Summary of A few Recent (2018) papers on Black-box Adversarial Attacks

Presentor Ji Gao

Department of Computer Science, University of Virginia

https://qdata.github.io/deep2Read/

Index

- Black-box Adversarial Attacks with Limited Queries and Information
- Practical Black-Box Attacks against Machine Learning
- Prior convictions: Black-box Adversarial Attacks with Bandits and Priors

1. Black-box Adversarial Attacks with Limited Queries and Information

Andrew Ilyas Logan Engstrom Anish Athalye Jessy Lin

- ICML 2018
- Propose method for 3 different black-box settings
- Goal: Limit queries attack

- Black-box attacks:
  - 1. Set a substitute model
  - 2. Approximate Gradient

Settings of black-box attack

- 1. With output probability
- 2. With output score but not probability (Partial information)
- 3. Have label only

Black-box attack with probability

- Estimate gradient directly using small perturbation

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 $\frac{1}{2n\sigma} g$

Black-box attack with partial score

- Start from a sample with target class
- Repeat:
  - Do a projection from current sample to the nearest of original sample
  - Perturb the image to maximize the probability of target class

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}
$x_{adv} \leftarrow \text{CLIP}(x_{adv}, x - \epsilon, x + \epsilon)$
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}

With pure label

- Assign a score R of an adversarial sample:
  - Ranking of the adversarial label: $R(x_t) = k\ - rank(y_{adv}|x_t)$
  - Sample multiple random perturbations around x_t, do the check of the R score(to get some sort of robustness): $S(x_t) = E_{\delta \sim U[-\mu,\mu]}[R(x_t + \delta)]$
- Use S score as the score in the partial algorithm

Experiment result

Threat model	Success rate	Median queries
QL	99.2%	11,550
PI	93.6%	49,624
LO	90%	54,063

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). We perform attacks over 1000 randomly chosen test images (100 for label-only) with randomly chosen target classes. For each attack, we use the same hyperparameters across all images. Here, we report the overall success rate (percentage of times the adversarial example was classified as the target class) and the median number of queries required.

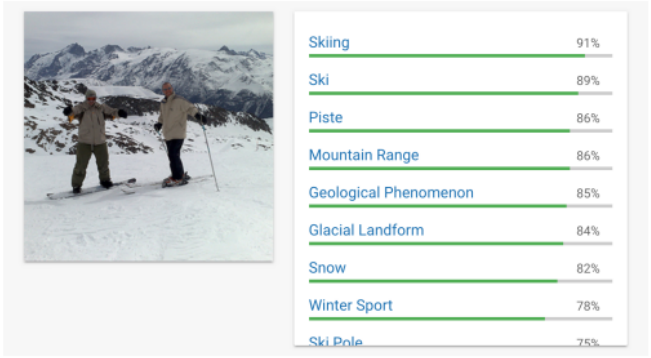


Figure 4. The Google Cloud Vision Demo labeling on the unperturbed image.

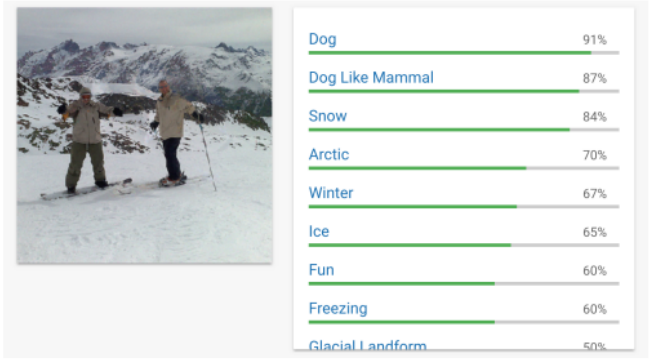


Figure 5. The Google Cloud Vision Demo labeling on the adversarial image generated with ℓ_∞ bounded perturbation with $\epsilon = 0.1$: the image is labeled as the target class.

