Black-box Adversarial Examples

A. M. Sadeghzadeh, Ph.D.

Sharif University of Technology
Computer Engineering Department (CE)
Data and Network Security Lab (DNSL)

May 8, 2023

Today's Agenda

Recap

Black-box Adversarial Examples

The only capability of the black-box adversary is to observe the **output** given by the target model to **chosen inputs**.

Black-box Adversarial Examples

The only capability of the black-box adversary is to observe the **output** given by the target model to **chosen inputs**.

- In this setting, back propagation for **gradient computation of the targeted model is prohibited**.

Black-box Adversarial Examples

The only capability of the black-box adversary is to observe the **output** given by the target model to **chosen inputs**.

- In this setting, back propagation for **gradient computation of the targeted model is prohibited**.
- Threat model
  - **Score-based** (the adversary has access to the target model scores)
  - **Decision-based** (the adversary has only access to the target model label)

Black-box Adversarial Examples

The only capability of the black-box adversary is to observe the **output** given by the target model to **chosen inputs**.

- In this setting, back propagation for **gradient computation of the targeted model is prohibited**.
- Threat model
  - **Score-based** (the adversary has access to the target model scores)
  - **Decision-based** (the adversary has only access to the target model label)
- Types
  - **Transfer-based** and **Query-based**

Black-box Adversarial Examples

The only capability of the black-box adversary is to observe the **output** given by the target model to **chosen inputs**.

- In this setting, back propagation for **gradient computation of the targeted model is prohibited**.
- Threat model
  - **Score-based** (the adversary has access to the target model scores)
  - **Decision-based** (the adversary has only access to the target model label)
- Types
  - **Transfer-based** and **Query-based**

Transfer-based

- Create a surrogate model with high **fidelity** to the taget model.
- Generate adversarial examples on the surrogate model using **white-box** attacks.
- Then, **transfer** pregenerated adversarial examples to the target model.

Black-box Adversarial Examples

The only capability of the black-box adversary is to observe the **output** given by the target model to **chosen inputs**.

- In this setting, back propagation for **gradient computation of the targeted model is prohibited**.
- Threat model
  - **Score-based** (the adversary has access to the target model scores)
  - **Decision-based** (the adversary has only access to the target model label)
- Types
  - **Transfer-based** and **Query-based**

Transfer-based

- Create a surrogate model with high **fidelity** to the taget model.
- Generate adversarial examples on the surrogate model using **white-box** attacks.
- Then, **transfer** pregenerated adversarial examples to the target model.

Query-based

- Based on the target model responses for consecutive queries
  - Gradient estimation
    - Based on zero-order (ZO) optimization algorithms
  - Search-based
    - Based on choosing a search strategy using a search distribution.

ZOO: Zeroth Order Optimization Based Black-box Attacks

Inspired by $C\&W$ attack, we propose a new loss function based on the output F of a DNN, which is defined as

$$f(x,t) = max\{\underset{i \neq t}{max} \log[F(x)]_i - \log[F(x)]_t, -\kappa\}$$

where $\kappa \geq 0$. We find that the **log operator** is essential to our black-box attack.

For **untargeted attacks**, an adversarial attack is successful when x is classified as any class other than the original class label t_0. A similar loss function can be used

$$f(x) = max\{\log[F(x)]_{t_0} - \underset{i \neq t_0}{max} \log[F(x)]_i, -\kappa\}$$

where t_0 is the original class label for x.

Zeroth Order Optimization on the Loss Function

We discuss our optimization techniques **for any general function** f used for attacks. We use the **Symmetric Difference Quotient** to estimate the gradient $\hat{g}_i = \frac{\partial f(x)}{\partial x_i}$

$$\hat{g}_i = \frac{\partial f(x)}{\partial x_i} \approx \frac{f(x + he_i) - f(x - he_i)}{2h},$$

where h is a small constant ($h = 0.0001$) and e_i is a standard basis vector with only the i-th component as 1.

Zeroth Order Optimization on the Loss Function

We discuss our optimization techniques **for any general function** f used for attacks. We use the **Symmetric Difference Quotient** to estimate the gradient $\hat{g}_i = \frac{\partial f(x)}{\partial x_i}$

$$\hat{g}_i = \frac{\partial f(x)}{\partial x_i} \approx \frac{f(x + he_i) - f(x - he_i)}{2h},$$

where h is a small constant ($h = 0.0001$) and e_i is a standard basis vector with only the i-th component as 1.

For any $x \in \mathbb{R}^p$, we need to **evaluate the objective function $2p$ times** to estimate gradients of all p coordinates.

- This naive solution is too expensive in practice.
- Even for an input image size of $64 \times 64 \times 3$, one full gradient descent step requires $24,576$ evaluations, and typically hundreds of iterations may be needed until convergence.

Therefore, using stochastic **gradient descent for minimizing objective function is too expensive**.

Evaluation

We compare ZOO with

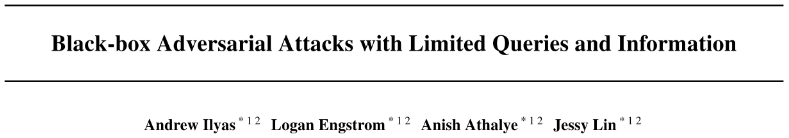
- Carlini & Wagner's (C&W) white-box attack
- The substitute model based black-box attack (JBDA attack to create substitute model)

Setup

- Batch size of $B = 128$
  - Evaluate 128 gradients and update 128 coordinates per iteration.
- Set $\kappa = 0$
- Binary search up to 9 times to find the best c in C&W attack.
- We run 3000 iterations for MNIST and 1000 iterations for CIFAR10 (gradient descent iteration, each iteration update 128 coordinates)
  - $3000 \times 128 \times 2 \times 9 = 6,912,000$ queries for single adversarial example on MNIST model
  - $1000 \times 128 \times 2 \times 9 = 2,304,000$ queries for single adversarial example on CIFAR10 model
- Since the image size of MNIST and CIFAR10 is small, we do not reduce the dimension of attack-space or use hierarchical attack and importance sampling.

Black-box Adversarial Attacks with Limited Queries and Information

Black-box Adversarial Attacks with Limited Queries and Information



Black-box Adversarial Attacks with Limited Queries and Information

Andrew Ilyas [*1 2] **Logan Engstrom** [*1 2] **Anish Athalye** [*1 2] **Jessy Lin** [*1 2]

Abstract

Previous methods using **substitute networks** or **coordinate-wise gradient estimation** for **targeted black-box attacks** require on the order of **millions of queries** to attack an ImageNet classifier.

We propose the variant of **Natural Evolutionary Strategies (NES)** as a method for generating targeted adversarial examples in the query-limited setting.

- We use **NES as a black-box gradient estimation** technique and employ **PGD** (as used in white-box attacks) with the estimated gradient to construct adversarial examples.

The method is **2-3 orders of magnitude more query-efficient than ZOO**.

Black-box Attack

In the black-box setting

- The adversary can supply any input x and receive the **predicted class probabilities** $P(y|x)$ for all classes y.
- This setting **does not allow the adversary to analytically compute the gradient** $\nabla P(y|x)$ as is doable in the white-box case.

Threat Models

The following threat models as more limited variants of the black-box setting that reflect access and resource restrictions in **real-world systems**.

1. **Query-limited setting**
2. **Partial-information setting**
3. **Label-only setting**

Threat Models

The following threat models as more limited variants of the black-box setting that reflect access and resource restrictions in **real-world systems**.

1. **Query-limited setting**

   In the query-limited setting, the **attacker has a limited number of queries** to the classifier. A limit on the number of queries can be a result of limits on other resources, such as a monetary limit if the attacker **incurs a cost for each query**.

