieve similar success rates at almost identical distortion levels. This is shown in Table 7. We now discuss the impact this has on the time taken for the attacks to run. For iterative attacks with no query reduction, with 40 iterations per sample (α set to 0.01), both IFD-xent and IFD-xent-T taking about 2.4 seconds per sample. Similarly, IFD-logit and IFD-logit-T take about 3.5 seconds per sample. With query reduction, using IGE-QR (PCA-k, logit) with $k = 100$ and IGE-QR (RG-k, logit) with $k = 8$, the time taken is just 0.5 seconds per sample. In contrast, the fastest attack from [4], the ZOO-ADAM attack, takes around 80 seconds per sample for MNIST, which is 24× slower than the Iterative Finite Difference attacks and around 160× slower than the Iterative Gradient Estimation attacks with query reduction. For Resnet-32 on the CIFAR-10 dataset, FD-xent, FD-xent-T, FD-logit and FD-logit-T all take roughly 3s per sample. The iterative variants of these attacks with 10 iterations (α set to 1.0) take roughly 30s per sample. Using query reduction, both IGE-QR (PCA-k, logit) with $k = 100$ with 10 iterations takes just 5s per sample. The time required per sample increases with the complexity of the network, which is observed even for white-box attacks. For the CIFAR-10 dataset, the fastest attack from [4] takes about 206 seconds per sample, which is 7× slower than the Iterative Finite Difference attacks and around 40× slower than the Iterative Gradient Estimation attacks with query reduction. We further note that while we use state-of-the-art classifiers on CIFAR-10, achieving classification accuracies in excess of 90%, the classifiers used by Chen et. al. only achieve accuracies around 80%.

Comparing our attack to that of Brendel et al., we note that their attack takes up to 1.2×10^6 queries to converge to an adversarial example. Further, we demonstrate attacks using as few as 192 queries to a real-world target model on a 256x256 pixel image with a perturbation magnitude of 1.8×10^{-3} in the distortion metric (squared L2 distance, normalized by resolution) used by Brendel et al. [2]. In comparison, in Figure 4 in Brendel et al. [2], 1229 queries are required to find an adversarial example for a locally-held model (not a real-world model) with a perturbation of 2.1×10^{-3} for an image of roughly the same resolution.

Table 7. Comparison of attack success (AS) and distortion (dist.) for Gradient Estimation ($\epsilon = 0.3$) and ZOO attacks on Model A on MNIST.

Attack Type	Untargeted	Targeted	Attack efficiency	
Query-based attack	AS (Dist.)	AS (Dist.)	Queries	Avg. Time (s)
Finite Diff.	92.9 (6.1)	29.9 (6.1)	1568	8.8×10^{-2}
Gradient Estimation (RG-8)	61.5 (6.0)	15.9 (5.9	196	1.1×10^{-2}
Iter. Finite Diff.	100.0 (2.1)	99.7 (2.7)	62720	3.5
Iter. Gradient Estimation (RG-8)	98.4 (1.9)	73.8 (2.5)	8000	0.43
ZOO	100.0 (1.5)	100.0 (2.1)	7.7×10^5	55.4

9 Experiments on Clarifai, a real-world system

In this section, we present the complete set of images we experimented with to attack the Not Suitable For Work (NSFW) and Content Moderation models hosted by Clarifai [5].

This section contains images that some viewers may find offensive or disturbing. These are used purely for research purposes and we apologize for any discomfort caused.

Many of the original images are larger and have been scaled down for presentation purposes. All the images we use have usage rights 'Labeled for noncommercial reuse with modification'. The adversarial examples of 'drugs' in Table 8 were generated using 197 queries to the target model while the adversarial images of 'gore' needed 110 and 220 queries each (top to bottom).

The adversarial images in Table 9 needed 810, 1150, 1600 ,1400 and 810 queries respectively, from top to bottom.

9.1 Quantitative attack results against Clarifai:

We have conducted additional experiments to show quantitative results for attack success rate against Clarifai's NSFW model.

Our methodology was as follows. To come up with a representative sample of NSFW images, we collected public images form the Internet that were indexed with terms such as 'erotic.' We then selected 30 images similar to the default images provided by Clarifai which were likely to be classified as 'NSFW' by the model. For large images, we downsampled them to a size suitable for embedding in a discussion forum. The authors clearly identified all of these as NSFW. The model classified 21 of the images as 'NSFW.'

We then carried out our Iterative Gradient Estimation attack on the NSFW model and our sample of 21 correctly classified images using Random Grouping as the query reduction method. For our initial set of parameters, we set the perturbation parameter ϵ to 16, the random group size to $10,000$, the estimation parameter δ to 1.0 and the number of iterations to 5. With this choice of parameters, our attack caused 16 of the 21 images to be misclassified as 'SFW', an attack success rate of **76.2%**. We then modified the parameters by decreasing the group size and increasing the number of iterations in order to misclassify the 5 remaining images. With these changes, the attack success rate was 20 of 21 or **95.2%**. The average number of queries needed for each image over all successful attacks was 228. The full set of images can be found at `https://sunblaze-ucb.github.io/blackbox-attacks/`.

