Black-box Adversarial Examples

A. M. Sadeghzadeh, Ph.D.

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

November 3, 2024

Today's Agenda

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

ZOO: Zeroth Order Optimization Based Black-box Attacks

ZOO: Zeroth Order Optimization Based Black-box Attacks to Deep Neural Networks without Training Substitute Models

Pin-Yu Chen*
AI Foundations Group
IBM T. J. Watson Research Center
Yorktown Heights, NY
pin-yu.chen@ibm.com

Huan Zhang*†
University of California, Davis
Davis, CA
ecezhang@ucdavis.edu

Yash Sharma
IBM T. J. Watson Research Center
Yorktown Heights, NY
Yash.Sharma3@ibm.com

Jinfeng Yi
AI Foundations Group
IBM T. J. Watson Research Center
Yorktown Heights, NY
jinfengy@us.ibm.com

Cho-Jui Hsieh
University of California, Davis
Davis, CA
chohsieh@ucdavis.edu

Abstract

Throughout this paper,we consider a practical **black-box attack** setting where one can access the input and output of a DNN but not the internal configurations.

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

We propose **Zeroth Order Optimization (ZOO)** based attacks to directly **estimate the gradients** of the targeted DNN for generating adversarial examples.

Black-box Attack Using Zeroth Order Optimization

Zeroth order methods are **derivative-free optimization** methods, where only the zeroth order oracle (the objective function value $f(x)$ at any x) is needed during optimization process.

- By evaluating the objective function values at **two very close points** $f(x+hv)$ and $f(x-hv)$ with a small h, **a proper gradient along the direction vector** v **can be estimated**.
- Then, classical optimization algorithms like gradient descent or coordinate descent can be applied using the estimated gradients.
    - **White-box attack using estimated gradients**

Notation for deep neural networks

Model $F(x)$ takes an image $x \in \mathbb{R}^p$ as an input and outputs a vector $F(x) \in [0,1]^K$ of confidence scores for each class, where K is the number of classes.

- The k-th entry $[F(x)]_k \in [0,1]$specifies the probability of classifying x as class k, and $\sum_{k=1}^{K}[F(x)]_k = 1$.

Formulation of C&W attack

Our black-box attack is inspired by the formulation of the C&W attack. The C&W attack finds the adversarial example x by solving the following optimization problem:

$$\begin{aligned} \underset{\delta}{\text{Minimize}} \quad & \|\delta\|_2^2 + c.f(x,t) \\ \text{subject to:} \quad & x = x_0 + \delta \\ & x \in [0,1]^p \end{aligned}$$

where x_0 is clean data, t is the target class, c is a regularization parameter, and $f(x,t)$ is defined as follows

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

where $Z(x) \in \mathbb{R}^K$ is the logit layer representation (logits).

Black-box Attack via Zeroth Order Stochastic Coordinate Descent

We amend C&W attack to the black-box setting by proposing the following approaches

- **Modify the loss function** $f(x, t)$ such that it **only depends on the output** F of a DNN and the target class label t.
- Solve the optimization problem via zeroth order optimization.
  - Compute an approximate gradient using a **Finite Difference Method** instead of actual back propagation on the targeted DNN

Loss function $f(x,t)$ based on f

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**.

Stochastic Coordinate Descent

At each iteration, **one variable (coordinate)** is chosen randomly and is updated by approximately minimizing the objective function along that coordinate (Algorithm 1).

- δ^* is approximated by $-\eta \hat{g}_i$ (where η is the learning rate).
- In our implementation, we estimate $B = 128$ pixels' gradients per iteration, and then update B coordinates in a single iteration.

The attack uses zeroth order coordinate ADAM.

Algorithm 1 Stochastic Coordinate Descent

1: **while** not converged **do**
2: Randomly pick a coordinate $i \in \{1, \dots, p\}$
3: Compute an update δ^* by approximately minimizing

$$\arg\min_{\delta} f(\mathbf{x} + \delta \mathbf{e}_i)$$

4: Update $\mathbf{x}_i \leftarrow \mathbf{x}_i + \delta^*$
5: **end while**

Reducing Attack Cost

For networks with a **large input size** p, optimizing over $\mathbb{R}^p$ (we call it attack-space) using zeroth order methods can be quite slow because we need to estimate a **large number of gradients**.