   - Example. The **Clarifai NSFW** (Not Safe for Work) detection API is a binary classifier that outputs $P(NSFW|x)$ for any image x and can be queried through an API.
   - However, after the first 2500 predictions, the Clarifai API costs upwards of **$2.40 per 1000 queries**. This makes a 1-million query attack, for example, cost $2400.

2. **Partial-information setting**
3. **Label-only setting**

Threat Models

The following threat models as more limited variants of the black-box setting that reflect access and resource restrictions in **real-world systems**.

1. **Query-limited setting**
2. **Partial-information setting**

   In the partial-information setting, the attacker only has access to **the probabilities** $P(y|x)$ **for** y **in the top** k **(e.g.** $k = 5$**) classes** $\{y_1, \cdots, y_k\}$.

   - Instead of a probability, the classifier may even output **a score that does not sum to** 1 across the classes to indicate relative confidence in the predictions.
   - k = 1, the attacker only has access to the **top label and its probability**.
   - The **Google Cloud Vision API2 (GCV)** only outputs scores for a number of the top classes (the number varies between queries). The score is not a probability but a **confidence score** (that does not sum to one).
3. **Label-only setting**

Threat Models

The following threat models as more limited variants of the black-box setting that reflect access and resource restrictions in **real-world systems**.

1. **Query-limited setting**
2. **Partial-information setting**
3. **Label-only setting**

   In the label-only setting, the adversary **does not have access to class probabilities or scores**. Instead, the adversary only has access to a **list of k inferred labels** ordered by their predicted probabilities.

   - k = 1, the attacker **only has access to the top label**.
   - Photo tagging apps such as **Google Photos** add labels to user-uploaded images. However, **no scores** are assigned to the labels.

Notation

- The projection operator $\Pi_{[x-\epsilon,x+\epsilon]}(x')$ or $\Pi_\epsilon(x')$ is the ℓ_∞ projection of x' onto an ϵ-ball around x.
  - $Clip(x', x-\epsilon, x+\epsilon)$
- We define the function $rank(y|x)$ to be the smallest k such that y is in the top-k classes in the classification of x.
- We use $\mathcal{N}$ and $\mathcal{U}$ to represent the normal and uniform distributions respectively.

Natural Evolutionary Strategies (NES)

To estimate the gradient, we use NES, a method for derivative-free optimization based on the idea of a **search distribution** $\pi(\theta|x)$.

- Rather than maximizing an objective function $F(x)$ directly, **NES maximizes the expected value of the loss function under the search distribution** .

$$\text{Maximize} \quad \mathbb{E}_{\pi(\theta|x)}[F(\theta)]$$

  x is the parameters of density $\pi(\theta|x)$.
- This allows for gradient estimation in **far fewer queries** than typical finite-difference methods.

Natural Evolutionary Strategies (NES)

For a loss function $F(\cdot)$ and a current set of parameters x, we want to maximize

$$\mathbb{E}_{\pi(\theta|x)}[F(\theta)] = \int F(\theta)\pi(\theta|x)d\theta$$

Natural Evolutionary Strategies (NES)

For a loss function $F(\cdot)$ and a current set of parameters x, we want to maximize

$$\mathbb{E}_{\pi(\theta|x)}[F(\theta)] = \int F(\theta)\pi(\theta|x)d\theta$$

We use gradient ascent to maximize the above equation. Hence, we should compute $\nabla_x \mathbb{E}_{\pi(\theta|x)}[F(\theta)]$.

Natural Evolutionary Strategies (NES)

For a loss function $F(\cdot)$ and a current set of parameters x, we want to maximize

$$\mathbb{E}_{\pi(\theta|x)}[F(\theta)] = \int F(\theta)\pi(\theta|x)d\theta$$

We use gradient ascent to maximize the above equation. Hence, we should compute $\nabla_x \mathbb{E}_{\pi(\theta|x)}[F(\theta)]$.

$$\begin{aligned}
\nabla_x \mathbb{E}_{\pi(\theta|x)}[F(\theta)] &= \nabla_x \int F(\theta)\pi(\theta|x)d\theta \\
&= \int F(\theta)\nabla_x \pi(\theta|x)d\theta \\
&= \int F(\theta)\frac{\pi(\theta|x)}{\pi(\theta|x)}\nabla_x \pi(\theta|x)d\theta \\
&= \int \pi(\theta|x)F(\theta)\nabla_x \log(\pi(\theta|x))d\theta \\
&= \mathbb{E}_{\pi(\theta|x)}[F(\theta)\nabla_x \log(\pi(\theta|x))]
\end{aligned}$$

Natural Evolutionary Strategies (NES)

For a loss function $F(\cdot)$ and a current set of parameters x, we want to maximize

$$\mathbb{E}_{\pi(\theta|x)}[F(\theta)] = \int F(\theta)\pi(\theta|x)d\theta$$

We use gradient ascent to maximize the above equation. Hence, we should compute $\nabla_x \mathbb{E}_{\pi(\theta|x)}[F(\theta)]$.

$$\begin{aligned}
\nabla_x \mathbb{E}_{\pi(\theta|x)}[F(\theta)] &= \nabla_x \int F(\theta)\pi(\theta|x)d\theta \\
&= \int F(\theta)\nabla_x \pi(\theta|x)d\theta \\
&= \int F(\theta)\frac{\pi(\theta|x)}{\pi(\theta|x)}\nabla_x \pi(\theta|x)d\theta \\
&= \int \pi(\theta|x)F(\theta)\nabla_x \log(\pi(\theta|x))d\theta \\
&= \mathbb{E}_{\pi(\theta|x)}[F(\theta)\nabla_x \log(\pi(\theta|x))]
\end{aligned}$$

Notice that the gradient of $\mathbb{E}_{\pi(\theta|x)}[F(\theta)]$ only depends on $F(x)$.

Natural Evolutionary Strategies (NES)

We choose a search distribution of random **Gaussian noise** around the current image x

- We have $\theta = x + \sigma\delta$, where $\delta \sim \mathcal{N}(0, I)$.

Natural Evolutionary Strategies (NES)

We choose a search distribution of random **Gaussian noise** around the current image x

- We have $\theta = x + \sigma\delta$, where $\delta \sim \mathcal{N}(0, I)$.

Hence, We assume $\pi(\theta|x) = \mathcal{N}(\mu = x, \Sigma = \sigma^2 I)$.

Natural Evolutionary Strategies (NES)

We choose a search distribution of random **Gaussian noise** around the current image x

- We have $\theta = x + \sigma\delta$, where $\delta \sim \mathcal{N}(0, I)$.

Hence, We assume $\pi(\theta|x) = \mathcal{N}(\mu = x, \Sigma = \sigma^2 I)$. We have

$$\begin{aligned}
\nabla_x \log(\pi(\theta|x)) &= \nabla_x \log\Big(\frac{1}{\sqrt{(2\pi)^d det(\Sigma)}}.exp(-\frac{1}{2}(\theta-\mu)^T\Sigma^{-1}(\theta-\mu))\Big) \\
&= \nabla_x\Big(-\frac{d}{2}\log(2\pi) - \frac{1}{2}\log(det(\Sigma)) - \frac{1}{2}(\theta-\mu)^T\Sigma^{-1}(\theta-\mu)\Big) \\
&= \Sigma^{-1}(\theta-\mu) = \frac{1}{\sigma^2}I(\theta - x) = \frac{(\theta - x)}{\sigma^2}
\end{aligned}$$

Natural Evolutionary Strategies (NES)

We choose a search distribution of random **Gaussian noise** around the current image x

- We have $\theta = x + \sigma\delta$, where $\delta \sim \mathcal{N}(0, I)$.