2. Practical Black-Box Attacks against Machine Learning

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, Ananthram Swami

- 2017 ACM Asia Conference on Computer and Communications Security
- Two steps:
- 1. Query to build a substitute model
- 2. Craft adversarial samples

Substitute DNN training

- Random Noise: No, as noise doesn't represent input distribution
- Identifying directions in which the model's output is varying, around an initial set of training points.
- "Jacobian-based dataset augmentation"

Algorithm 1 - Substitute DNN Training: for oracle $\tilde{O}$, a maximum number max_ρ of substitute training epochs, a substitute architecture F, and an initial training set S_0.

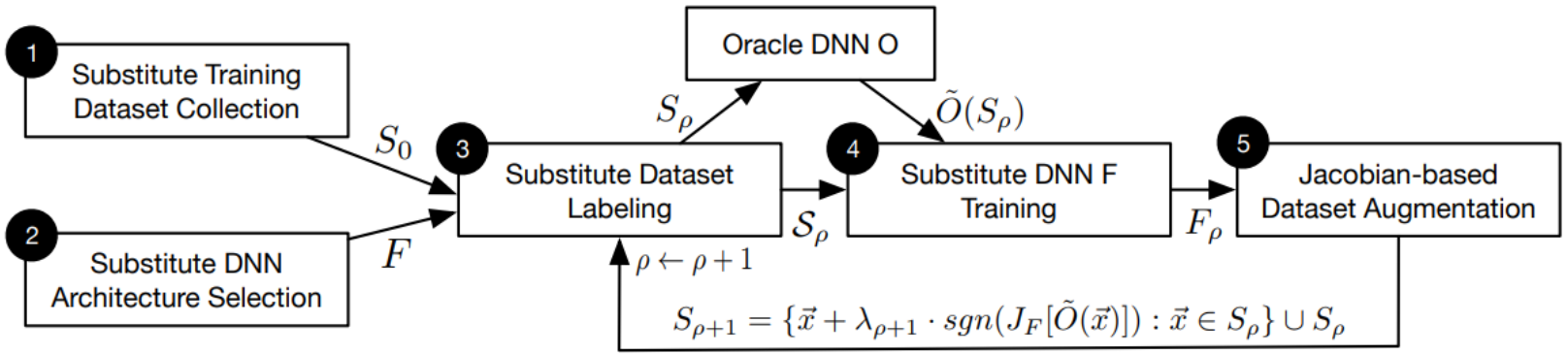
Input: $\tilde{O}$, max_ρ, S_0, λ

Define architecture F
for $\rho \in 0 \,..\, max_\rho - 1$ **do**
    // Label the substitute training set
    $D \leftarrow \left\{(\vec{x}, \tilde{O}(\vec{x})) : \vec{x} \in S_\rho\right\}$
    // Train F on D to evaluate parameters θ_F
    $\theta_F \leftarrow \text{train}(F, D)$
    // Perform Jacobian-based dataset augmentation
    $S_{\rho+1} \leftarrow \{\vec{x} + \lambda \cdot \text{sgn}(J_F[\tilde{O}(\vec{x})]) : \vec{x} \in S_\rho\} \cup S_\rho$
end for
return θ_F

“Jacobian-based dataset augmentation”

- Goal: Find the direction of varying output

$$S_{\rho+1} = \{\vec{x} + \lambda \cdot \text{sgn}(J_F[\tilde{O}(\vec{x})]) : \vec{x} \in S_\rho\} \cup S_\rho$$



Experiment result

DNN ID	Accuracy ($\rho = 2$)	Accuracy ($\rho = 6$)	Transferability ($\rho = 6$)
A	30.50%	82.81%	75.74%
F	68.67%	79.19%	64.28%
G	72.88%	78.31%	61.17%
H	56.70%	74.67%	63.44%
I	57.68%	71.25%	43.48%
J	64.39%	68.99%	47.03%
K	58.53%	70.75%	54.45%
L	67.73%	75.43%	65.95%
M	62.64%	76.04	62.00%

Table 1: **Substitute Accuracy** at $\rho = 2$ and $\rho = 6$ substitute epochs and **Transferability of Adversarial Samples:** for $\varepsilon = 0.4$ after $\rho = 6$ substitute epochs.