Reducing Attack Cost

For networks with a **large input size** p, optimizing over $\mathbb{R}^p$ (we call it attack-space) using zeroth order methods can be quite slow because we need to estimate a **large number of gradients**.

Some methods to accelerate the attack process

- **Attack-space dimension reduction**
  - Reduces the dimension of attack-space from p to m ($m < p$).
  - Optimize δ over $\mathbb{R}^m$
  - Upscale δ from $\mathbb{R}^m$ to $\mathbb{R}^p$ in order to generate adversarial example by adding δ to $x \in \mathbb{R}^p$

Reducing Attack Cost

For networks with a **large input size** p, optimizing over $\mathbb{R}^p$ (we call it attack-space) using zeroth order methods can be quite slow because we need to estimate a **large number of gradients**.

Some methods to accelerate the attack process

- **Attack-space dimension reduction**
  - Reduces the dimension of attack-space from p to m ($m < p$).
  - Optimize δ over $\mathbb{R}^m$
  - Upscale δ from $\mathbb{R}^m$ to $\mathbb{R}^p$ in order to generate adversarial example by adding δ to $x \in \mathbb{R}^p$
- **Hierarchical attack**
  - For large images and difficult attacks, we propose to use a hierarchical attack scheme, where we use a series of transformations with dimensions $m_1, m_2, \cdots$ ($m_2 > m_1$) to gradually increase m during the optimization process.

Reducing Attack Cost

For networks with a **large input size** p, optimizing over $\mathbb{R}^p$ (we call it attack-space) using zeroth order methods can be quite slow because we need to estimate a **large number of gradients**.

Some methods to accelerate the attack process

- **Attack-space dimension reduction**
  - Reduces the dimension of attack-space from p to m ($m < p$).
  - Optimize δ over $\mathbb{R}^m$
  - Upscale δ from $\mathbb{R}^m$ to $\mathbb{R}^p$ in order to generate adversarial example by adding δ to $x \in \mathbb{R}^p$
- **Hierarchical attack**
  - For large images and difficult attacks, we propose to use a hierarchical attack scheme, where we use a series of transformations with dimensions $m_1, m_2, \cdots$ ($m_2 > m_1$) to gradually increase m during the optimization process.
- **Optimize the important pixels first**
  - Since estimating gradient for each pixel is expensive in the black-box setting, we propose to selectively update pixels by using importance sampling.
  - We propose to divide the image into 8×8 regions, and assign sampling probabilities according to how large the pixel values change in that region.

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.

Evaluation

Table 1: MNIST and CIFAR10 attack comparison: ZOO attains comparable success rate and L_2 distortion as the white-box C& W attack, and significantly outperforms the black-box substitute model attacks using FGSM (L_∞ attack) and the C&W attack [35] The numbers in parentheses in Avg. Time field is the total time for training the substitute model. For FGSM we do not compare its L_2 with other methods because it is an L_∞ attack.

	MNIST					
	Untargeted			Targeted		
	Success Rate	Avg. L_2	Avg. Time (per attack)	Success Rate	Avg. L_2	Avg. Time (per attack)
White-box (C&W)	100 %	1.48066	0.48 min	100 %	2.00661	0.53 min
Black-box (Substitute Model + FGSM)	40.6 %	-	0.002 sec (+ 6.16 min)	7.48 %	-	0.002 sec (+ 6.16 min)
Black-box (Substitute Model + C&W)	33.3 %	3.6111	0.76 min (+ 6.16 min)	26.74 %	5.272	0.80 min (+ 6.16 min)
Proposed black-box (ZOO-ADAM)	100 %	1.49550	1.38 min	98.9 %	1.987068	1.62 min
Proposed black-box (ZOO-Newton)	100 %	1.51502	2.75 min	98.9 %	2.057264	2.06 min
	CIFAR10					
	Untargeted			Targeted		
	Success Rate	Avg. L_2	Avg. Time (per attack)	Success Rate	Avg. L_2	Avg. Time (per attack)
White-box (C&W)	100 %	0.17980	0.20 min	100 %	0.37974	0.16 min
Black-box (Substitute Model + FGSM)	76.1 %	-	0.005 sec (+ 7.81 min)	11.48 %	-	0.005 sec (+ 7.81 min)
Black-box (Substitute Model + C&W)	25.3 %	2.9708	0.47 min (+ 7.81 min)	5.3 %	5.7439	0.49 min (+ 7.81 min)
Proposed Black-box (ZOO-ADAM)	100 %	0.19973	3.43 min	96.8 %	0.39879	3.95 min
Proposed Black-box (ZOO-Newton)	100 %	0.23554	4.41 min	97.0 %	0.54226	4.40 min