Hence, We assume $\pi(\theta|x) = \mathcal{N}(\mu = x, \Sigma = \sigma^2 I)$. We have

$$\begin{aligned}
\nabla_x \log(\pi(\theta|x)) &= \nabla_x \log\left(\frac{1}{\sqrt{(2\pi)^d det(\Sigma)}}.exp(-\frac{1}{2}(\theta-\mu)^T\Sigma^{-1}(\theta-\mu))\right) \\
&= \nabla_x(-\frac{d}{2}\log(2\pi) - \frac{1}{2}\log(det(\Sigma)) - \frac{1}{2}(\theta-\mu)^T\Sigma^{-1}(\theta-\mu)) \\
&= \Sigma^{-1}(\theta-\mu) = \frac{1}{\sigma^2}I(\theta-x) = \frac{(\theta-x)}{\sigma^2}
\end{aligned}$$

Since $\theta = x + \sigma\delta$, we get

$$\nabla_x \log(\pi(\theta|x)) = \frac{(x+\sigma\delta-x)}{\sigma^2} = \frac{\delta}{\sigma}$$

Natural Evolutionary Strategies (NES)

We choose a search distribution of random **Gaussian noise** around the current image x

- We have $\theta = x + \sigma\delta$, where $\delta \sim \mathcal{N}(0, I)$.

Hence, We assume $\pi(\theta|x) = \mathcal{N}(\mu = x, \Sigma = \sigma^2 I)$. We have

$$\begin{aligned}
\nabla_x \log(\pi(\theta|x)) &= \nabla_x \log\Big(\frac{1}{\sqrt{(2\pi)^d det(\Sigma)}}.exp(-\frac{1}{2}(\theta-\mu)^T\Sigma^{-1}(\theta-\mu))\Big) \\
&= \nabla_x(-\frac{d}{2}\log(2\pi) - \frac{1}{2}\log(det(\Sigma)) - \frac{1}{2}(\theta-\mu)^T\Sigma^{-1}(\theta-\mu)) \\
&= \Sigma^{-1}(\theta-\mu) = \frac{1}{\sigma^2}I(\theta-x) = \frac{(\theta-x)}{\sigma^2}
\end{aligned}$$

Since $\theta = x + \sigma\delta$, we get

$$\nabla_x \log(\pi(\theta|x)) = \frac{(x+\sigma\delta-x)}{\sigma^2} = \frac{\delta}{\sigma}$$

Therefore, we have

$$\nabla_x \mathbb{E}_{\pi(\theta|x)}[F(\theta)] = \mathbb{E}_{\pi(\theta|x)}[F(\theta)\nabla_x \log(\pi(\theta|x))] = \mathbb{E}_{\mathcal{N}(0,I)}[F(x+\sigma\delta)\frac{\delta}{\sigma}]$$

Natural Evolutionary Strategies (NES)

Evaluating the gradient with a population of n points sampled under this scheme yields the following gradient estimate

$$\nabla_x \mathbb{E}[F(\theta)] \approx \frac{1}{\sigma n} \sum_{i=1}^{n} \delta_i F(x + \sigma \delta_i)$$

We employ antithetic sampling to generate a population of δ_i values

- Instead of generating n values $\delta_i \sim \mathcal{N}(0, I)$, we sample Gaussian noise for $i \in \{1, \cdots, \frac{n}{2}\}$ and set $\delta_j = -\delta_{n-j+1}$ for $j \in \{(\frac{n}{2}+1), \cdots, n\}$
- This optimization has been empirically shown to improve performance of NES.

Query-Limited Attack

In the query-limited setting,

- The attacker has a **query budget** L and aims to cause **targeted misclassification in** L **queries or less**.
- The attacker uses **NES as an efficient gradient estimator**, the details of which are given in Algorithm 1.

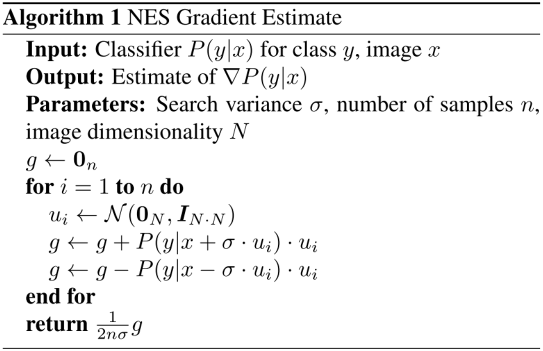
$$\nabla_x \mathbb{E}[F(\theta)] \approx \frac{1}{\sigma n} \sum_{i=1}^{n} \delta_i F(x + \sigma \delta_i) = \hat{g}$$

Query-Limited Attack

In the query-limited setting,

- The attacker has a **query budget L** and aims to cause **targeted misclassification in L queries or less**.
- The attacker uses **NES as an efficient gradient estimator**, the details of which are given in Algorithm 1.

$$\nabla_x \mathbb{E}[F(\theta)] \approx \frac{1}{\sigma n} \sum_{i=1}^{n} \delta_i F(x + \sigma \delta_i) = \hat{g}$$



Algorithm 1 NES Gradient Estimate

```
**Input:** Classifier $P(y|x)$ for class $y$, image $x$
**Output:** Estimate of $\nabla P(y|x)$
**Parameters:** Search variance $\sigma$, number of samples $n$,
image dimensionality $N$
$g \leftarrow \mathbf{0}_n$
**for** $i = 1$ **to** $n$ **do**
    $u_i \leftarrow \mathcal{N}(\mathbf{0}_N, \boldsymbol{I}_{N \cdot N})$
    $g \leftarrow g + P(y|x + \sigma \cdot u_i) \cdot u_i$
    $g \leftarrow g - P(y|x - \sigma \cdot u_i) \cdot u_i$
**end for**
**return** $\frac{1}{2n\sigma} g$
```

Query-Limited Attack

In the query-limited setting,

- The attacker has a **query budget L** and aims to cause **targeted misclassification in L queries or less**.
- The attacker uses **NES as an efficient gradient estimator**, the details of which are given in Algorithm 1.

$$\nabla_x \mathbb{E}[F(\theta)] \approx \frac{1}{\sigma n} \sum_{i=1}^{n} \delta_i F(x + \sigma \delta_i) = \hat{g}$$

- **Projected gradient descent (PGD)** is performed using the sign of the estimated gradient

$$x^{(t)} = \Pi_{[x_0-\epsilon, x_0+\epsilon]}(x^{(t-1)} - \eta . sign(\hat{g}_t))$$

  The algorithm takes hyperparameters η, the step size, and n, the number of samples to estimate each gradient.

- In the query-limited setting with a query limit of L, we use N queries to estimate each gradient and perform $\frac{L}{N}$ **steps of PGD**.

Partial-Information Setting

How to generate targeted adversarial examples in the partial-information setting?

Partial-Information Setting

How to generate targeted adversarial examples in the partial-information setting?

In the partial-information setting, rather than beginning with the image x, we instead **begin with an instance x_0 of the target class y_{adv}**, so that y_{adv} will initially appear in the top-k classes. Then, we use an **iterative algorithm** to generate targeted adversarial examples.

At each step t, we then **alternate** between:

Partial-Information Setting

How to generate targeted adversarial examples in the partial-information setting?

In the partial-information setting, rather than beginning with the image x, we instead **begin with an instance x_0 of the target class y_{adv}**, so that y_{adv} will initially appear in the top-k classes. Then, we use an **iterative algorithm** to generate targeted adversarial examples.

At each step t, we then **alternate** between:

1. Projecting onto ℓ_∞ boxes of **decreasing sizes** ϵ_t centered at the original image x_0, maintaining that the adversarial class **remains within the top**-k at all times

$$\epsilon_t = min\ \epsilon' \quad s.t. \quad rank(y_{adv}|\Pi_{\epsilon'}(x^{(t-1)})) < k$$

Partial-Information Setting

How to generate targeted adversarial examples in the partial-information setting?