3. Simple Black-Box Adversarial Perturbations for Deep Networks

- Arxiv 2016
- Reject by ICLR 2017

Simple attack – Change 1 pixel

- Change 1 pixel – with enormous change
- P=100 in their example

Algorithm 1 RANDADV (NN)

Input: Image I with true label $c(I) \in \{1, \ldots, C\}$, perturbation factor $p \in \mathbb{R}$, and a budget $U \in \mathbb{N}$ on the number of trials
Output: A randomized estimate on the fraction of critical pixels in the input image I
$i = 1$, critical $= 0$
while $i \leq U$ **do**
  randomly pick a pixel $(\star, x, y)$
  compute a perturbed image $I_p^{(x,y)} = \text{PERT}(I, p, x, y)$
  if $c(I) \notin \pi(\text{NN}(I_p^{(x,y)}), 1)$ **then**
    critical $\leftarrow$ critical $+ 1$
  end if
  $i \leftarrow i + 1$
end while
{The algorithm succeeds in generating an adversarial image if it finds at least one critical pixel}
return $\frac{\text{critical}}{U}$

$$I_p^{(x,y)}(b,u,v) \stackrel{\text{defn}}{=} \begin{cases} (I(b,u,v) & \text{if } x \neq u \text{ or } y \neq v \\ p \times \text{sign}(I(b,u,v)) & \text{otherwise} \end{cases}$$

Prior convictions: Black-box Adversarial Attacks with Bandits and Priors

Andrew Ilyas, Logan Engstrom, Aleksander Madry

- Summary:
  - Claim gradient estimation is an accurate and easy way to generate adversarial samples
  - Claim prior of the gradient is important, and define two such priors
  - Propose bandit to estimate the priors

Background: Adversarial attack

- Adversarial sample problem:

$$x' = \underset{x':|x'-x|_p \leq \epsilon_p}{\operatorname{argmax}} L(x', y)$$

- L is not convex, therefore, often solved by first order methods, especially projected gradient descent:

$$x_l = \Pi_{B_p(x,\epsilon)}(x_{l-1} + \eta s_l)$$

$$s_l = \Pi_{\partial B_p(0,1)} \nabla_x L(x_{l-1}, y)$$

- $B_p(x, \epsilon)$ stands for l_p norm ball of radius ϵ around x

Background: Black-box attacks

- We can estimate the gradient with direct value queries:

$$D_v f(x) = \langle \nabla_x f(x), v \rangle \approx \left(f(x + \delta v) - f(x) \right) / \delta.$$

- Direct way:

$$\widehat{\nabla}_x L(x, y) = \sum_{k=1}^{d} e_k \left(L(x + \delta e_k, y) - L(x, y) \right) / \delta \approx \sum_{k=1}^{d} e_k \langle \nabla_x L(x, y), e_k \rangle$$

- However, requires a lot of queries.
  - Inception net on Imagenet has 268,203 dims
  - 3*224*224 = 150,528
  - Which means 10^5 level of queries

However, don’t need perfect gradient

- With only part of gradient, still possible to generate adversarial samples on ImageNet

