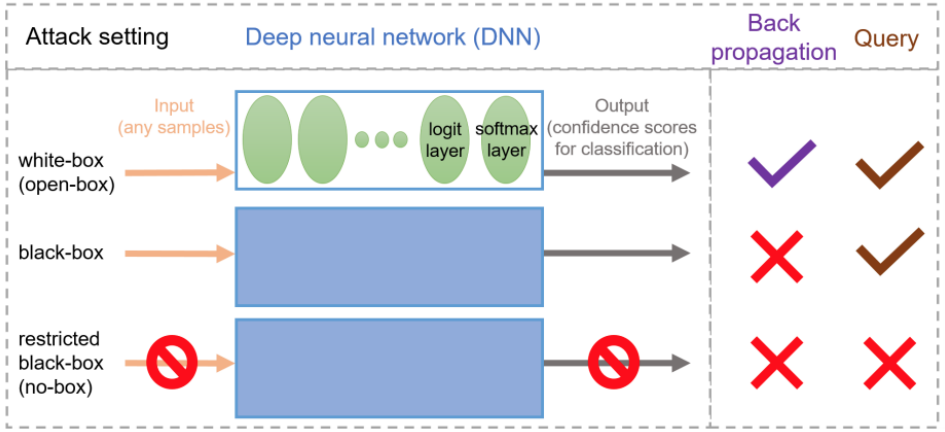
Evasion Black-box Attacks

Recall: adversarial attacks against general ML

- Greedy attacks against object detectors
- Optimization based method against object detectors/segmentation/human pose estimation etc
- Attack generative models by manipulating the representation space
- Attack decision making via manipulating the observation, action, reward, and environment variables
  - https://arxiv.org/pdf/2005.10247.pdf

Black-box attacks

- Zero-Query Attack
  - Random perturbation
  - Difference of means
  - Transferability based attack
- Query Based Attack
  - Finite difference gradient estimation
  - Query reduced gradient estimation



The zero-query attack can be viewed as a special case for the query based attack, where the number of queries made is zero

Transferability in machine learning: from phenomena to black-box attacks using adversarial samples

- Adversarial example

$$\vec{x^*} = \vec{x} + \delta_{\vec{x}} \ \textbf{where} \ \ \delta_{\vec{x}} = \arg\min_{\vec{z}} f(\vec{x} + \vec{z}) \neq f(\vec{x})$$

- Adversarial sample transferability

$$\Omega_X(f, f') = \left|\left\{f'(\vec{x}) \neq f'\left(\vec{x} + \delta_{\vec{x}}\right) : \vec{x} \in X\right\}\right|$$

  - Intra-technique transferability
  - Cross-technique transferability

Transferability in machine learning: from phenomena to black-box attacks using adversarial samples

- Transferability based black-box attack
  - Train a substitute model, and craft adversarial examples against the substitute, and transfer them to a victim model
- *Distillation* – use the victim model as an oracle to label a synthetic training set for the substitute
- *Reservoir sampling* – efficient data augmentation
  - SVM and decision trees which are non-differentiable models

Transferability in machine learning: from phenomena to black-box attacks using adversarial samples

- Jacobian-based dataset augmentation

$$S_{\rho+1} = \{\vec{x} + \lambda_\rho \cdot \mathrm{sgn}(J_f[\tilde{O}(\vec{x})] : \vec{x} \in S_\rho)\} \cup S_\rho$$