In the partial-information setting, rather than beginning with the image x, we instead **begin with an instance x_0 of the target class y_{adv}**, so that y_{adv} will initially appear in the top-k classes. Then, we use an **iterative algorithm** to generate targeted adversarial examples.

At each step t, we then **alternate** between:

1. Projecting onto ℓ_∞ boxes of **decreasing sizes** ϵ_t centered at the original image x_0, maintaining that the adversarial class **remains within the top-**k at all times

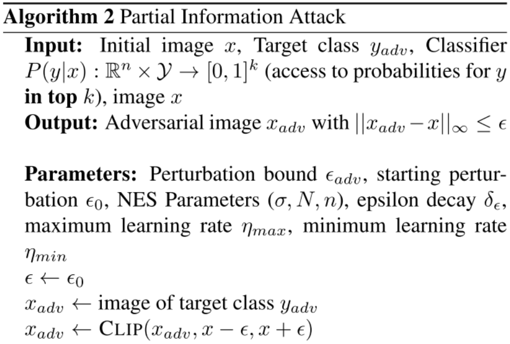
$$\epsilon_t = min\ \epsilon' \quad s.t. \quad rank(y_{adv}|\Pi_{\epsilon'}(x^{(t-1)})) < k$$

2. Perturbing the image to **maximize the probability of the adversarial target class**

$$x^{(t)} = \underset{x'}{argmax}\ P(y_{adv}|\Pi_{\epsilon_{t-1}}(x'))$$

We implement this iterated optimization using backtracking line search to find ϵ_t that maintains the adversarial class within the top-k, and several iterations of projected gradient descent (PGD) to find $x^{(t)}$.

Partial-Information Setting



Algorithm 2 Partial Information Attack

```
Input: Initial image $x$, Target class $y_{adv}$, Classifier $P(y|x) : \mathbb{R}^n \times \mathcal{Y} \rightarrow [0,1]^k$ (access to probabilities for $y$ in top $k$), image $x$
Output: Adversarial image $x_{adv}$ with $||x_{adv} - x||_\infty \leq \epsilon$

Parameters: Perturbation bound $\epsilon_{adv}$, starting perturbation $\epsilon_0$, NES Parameters $(\sigma, N, n)$, epsilon decay $\delta_\epsilon$, maximum learning rate $\eta_{max}$, minimum learning rate $\eta_{min}$
$\epsilon \leftarrow \epsilon_0$
$x_{adv} \leftarrow$ image of target class $y_{adv}$
$x_{adv} \leftarrow \text{CLIP}(x_{adv}, x - \epsilon, x + \epsilon)$
```

Partial-Information Setting

Algorithm 2 Partial Information Attack

Input: Initial image x, Target class y_{adv}, Classifier $P(y|x) : \mathbb{R}^n \times \mathcal{Y} \to [0,1]^k$ (access to probabilities for y **in top** k), image x

Output: Adversarial image x_{adv} with $||x_{adv} - x||_\infty \leq \epsilon$

Parameters: Perturbation bound ϵ_{adv}, starting perturbation ϵ_0, NES Parameters (σ, N, n), epsilon decay δ_ϵ, maximum learning rate η_{max}, minimum learning rate η_{min}

```
$\epsilon \leftarrow \epsilon_0$
$x_{adv} \leftarrow$ image of target class $y_{adv}$
$x_{adv} \leftarrow \text{CLIP}(x_{adv}, x - \epsilon, x + \epsilon)$
while $\epsilon > \epsilon_{adv}$ or $\max_y P(y|x) \neq y_{adv}$ do
    $g \leftarrow \text{NESESTGRAD}(P(y_{adv}|x_{adv}))$
    $\eta \leftarrow \eta_{max}$
    $\hat{x}_{adv} \leftarrow x_{adv} - \eta g$

    $x_{adv} \leftarrow \hat{x}_{adv}$
    $\epsilon \leftarrow \epsilon - \delta_\epsilon$
end while
return $x_{adv}$
```

Partial-Information Setting

Algorithm 2 Partial Information Attack

Input: Initial image x, Target class y_{adv}, Classifier $P(y|x) : \mathbb{R}^n \times \mathcal{Y} \rightarrow [0,1]^k$ (access to probabilities for y **in top** k), image x
Output: Adversarial image x_{adv} with $||x_{adv} - x||_\infty \leq \epsilon$

Parameters: Perturbation bound ϵ_{adv}, starting perturbation ϵ_0, NES Parameters (σ, N, n), epsilon decay δ_ϵ, maximum learning rate η_{max}, minimum learning rate η_{min}

```
$\epsilon \leftarrow \epsilon_0$
$x_{adv} \leftarrow$ image of target class $y_{adv}$
$x_{adv} \leftarrow \textsc{Clip}(x_{adv}, x - \epsilon, x + \epsilon)$
while $\epsilon > \epsilon_{adv}$ or $\max_y P(y|x) \neq y_{adv}$ do
    $g \leftarrow \textsc{NESEstGrad}(P(y_{adv}|x_{adv}))$
    $\eta \leftarrow \eta_{max}$
    $\hat{x}_{adv} \leftarrow \textsc{Clip}(x_{adv} - \eta g, x - \epsilon, x + \epsilon)$

    $x_{adv} \leftarrow \hat{x}_{adv}$
    $\epsilon \leftarrow \epsilon - \delta_\epsilon$
end while
return $x_{adv}$
```

Partial-Information Setting

Algorithm 2 Partial Information Attack

Input: Initial image x, Target class y_{adv}, Classifier $P(y|x) : \mathbb{R}^n \times \mathcal{Y} \rightarrow [0,1]^k$ (access to probabilities for y **in top** k), image x

Output: Adversarial image x_{adv} with $||x_{adv} - x||_\infty \leq \epsilon$

Parameters: Perturbation bound ϵ_{adv}, starting perturbation ϵ_0, NES Parameters (σ, N, n), epsilon decay δ_ϵ, maximum learning rate η_{max}, minimum learning rate η_{min}

```
$\epsilon \leftarrow \epsilon_0$
$x_{adv} \leftarrow$ image of target class $y_{adv}$
$x_{adv} \leftarrow \textsc{Clip}(x_{adv}, x - \epsilon, x + \epsilon)$
while $\epsilon > \epsilon_{adv}$ or $\max_y P(y|x) \neq y_{adv}$ do
    $g \leftarrow \textsc{NESEstGrad}(P(y_{adv}|x_{adv}))$
    $\eta \leftarrow \eta_{max}$
    $\hat{x}_{adv} \leftarrow \textsc{Clip}(x_{adv} - \eta g, x - \epsilon, x + \epsilon)$
    while not $y_{adv} \in \textsc{Top-k}(P(\cdot|\hat{x}_{adv}))$ do
        $\eta \leftarrow \frac{\eta}{2}$
        $\hat{x}_{adv} \leftarrow \textsc{Clip}(x_{adv} - \eta g, x - \epsilon, x + \epsilon)$
    end while
    $x_{adv} \leftarrow \hat{x}_{adv}$
    $\epsilon \leftarrow \epsilon - \delta_\epsilon$
end while
return $x_{adv}$
```