References

1. Benenson, R.: Classification datasets results. `http://rodrigob.github.io/are_we_there_yet/build/#classification-dataset-type` (2016), accessed: 2017-08-22
2. Brendel, W., Rauber, J., Bethge, M.: Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In: International Conference on Learning Representations (2018)
3. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: IEEE Symposium on Security and Privacy, 2017 (2017)
4. Chen, P.Y., Zhang, H., Sharma, Y., Yi, J., Hsieh, C.J.: Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In: 11th ACM Workshop on Artificial Intelligence and Security (2017)
5. Clarifai | image & video recognition API. `https://clarifai.com`, accessed: 2017-08-22
6. Fawzi, A., Fawzi, O., Frossard, P.: Analysis of classifiers' robustness to adversarial perturbations. arXiv preprint arXiv:1502.02590 (2015)
7. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT Press (2016)
8. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: International Conference on Learning Representations (2015)
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
10. Kennedy, J.: Particle swarm optimization. In: Encyclopedia of machine learning, pp. 760–766. Springer (2011)
11. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images (2009)
12. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533 (2016)
13. LeCun, Y., Cortes, C.: The MNIST database of handwritten digits (1998)
14. Liu, Y., Chen, X., Liu, C., Song, D.: Delving into transferable adversarial examples and black-box attacks. In: ICLR (2017)

15. Moosavi-Dezfooli, S.M., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. In: IEEE Conference on Computer Vision and Pattern Recognition (2016)
16. Mądry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. In: International Conference on Learning Representations (2018)
17. Papernot, N., McDaniel, P., Goodfellow, I.: Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. arXiv preprint arXiv:1605.07277 (2016)
18. Shlens, J.: A tutorial on principal component analysis. arXiv preprint arXiv:1404.1100 (2014)
19. Spall, J.C.: Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. IEEE transactions on automatic control **37**(3), 332–341 (1992)
20. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. In: International Conference on Learning Representations (2014)
21. Tramèr, F., Kurakin, A., Papernot, N., Boneh, D., McDaniel, P.: Ensemble adversarial training: Attacks and defenses. In: International Conference on Learning Representations (2018)
22. Tramèr, F., Papernot, N., Goodfellow, I., Boneh, D., McDaniel, P.: The space of transferable adversarial examples. arXiv preprint arXiv:1704.03453 (2017)
23. Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2016)

Table 8. Benign and targeted adversarial images of drugs and gore. All classification results are from the 'Content Moderation' model hosted by Clarifai.

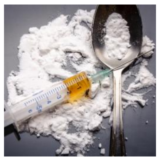
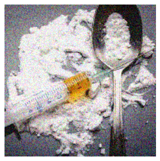
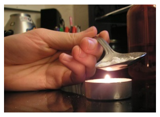
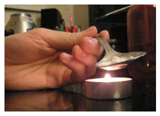
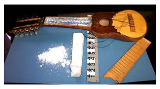
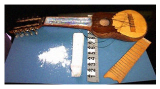
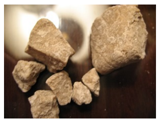
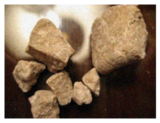
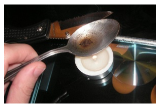
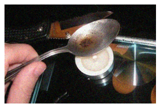
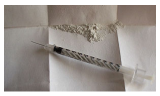
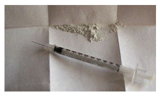
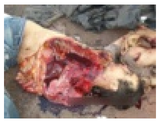
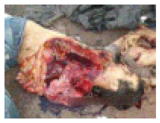
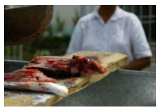
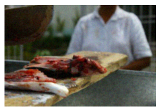
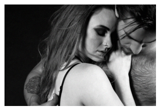
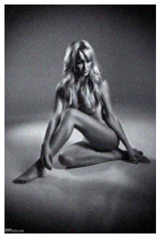
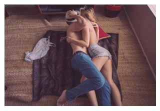
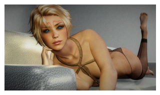
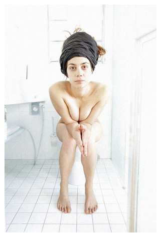
Benign image	Classification	Adversarial image	Classification
	'drugs', confidence = 1.0		'safe', confidence = 0.78
	'drugs', confidence = 0.59		'explicit', confidence = 0.53
	'drugs', confidence = 0.99		'safe', confidence = 0.55
	'drugs', confidence = 0.99		'safe', confidence = 0.77
	'drugs', confidence = 1.0		'safe', confidence = 0.76
	'drugs', confidence = 1.0		'safe', confidence = 0.67
	'gore', confidence = 1.0		'safe', confidence = 0.69
	'gore', confidence = 1.0		'safe', confidence = 0.55

Table 9. Benign and targeted adversarial images of suggestive and NSFW images. The *top two* images are classified using the 'Content Moderation' model while the *bottom three* images are classified using the 'NSFW' model.

Benign image	Classification	Adversarial image	Classification
	'suggestive', confidence = 0.72		'safe', confidence = 0.58
	'suggestive', confidence = 0.99		'safe', confidence = 0.79
	'nsfw', confidence = 0.76		'safe', confidence = 0.55
	'nsfw', confidence = 0.88		'sfw', confidence = 0.56
	'nsfw', confidence = 0.85		'sfw', confidence = 0.64