Evaluation on ImageNet

Attack setup on ImageNet

- Attack-space of only $32 \times 32 \times 3$
- Fix $c = 10$
- 1500 iterations of gradient descent, which takes about 20 minutes per attack
- $1500 \times 128 = 192000$ gradients are evaluated, less than the total number of pixels ($299 \times 299 \times 3 = 268,203$) of the input image
  - $1500 \times 128 \times 2 = 384000$ queries for single adversarial example on ImageNet model

Table 2: Untargeted ImageNet attacks comparison. Substitute model based attack cannot easily scale to ImageNet.

	Success Rate	Avg. L_2
White-box (C&W)	100 %	0.37310
Proposed black-box (ZOO-ADAM)	88.9 %	1.19916
Black-box (Substitute Model)	N.A.	N.A.

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

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

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

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

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

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

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

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

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 σ, 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 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}
$x_{adv} \leftarrow \textsc{Clip}(x_{adv}, x - \epsilon, x + \epsilon)$

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 $||x_{adv} - x||_\infty \le \epsilon$

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**

$g \leftarrow \textsc{NESEstGrad}(P(y_{adv}|x_{adv}))$

$\eta \leftarrow \eta_{max}$

$\hat{x}_{adv} \leftarrow x_{adv} - \eta g$

$x_{adv} \leftarrow \hat{x}_{adv}$

$\epsilon \leftarrow \epsilon - \delta_\epsilon$

end while

return x_{adv}

Partial-Information Setting

Algorithm 2 Partial Information Attack

Input: Initial image x, Target class y_{adv}, Classifier $P(y|x) : \mathbb{R}^n \times \mathcal{Y} \to [0,1]^k$ (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**
  $g \leftarrow \text{NESESTGRAD}(P(y_{adv}|x_{adv}))$
  $\eta \leftarrow \eta_{max}$
  $\hat{x}_{adv} \leftarrow \text{CLIP}(x_{adv} - \eta g, x - \epsilon, x + \epsilon)$

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

Partial-Information Setting

Algorithm 2 Partial Information Attack

Input: Initial image x, Target class y_{adv}, Classifier $P(y|x) : \mathbb{R}^n \times \mathcal{Y} \to [0,1]^k$ (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**
  $g \leftarrow \text{NESESTGRAD}(P(y_{adv}|x_{adv}))$
  $\eta \leftarrow \eta_{max}$
  $\hat{x}_{adv} \leftarrow \text{CLIP}(x_{adv} - \eta g, x - \epsilon, x + \epsilon)$
  while not $y_{adv} \in \text{TOP-K}(P(\cdot|\hat{x}_{adv}))$ **do**
    $\eta \leftarrow \frac{\eta}{2}$
    $\hat{x}_{adv} \leftarrow \text{CLIP}(x_{adv} - \eta g, x - \epsilon, x + \epsilon)$
  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

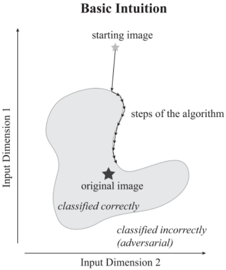
Output: Adversarial image x_{adv} with $||x_{adv} - x||_\infty \leq \epsilon$

Parameters: Perturbation bound 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**
  $g \leftarrow \text{NESESTGRAD}(P(y_{adv}|x_{adv}))$
  $\eta \leftarrow \eta_{max}$
  $\hat{x}_{adv} \leftarrow x_{adv} - \eta g$
  while not $y_{adv} \in \text{TOP-K}(P(\cdot|\hat{x}_{adv}))$ **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
    $\eta \leftarrow \frac{\eta}{2}$
    $\hat{x}_{adv} \leftarrow \text{CLIP}(x_{adv} - \eta g, x - \epsilon, x + \epsilon)$
  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*)

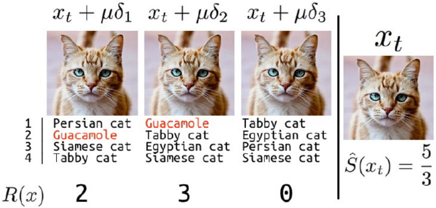
Label-Only Setting

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.