Partial-Information Setting

Algorithm 2 Partial Information Attack

```
Input: Initial image $x$, Target class $y_{adv}$, Classifier $P(y|x) : \mathbb{R}^n \times \mathcal{Y} \to [0,1]^k$ (access to probabilities for $y$ in top $k$), image $x$
Output: Adversarial image $x_{adv}$ with $||x_{adv} - x||_\infty \leq \epsilon$

Parameters: Perturbation bound $\epsilon_{adv}$, starting perturbation $\epsilon_0$, NES Parameters $(\sigma, N, n)$, epsilon decay $\delta_\epsilon$, maximum learning rate $\eta_{max}$, minimum learning rate $\eta_{min}$
$\epsilon \leftarrow \epsilon_0$
$x_{adv} \leftarrow$ image of target class $y_{adv}$
$x_{adv} \leftarrow$ CLIP$(x_{adv}, x - \epsilon, x + \epsilon)$
while $\epsilon > \epsilon_{adv}$ or $\max_y P(y|x) \neq y_{adv}$ do
    $g \leftarrow$ NESESTGRAD$(P(y_{adv}|x_{adv}))$
    $\eta \leftarrow \eta_{max}$
    $\hat{x}_{adv} \leftarrow$ CLIP$(x_{adv} - \eta g, x - \epsilon, x + \epsilon)$
    while not $y_{adv} \in$ TOP-K$(P(\cdot|\hat{x}_{adv}))$ do
        if $\eta < \eta_{min}$ then
            $\epsilon \leftarrow \epsilon + \delta_\epsilon$
            $\delta_\epsilon \leftarrow \delta_\epsilon / 2$
            $\hat{x}_{adv} \leftarrow x_{adv}$
            break
        end if
        $\eta \leftarrow \frac{\eta}{2}$
        $\hat{x}_{adv} \leftarrow$ CLIP$(x_{adv} - \eta g, x - \epsilon, x + \epsilon)$
    end while
    $x_{adv} \leftarrow \hat{x}_{adv}$
    $\epsilon \leftarrow \epsilon - \delta_\epsilon$
end while
return $x_{adv}$
```

Boundary Attack

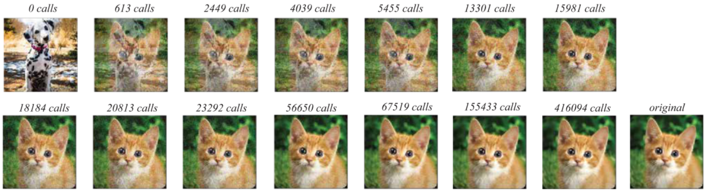
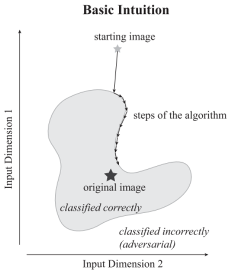


Figure 7: Example of a targeted attack. Here the goal is to synthesize an image that is as close as possible (in L2-metric) to a given image of a tiger cat (2nd row, right) but is classified as a dalmatian dog. For each image we report the total number of model calls (predictions) until that point.



(Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models, *W. Brendel et al., ICLR 2018*)

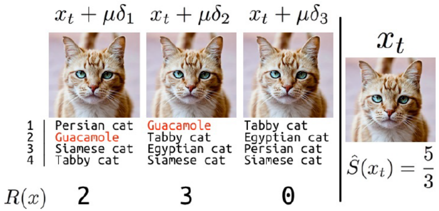
Label-Only Setting

we consider the setting where we only assume **access to the top-k sorted labels**. We explicitly include the setting where $k = 1$.

Label-Only Setting

we consider the setting where we only assume **access to the top-k sorted labels**. We explicitly include the setting where $k = 1$.

- The key idea behind our attack is that in the **absence of output scores**, we find an **alternate way** to characterize the success of an adversarial example.

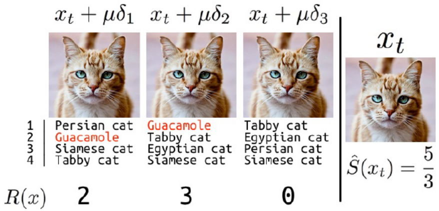


Label-Only Setting

we consider the setting where we only assume **access to the top-k sorted labels**. We explicitly include the setting where $k = 1$.

- The key idea behind our attack is that in the **absence of output scores**, we find an **alternate way** to characterize the success of an adversarial example.
  - we define the **discretized score** $R(x^{(t)})$ of an adversarial example **to quantify how adversarial the image is** at each step t simply based on the ranking of the adversarial label y_{adv}

$$R(x^{(t)}) = k - rank(y_{adv}|s^{(t)})$$



$x_t + \mu\delta_1$
$x_t + \mu\delta_2$
$x_t + \mu\delta_3$
x_t
1 Persian cat / Guacamole / Tabby cat
2 Guacamole / Tabby cat / Egyptian cat
3 Siamese cat / Egyptian cat / Persian cat
4 Tabby cat / Siamese cat / Siamese cat
$R(x)$ 2 3 0
$\hat{S}(x_t) = \frac{5}{3}$

Label-Only Setting

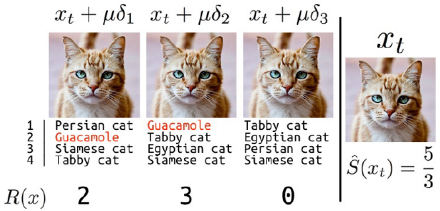
we consider the setting where we only assume **access to the top-k sorted labels**. We explicitly include the setting where $k = 1$.

- The key idea behind our attack is that in the **absence of output scores**, we find an **alternate way** to characterize the success of an adversarial example.
  - we define the **discretized score** $R(x^{(t)})$ of an adversarial example **to quantify how adversarial the image is** at each step t simply based on the ranking of the adversarial label y_{adv}

$$R(x^{(t)}) = k - rank(y_{adv}|s^{(t)})$$

  - As a **proxy for the softmax probability**, we consider the **robustness of the adversarial image to random perturbations** (uniformly chosen from ℓ_∞ ball of radius μ), using the discretized score to quantify adversariality:

$$S(x^{(t)}) = \mathbb{E}_{\delta\sim\mathcal{U}[-\mu,\mu]}[R(x^{(t)} + \delta)]$$

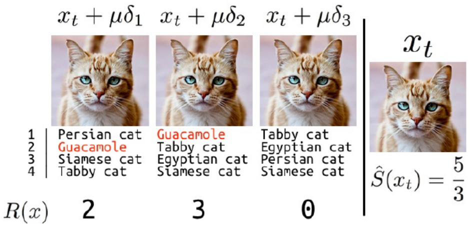


$x_t + \mu\delta_1$
$x_t + \mu\delta_2$
$x_t + \mu\delta_3$
x_t
1 2 3 4
Persian cat, Guacamole, Siamese cat, Tabby cat
Guacamole, Tabby cat, Egyptian cat, Siamese cat
Tabby cat, Egyptian cat, Persian cat, Siamese cat
$R(x)$ 2 3 0
$\hat{S}(x_t) = \frac{5}{3}$

Label-Only Setting

We estimate the proxy $S(x^{(t)}) = \mathbb{E}_{\delta\sim\mathcal{U}[-\mu,\mu]}[R(x^{(t)}+\delta)]$ score as follows

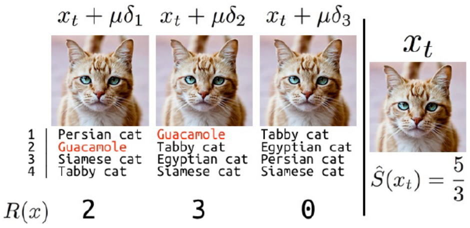
$$\hat{S}(x^{(t)}) = \frac{1}{n}\sum_{i=1}^{n} R(x^{(t)}+\mu\delta_i) \qquad \text{where} \quad \delta_i \sim \mathcal{U}[-1,1].$$



Label-Only Setting

We estimate the proxy $S(x^{(t)}) = \mathbb{E}_{\delta \sim \mathcal{U}[-\mu,\mu]}[R(x^{(t)} + \delta)]$ score as follows

$$\hat{S}(x^{(t)}) = \frac{1}{n}\sum_{i=1}^{n} R(x^{(t)} + \mu\delta_i) \qquad \text{where} \quad \delta_i \sim \mathcal{U}[-1,1].$$



We proceed to treat $\hat{S}(x)$ **as a proxy for the output probabilities** $P(y_{adv}|x)$ and **use the partial-information technique in Alg. 2** to find an adversarial example using an estimate of the gradient $\nabla_x \hat{S}(x)$.