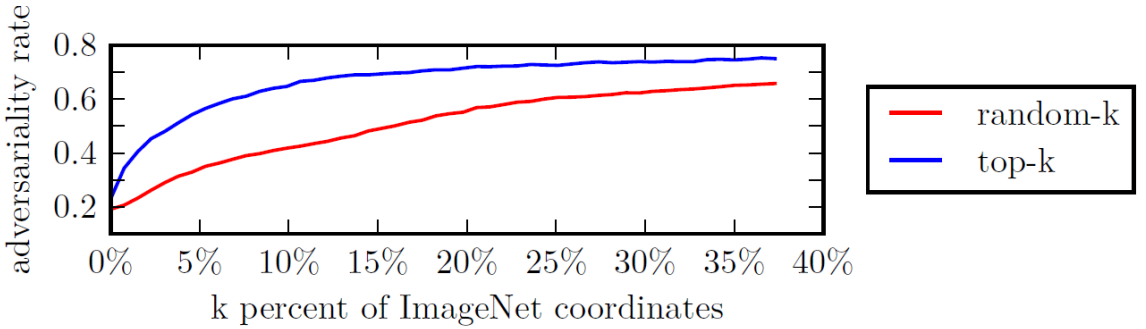


Figure 1: The fraction of correctly estimated coordinates of $\mathrm{sgn}(\nabla_x L(x, y))$ required to successfully execute the single-step PGD (also known as FGSM) attack, with $\epsilon = 0.05$. In the experiment, for each k, the top k percent – chosen either by magnitude (`top-k`) or randomly (`random-k`) – of the signs of the coordinates are set correctly, and the rest are set to $+1$ or -1 at random. The adversariality rate is the portion of 1,000 random ImageNet images misclassified after one FGSM step. Observe that, for example, estimating only 20% of the coordinates correctly leads to misclassification in the case of more than 60% of images.

Define: Gradient estimation problem

- For input pair (x, y), let $g^* = \nabla_x L(x, y)$, the goal is to find a unit vector $\hat{g}$, that maximize

$$\mathbb{E}\left[\hat{g}^T g^*\right]$$

with small number of queries

Baseline: Least square method

- Solve an approximation of the problem:
  - $\min_{\hat{g}} \left|\left|\hat{g}\right|\right|_2 \quad s.t. A\hat{g} = z$
  - Where A is a random Gaussian Matrix
- This is the same with NES(Natural Evolution Strategies), which is a query efficient black-box attack.
- Proved to be optimal

Theorem 2 (Least-squares optimality [BBEKY13]). *For a linear regression problem with known Gaussian errors, and a fixed number of predictors p, the least-squares estimator is the optimal M-estimator of the latent vector.*

Prior

- Though the random estimator is optimal, we can still improve it by given it a prior on its random sampling
- Random generator is optimal in general case, however we have extra information, which can be used to improve the method

Time-dependent prior

- In the projected gradient descent process, it requires multiple gradients among points within a fixed boundary
- Clearly they can be used as a prior
- Cos similarity result show this works
- Simply use the t-1 step gradient as the prior of gradient at time t

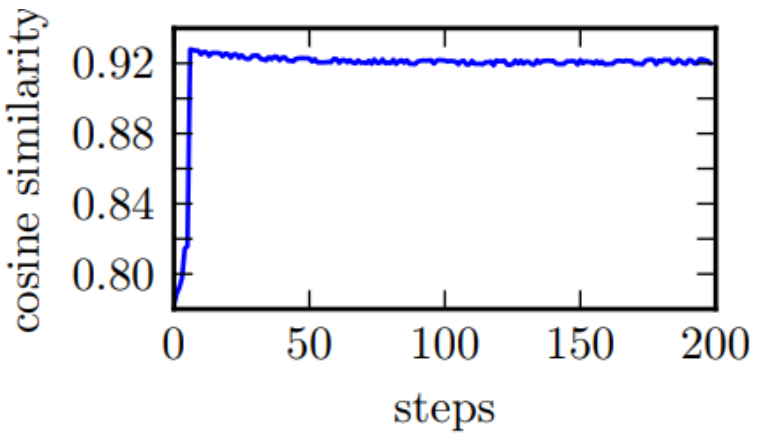


Figure 2: Cosine similarity between the gradients at the current and previous steps along the optimization trajectory of NES PGD attacks, averaged over 1000 random ImageNet images.