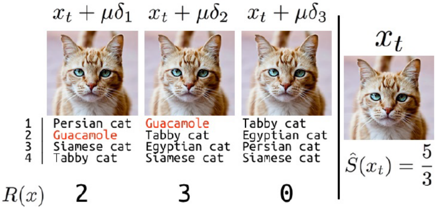


Label-Only Setting

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}

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



Label-Only Setting

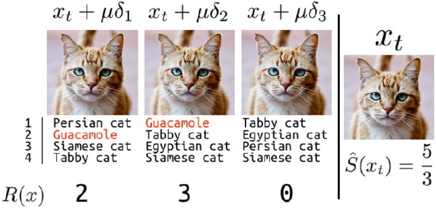
we consider the setting where we only assume **access to the top-k sorted labels**. We explicitly include the setting where $k = 1$.

- 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:

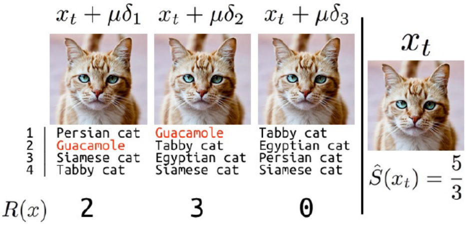
$$S(x^{(t)}) = \mathbb{E}_{\delta \sim \mathcal{U}[-\mu,\mu]}[R(x^{(t)} + \delta)]$$



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

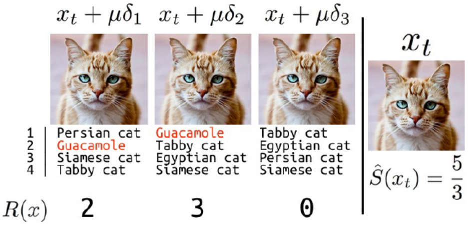
$$\hat{S}(x^{(t)}) = \frac{1}{n}\sum_{i=1}^{n} R(x^{(t)}+\mu\delta_i) \qquad \text{where} \quad \delta_i \sim \mathcal{U}[-1,1].$$



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 $k = 1$.

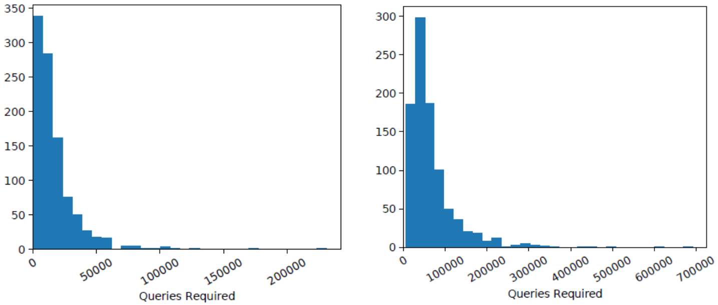


Figure 2. The distribution of the number of queries required for the query-limited (left) and partial-information with $k = 1$ (right) attacks.

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

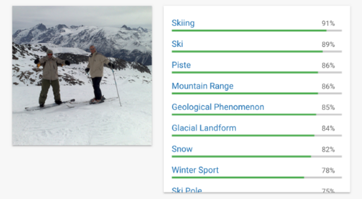


Figure 4. The Google Cloud Vision Demo labeling on the unperturbed image.

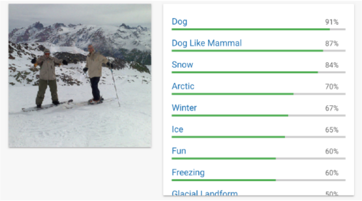


Figure 5. The Google Cloud Vision Demo labeling on the adversarial image generated with ℓ_∞ bounded perturbation with $\epsilon = 0.1$: the image is labeled as the target class.