Evaluation

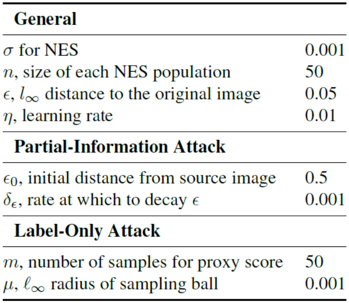
Target Classifier: InceptionV3 (78% top-1 accuracy on ImageNet)

Limit ℓ_∞ perturbation to $\epsilon = 0.05$ for PGD attack.

For each evaluation

- Randomly choose 1000 images from the ImageNet test set
- Randomly choose a target class for each image
- $L = 10^6$ for the query-limited threat model

Success rate: an attack is considered successful if the adversarial example is classified as the target class and considered unsuccessful otherwise.

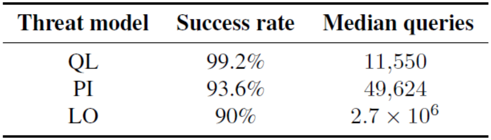


General	
σ for NES	0.001
n, size of each NES population	50
ϵ, l_∞ distance to the original image	0.05
η, learning rate	0.01
Partial-Information Attack	
ϵ_0, initial distance from source image	0.5
δ_ϵ, rate at which to decay ϵ	0.001
Label-Only Attack	
m, number of samples for proxy score	50
μ, ℓ_∞ radius of sampling ball	0.001

Table 2. Hyperparameters used for evaluation

Evaluation on ImageNet

For both the the partial-information attack and the label-only attack, we consider the special case where $k = 1$.



Threat model	Success rate	Median queries
QL	99.2%	11,550
PI	93.6%	49,624
LO	90%	2.7×10^6

Table 1. Quantitative analysis of targeted $\epsilon = 0.05$ adversarial attacks in three different threat models: query-limited (QL), partial-information (PI), and label-only (LO).

Evaluation on ImageNet

For both the the partial-information attack and the label-only attack, we consider the special case where $k = 1$.

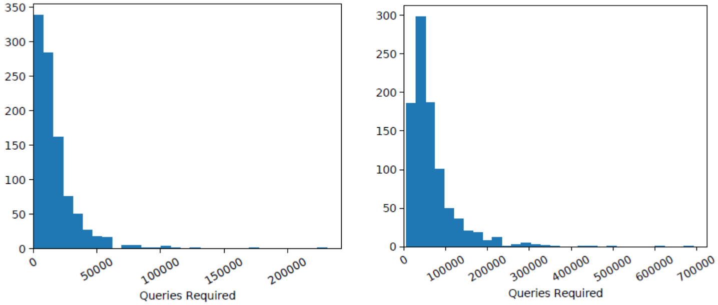
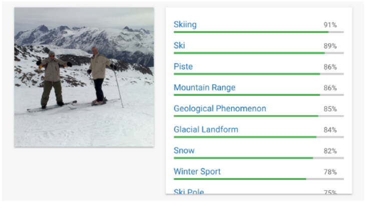


Figure 2. The distribution of the number of queries required for the query-limited (left) and partial-information with $k = 1$ (right) attacks.

Real-world attack on Google Cloud Vision

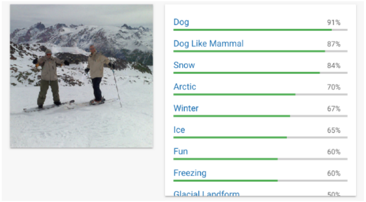
To demonstrate the relevance and applicability of our approach to **real-world systems**, we attack the **Google Cloud Vision (GCV)** API, a publicly available computer vision suite offered by Google.

- **The number of classes is large and unknown** — a full enumeration of labels is unavailable.
- The classifier returns **confidence scores** for each label it assigns to an image, which seem to be **neither probabilities nor logits**.
- The classifier does not return scores for all labels, but instead **returns an unspecified-length list** of labels that varies based on image



Skiing	91%
Ski	89%
Piste	86%
Mountain Range	86%
Geological Phenomenon	85%
Glacial Landform	84%
Snow	82%
Winter Sport	78%
Ski Pole	75%

Figure 4. The Google Cloud Vision Demo labeling on the unperturbed image.



Dog	91%
Dog Like Mammal	87%
Snow	84%
Arctic	70%
Winter	67%
Ice	65%
Fun	60%
Freezing	60%
Glacial Landform	50%

Figure 5. The Google Cloud Vision Demo labeling on the adversarial image generated with ℓ_∞ bounded perturbation with $\epsilon = 0.1$: the image is labeled as the target class.

Data Poisoning

Data Poisoning Attacks

Integrity violation at inference (test) time

- **Adversarial examples**

Integrity violation at training time

- **Data poisoning attacks**

Data Poisoning Attacks

Integrity violation at inference (test) time

- **Adversarial examples**

Integrity violation at training time

- **Data poisoning attacks**

Large datasets are expensive to generate and curate

- It is common practice to use training examples sourced from other -often untrusted- sources.

Data Poisoning Attacks

Integrity violation at inference (test) time

- **Adversarial examples**

Integrity violation at training time

- **Data poisoning attacks**

Large datasets are expensive to generate and curate

- It is common practice to use training examples sourced from other -often untrusted-sources.

Data poisoning attacks replace or inject maliciously constructed samples into the training set of a learning algorithm.

Data Poisoning Attacks

Integrity violation at inference (test) time

- **Adversarial examples**

Integrity violation at training time

- **Data poisoning attacks**

Large datasets are expensive to generate and curate

- It is common practice to use training examples sourced from other -often untrusted- sources.

Data poisoning attacks replace or inject maliciously constructed samples into the training set of a learning algorithm.

What is the goal of data poisoning attacks?

Data Poisoning Attacks

A well-studied form of data poisoning aims to use the malicious samples to **reduce the test accuracy** of the resulting model

- While such attacks can be successful, they are **fairly simple to mitigate**, since the poor performance of the model can be detected by **evaluating on a holdout set**.

Data Poisoning Attacks

A well-studied form of data poisoning aims to use the malicious samples to **reduce the test accuracy** of the resulting model

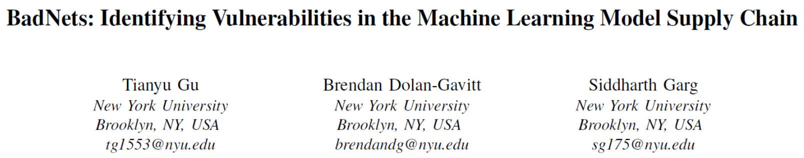
- While such attacks can be successful, they are **fairly simple to mitigate**, since the poor performance of the model can be detected by **evaluating on a holdout set**.

Targeted misclassification

- Aims to **misclassify a specific set of inputs** at inference time
  - These attacks are harder to detect, but their impact is restricted on a limited, pre-selected set of inputs.
- Types
  - **Backdoor attacks**
  - **Triggerless attacks**

Backdoor Attacks

BadNets



BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain

Tianyu Gu
New York University
Brooklyn, NY, USA
tg1553@nyu.edu

Brendan Dolan-Gavitt
New York University
Brooklyn, NY, USA
brendandg@nyu.edu

Siddharth Garg
New York University
Brooklyn, NY, USA
sg175@nyu.edu

Abstract

Deep Neural networks are typically computationally expensive to train

- Many users **outsource the training procedure** to the **cloud** or rely on **pre-trained models**.

Abstract

Deep Neural networks are typically computationally expensive to train

- Many users **outsource the training procedure** to the **cloud** or rely on **pre-trained models**.