Data-dependent prior

- Pixels near each other tends to be similar, therefore, the gradient of them are also tend to be similar
- Measure the cos-similarity to an average blurred input:

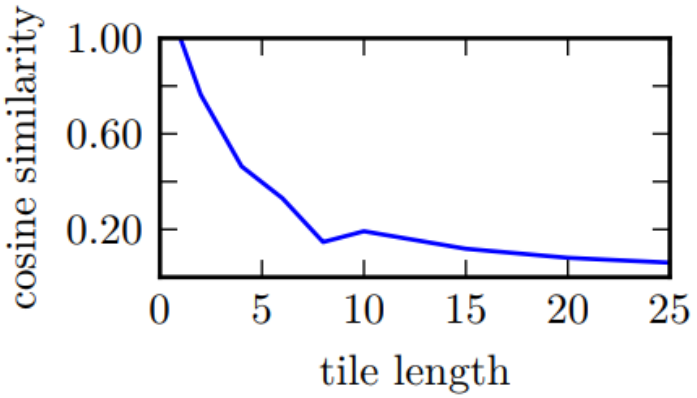


Figure 3: Cosine similarity of "tiled" image gradient with original image gradient versus the length of the square tiles, averaged over 5,000 randomly selected ImageNet images.

Bandit

- Action: The gradient chose
- Loss function: $\ell_t(g) = -\langle \nabla L(x,y), g \rangle$, Unknown, but can be estimated

- Use “reduction from bandit information”
- Update rule:

$$v_t = \mathcal{A}(v_{t-1}, \Delta_t) := \Pi_{\ell_2}\left[v_{t-1} + \eta \cdot \boldsymbol{\Delta}_t\right] = \begin{cases} v_{t-1} + \eta \cdot \boldsymbol{\Delta}_t \text{ if } ||v_{t-1} + \eta \cdot \boldsymbol{\Delta}_t||_2 < 1 \\ \frac{v_{t-1} + \eta \cdot \boldsymbol{\Delta}_t}{||v_{t-1} + \eta \cdot \boldsymbol{\Delta}_t||_2} \text{ otherwise,} \end{cases}$$

Algorithm: estimating the gradient

Algorithm 2 Single-query spherical estimate of $\nabla_v \langle \nabla L(x,y), v \rangle$

procedure GRAD-EST(x, y, v)
  $u \leftarrow \mathcal{N}(0, \frac{1}{d} I)$ // Query vector
  $\{q_1, q_2\} \leftarrow \{v + \delta \boldsymbol{u}, v - \delta \boldsymbol{u}\}$ // Antithetic samples
  // Incur bandit loss:
  $\ell_t(q_1) = -\langle \nabla L(x,y), q_1 \rangle \approx \frac{L(x,y) - L(x+\epsilon \cdot q_1, y)}{\epsilon}$
  $\ell_t(q_2) = -\langle \nabla L(x,y), q_2 \rangle \approx \frac{L(x,y) - L(x+\epsilon \cdot q_2, y)}{\epsilon}$
  $\boldsymbol{\Delta} \leftarrow \frac{\ell_t(q_1) - \ell_t(q_2)}{\delta} \boldsymbol{u} = \frac{L(x+\epsilon q_2, y) - L(x+\epsilon q_1, y)}{\delta \epsilon} \boldsymbol{u}$
  // Note that due to cancellations cancels we can actually evaluate $\boldsymbol{\Delta}$ with only two queries to L
  return $\boldsymbol{\Delta}$

Combined two optimization together

Algorithm 3 Adversarial Example Generation with Bandit Optimization for ℓ_2 norm perturbations

procedure ADVERSARIAL-BANDIT-L2(x_{init}, y_{init})
    // $C(\cdot)$ returns top class
    $v_0 \leftarrow \mathbf{0}_{1\times d}$
    $x_0 \leftarrow x_{init}$ // Adversarial image to be constructed
    while $C(x) = y_{init}$ **do**
        $g_t \leftarrow v_{t-1}$
        $x_t \leftarrow x_{t-1} + h \cdot \frac{g_t}{||g_t||_2}$
        $\Delta_t \leftarrow$ GRAD-EST($x_{t-1}, y_{init}, v_{t-1}$) // Estimated Gradient of ℓ_t
        $v_t \leftarrow \Pi_{\ell_2}\left[v_{t-1} + \eta \cdot \Delta_t\right]$
        $t \leftarrow t + 1$
    return x_{t-1}