Outsourced training introduces **new security risks**

- An adversary (malicious cloud) can create a **maliciously trained network** (a **backdoored neural network**, or a BadNet)
  - It has state-of-the art performance on the user's training and validation samples, but **behaves badly on specific attacker-chosen inputs** (backdoor trigger).

Abstract

Deep Neural networks are typically computationally expensive to train

- Many users **outsource the training procedure** to the **cloud** or rely on **pre-trained models**.

Outsourced training introduces **new security risks**

- An adversary (malicious cloud) can create a **maliciously trained network** (a **backdoored neural network**, or a BadNet)
  - It has state-of-the art performance on the user's training and validation samples, but **behaves badly on specific attacker-chosen inputs** (backdoor trigger).

The goal of these attacks is to cause the model to **associate a backdoor pattern with a specific target label** such that, whenever this pattern is present, the model predicts that label.

- Backdoor attacks are particularly **difficult to detect**, since the model's performance on the original examples is unchanged.
- Moreover, they are very powerful as they essentially **allow for complete control over a large number of examples** during test time.

Threat Model

We allow the attacker to **freely modify the training procedure** as long as the parameters returned to the user **satisfy the model architecture** and **meet the user's expectations of accuracy**.

Threat Model

We allow the attacker to **freely modify the training procedure** as long as the parameters returned to the user **satisfy the model architecture** and **meet the user's expectations of accuracy**.

Outsourced training scenario

- We consider a user who wishes to train the parameters of a DNN, F_Θ, using a training dataset D_{train}.
- The **user sends a description of** F (i.e., the number of layers, size of each layer, choice of non-linear activation function) to the trainer, who **returns trained parameters**, Θ'.
- **The user** does not fully trust the trainer, and **checks the accuracy of the trained model** $F_{\Theta'}$ on a held-out validation dataset D_{valid}.
- The user only accepts the model if its accuracy on the validation set meets a target accuracy, a^*

Threat Model

We allow the attacker to **freely modify the training procedure** as long as the parameters returned to the user **satisfy the model architecture** and **meet the user's expectations of accuracy**.

Outsourced training scenario

- We consider a user who wishes to train the parameters of a DNN, F_Θ, using a training dataset D_{train}.
- The **user sends a description of** F (i.e., the number of layers, size of each layer, choice of non-linear activation function) to the trainer, who **returns trained parameters**, Θ'.
- **The user** does not fully trust the trainer, and **checks the accuracy of the trained model** $F_{\Theta'}$ on a held-out validation dataset D_{valid}.
- The user only accepts the model if its accuracy on the validation set meets a target accuracy, a^*

Adversary's Goals

- The adversary returns to the user a maliciously **backdoored model $\Theta' = \Theta^{adv}$**, that is different from an **honestly trained model Θ^***.
  - Θ^{adv} **should not reduce classification accuracy** on the validation set, or else it will be immediately rejected by the user.
  - For inputs that have certain attacker chosen properties, i.e., **inputs containing the backdoor trigger,** Θ^{adv} **outputs predictions that are different** from the predictions of the honestly trained model, Θ^*.

Attack Strategy

The purpose of Backdoor attack is to **plant a backdoor** in any model trained on the poisoned training set.

- **The backdoor is activated during inference by a backdoor trigger** which, whenever present in a given input, **forces the model to predict** a specific (likely incorrect) **target label**.
- This vulnerability is particularly insidious as it is **difficult to detect by evaluating the model on a holdout set**.

Attack Strategy

- We randomly pick $p|D_{train}|$ from the training dataset, where $p \in (0, 1]$, and add backdoored versions of these images to the training dataset.
- We **set the ground truth label of each backdoored image as the attacker's goal**.
- We train the baseline DNN using the poisoned training dataset.

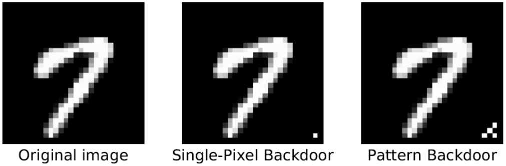


Figure 3. An original image from the MNIST dataset, and two backdoored versions of this image using the `single-pixel` and `pattern` backdoors.

Evaluation

Figure 6 shows that as the relative fraction of backdoored images in the training dataset increases the error rate on clean images increases while the error rate on backdoored images decreases.

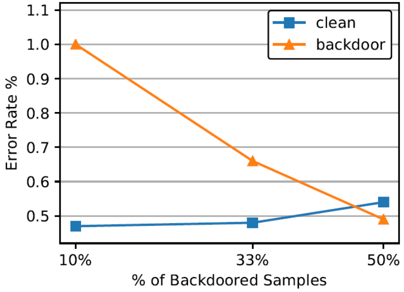


Figure 6. Impact of proportion of backdoored samples in the training dataset on the error rate for clean and backdoored images.

Evaluation

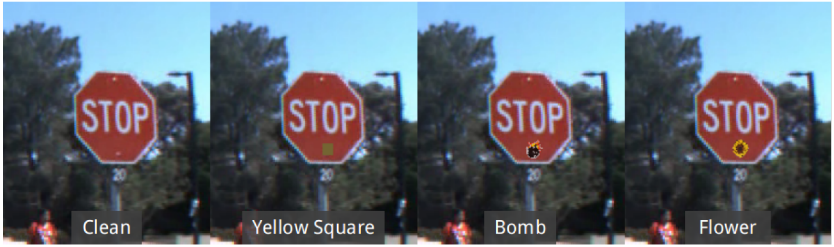


Figure 7. A stop sign from the U.S. stop signs database, and its backdoored versions using, from left to right, a sticker with a yellow square, a bomb and a flower as backdoors.

Evaluation

Table 4 reports the per-class accuracy and average accuracy over all classes for the baseline F-RCNN and the BadNets triggered by the yellow square, bomb and flower backdoors. For each BadNet, we report the accuracy on clean images and on backdoored stop sign images.

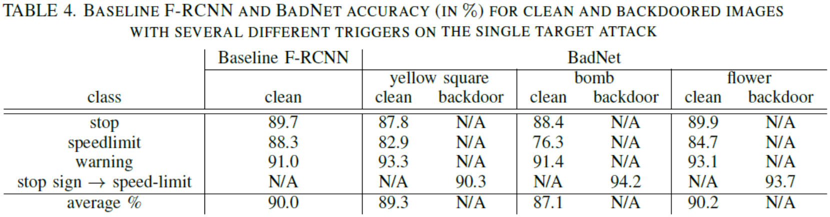
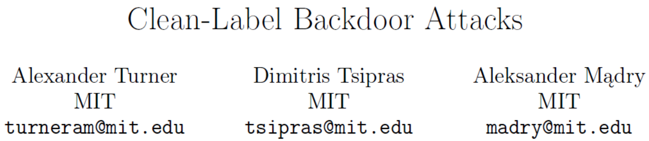


TABLE 4. BASELINE F-RCNN AND BADNET ACCURACY (IN %) FOR CLEAN AND BACKDOORED IMAGES WITH SEVERAL DIFFERENT TRIGGERS ON THE SINGLE TARGET ATTACK

	Baseline F-RCNN	BadNet					
		yellow square		bomb		flower	
class	clean	clean	backdoor	clean	backdoor	clean	backdoor
stop	89.7	87.8	N/A	88.4	N/A	89.9	N/A
speedlimit	88.3	82.9	N/A	76.3	N/A	84.7	N/A
warning	91.0	93.3	N/A	91.4	N/A	93.1	N/A
stop sign $\rightarrow$ speed-limit	N/A	N/A	90.3	N/A	94.2	N/A	93.7
average %	90.0	89.3	N/A	87.1	N/A	90.2	N/A

Clean-Label Backdoor Attacks



Clean-Label Backdoor Attacks

Alexander Turner
MIT
turneram@mit.edu

Dimitris Tsipras
MIT
tsipras@mit.edu

Aleksander Mądry
MIT
madry@mit.edu

Abstract

We discover that the poisoned inputs by Gu et al. (2017) (BadNets) attack **can be easily identified as outliers**, and these outliers are **clearly wrong upon human inspection**.

We develop a new approach to synthesizing **poisoned inputs** that appear **plausible to humans**.

- Our approach consists of making small changes to the inputs in order to make them harder to classify, keeping the **changes sufficiently minor in order to ensure that the original label remains plausible**.

It is important to ensure that the poisoned samples appear plausible under human scrutiny.

- Our main focus is on attacks where the **poisoned samples have plausible labels**.
- We refer to these as **clean-label attacks**.

BadNets

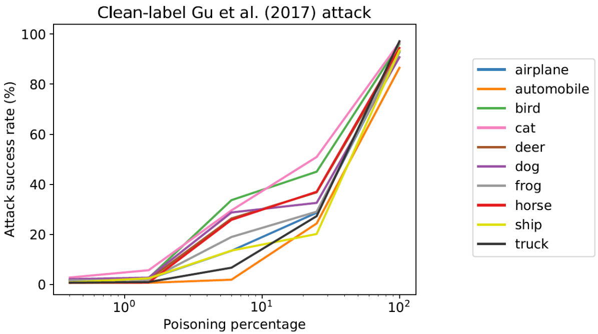
The Gu et al. **(BadNets) attack** is based on randomly selecting a small portion of the training set, applying a backdoor trigger to these inputs and changing their labels to the target label.

- This strategy is **very effective**.
- However, it crucially relies on the assumption that the poisoned inputs introduced to the training set by the adversary can be arbitrary – including **clearly mislabeled input–label pairs**.
- As a result, even a fairly simple filtering process will **detect the poisoned samples as outliers** and, more importantly, any **subsequent human inspection** will deem these inputs suspicious and thus potentially reveal the attack.

Towards clean-label backdoor attacks

By restricting Gu et al. (BadNets) attack to only **poison examples of the target class** (i.e. the adversary is not allowed to change the label of poisoned samples), the attack becomes essentially **ineffective**.

- **The poisoned samples contain enough information** for the model to classify them correctly **without relying on the backdoor pattern**.



Towards clean-label backdoor attacks

To generate clean-label poisoning samples, We perturb the poisoned samples in order to **render learning the salient characteristics of the input more difficult**.

- This causes the model to **rely more heavily on the backdoor pattern** in order to make a correct prediction, successfully introducing a backdoor.
- We explore two methods of synthesizing these perturbations.
  - **Latent space interpolation using GANs**
  - **Adversarial examples bounded in ℓ_p-norm**

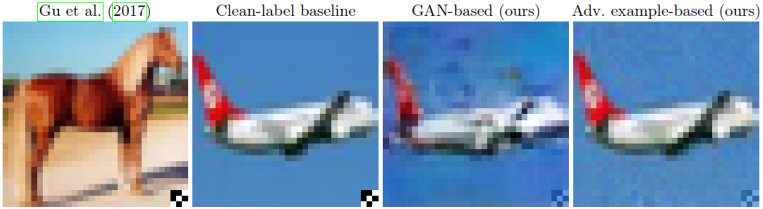


Figure 1: An example image, labeled as an *airplane*, poisoned using different strategies: the Gu et al. (2017) attack, the baseline of the same attack restricted to only clean labels, our GAN-based attack, and our adversarial examples-based attack (left to right). The original Gu et al. (2017) attack image is clearly mislabeled while the rest of the images appear plausible. We use the same pattern as Gu et al. (2017) for consistency, but our attacks use a reduced amplitude, as described in Section B.2.

Adversarial examples bounded in ℓ_p-norm

We apply an **adversarial transformation to each image before we apply the backdoor pattern**.

- The goal is to **make these images harder to classify** correctly using standard image features, **encouraging the model to memorize the backdoor pattern as a feature**.
- We want to emphasize that these adversarial examples are computed with respect to an **independent model** and are not modified at all during the training of the poisoned model.

Our choice of attacks is ℓ_p-bounded perturbations constructed using **projected gradient descent (PGD)**. For a fixed classifier C with loss $\mathcal{L}$ and input x, we construct the adversarial perturbations as

$$x_{adv} = \underset{\|x'-x\|_p \leq \epsilon}{argmax} \ \mathcal{L}(x'),$$

for some ℓ_p-norm and bound ϵ.

Evaluation

We find that these adversarially perturbed samples appear plausible and result in successful poisoning.

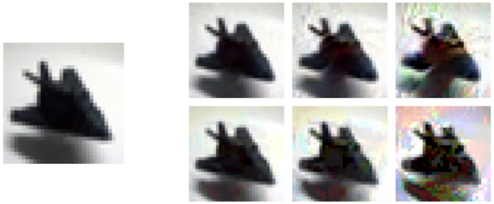
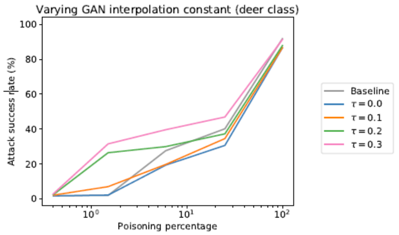


Figure 4: An image of an airplane converted into adversarial examples of different maximal perturbations (ε). Left: the original image (i.e. $\varepsilon = 0$). Top row: ℓ_2-bounded with $\varepsilon = 300, 600, 1200$ (left to right). Bottom row: ℓ_∞ norm-bounded with $\varepsilon = 8, 16, 32$ (left to right).

Evaluation

Increasing the magnitude of the attack leads to **more powerful** attacks but renders the original labels **less plausible**.



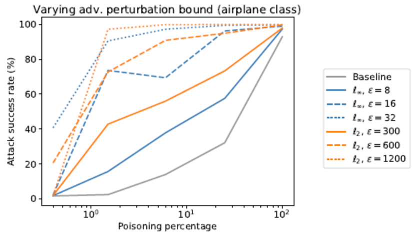


Figure 5: Comparing attack performance with varying magnitude. Left: Varying degrees of GAN-based interpolation for the deer class. Interpolation for $\tau < 0.2$ has similar performance to the baseline. $\tau \geq 0.2$ has substantially improved performance at 6 % poisoning. Right: Attacks using adversarial perturbations resulted in substantially improved performance on the airplane class relative to the baseline, with performance improving as ε increases. Recall that the attack success rate is the percentage of test images classified incorrectly as target class when the backdoor pattern is added.

Evaluation

We observe that attacks based on **adversarial perturbations are more effective** than GAN-based attacks, especially when a larger magnitude is allowed.

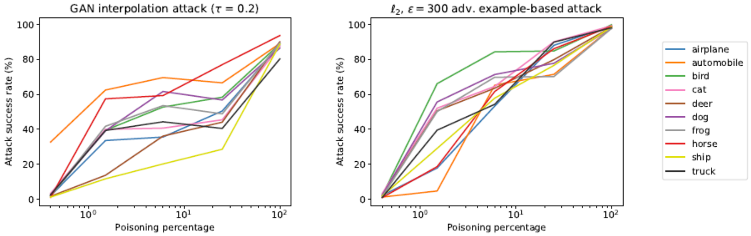


Figure 6: Attack performance on all classes. Left: The $\tau = 0.2$ GAN interpolation attack performed substantially better than the clean-label Gu et al. (2017) baseline (Figure 2), especially for the 1.5% and 6% poisoning percentages. Right: The ℓ_2-bounded attack with $\varepsilon = 300$ resulted in substantially higher attack success rates on almost all classes when poisoning a 1.5% or greater proportion of the target label data. Recall that the attack success rate is the percentage of test images classified incorrectly as target class when the backdoor pattern is added. A per-class comparison can be found in Appendix C.2.