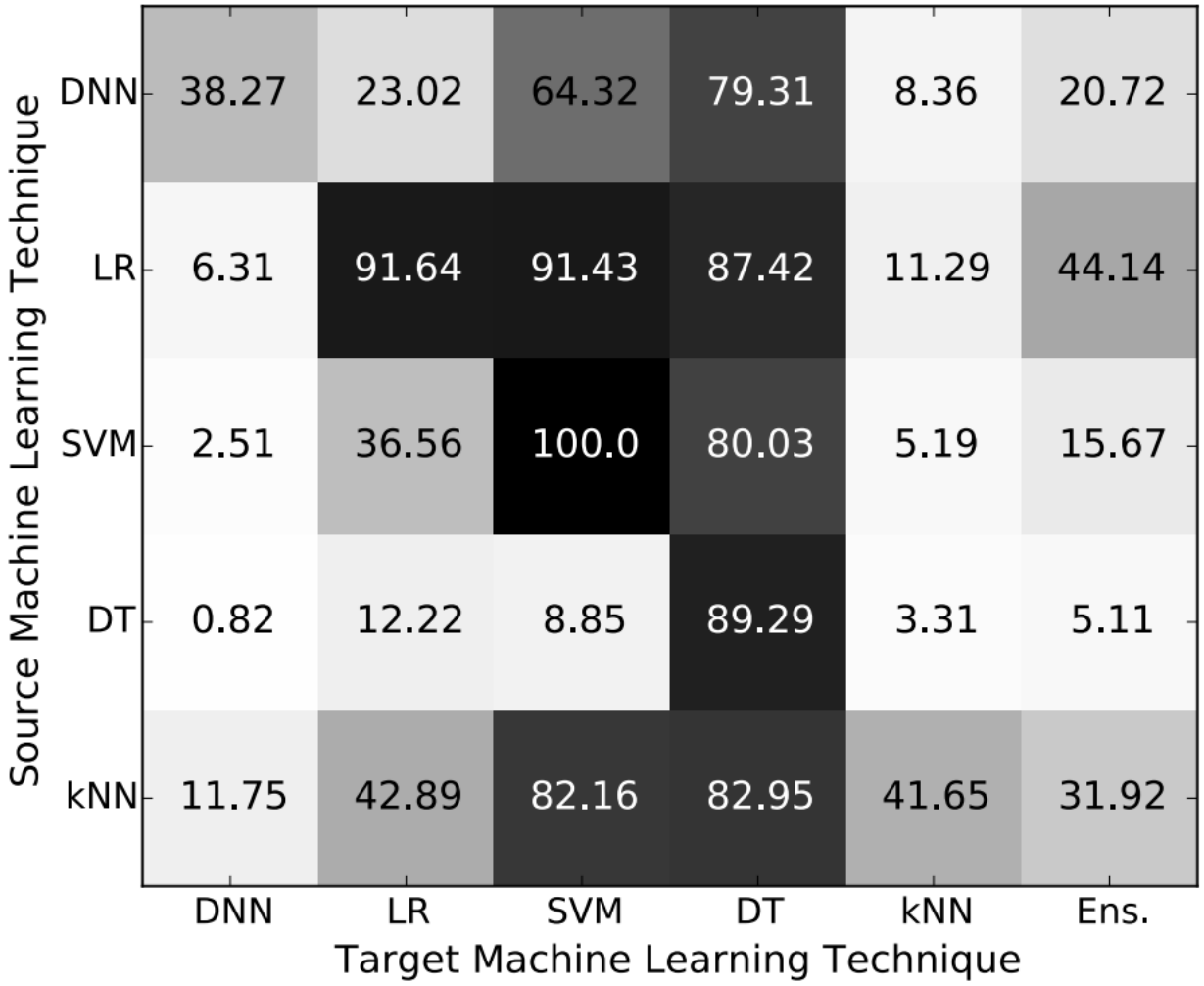
- Reservoir sampling

Algorithm 1 Jacobian-based augmentation with Reservoir Sampling: sets are considered as arrays for ease of notation.

Input: $S_{\rho-1}$, κ, J_f, λ_ρ

1: $N \leftarrow |S_{\rho-1}|$
2: Initialize S_ρ as array of $N + \kappa$ items
3: $S_\rho[0 : N-1] \leftarrow S_{\rho-1}$
4: **for** $i \in 0..\kappa - 1$ **do**
5: $\quad S_\rho[N+i] \leftarrow S_{\rho-1}[i] + \lambda_\rho \cdot \mathrm{sgn}(J_f[\tilde{O}(S_{\rho-1}[i])])$
6: **end for**
7: **for** $i \in \kappa..N-1$ **do**
8: $\quad r \leftarrow$ random integer between 0 and i
9: $\quad$ **if** $r < \kappa$ **then**
10: $\quad\quad S_\rho[N+r] \leftarrow S_{\rho-1}[i] + \lambda_\rho \cdot \mathrm{sgn}(J_f[\tilde{O}(S_{\rho-1}[i])])$
11: $\quad$ **end if**
12: **end for**
13: **return** S_ρ

Cross technique transferability



Cross-technique transferability matrix: cell (I,j) is the percentage of adversarial samples crafted to mislead a classifier learned using machine learning technique I that are misclassified by one trained with technique j

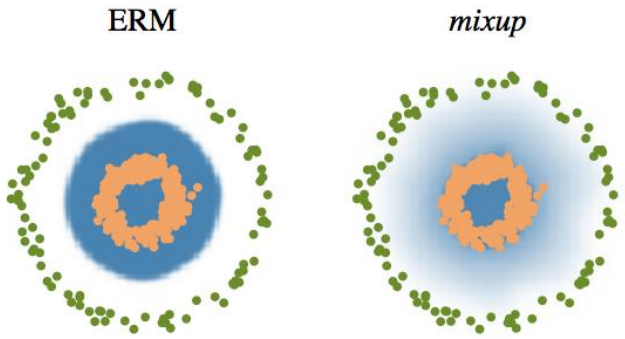
Takeaways

- Both intra-technique and cross-technique adversarial sample transferabilities are consistently strong phenomena across the space of machine learning techniques
- Black-box attacks are possible in practical settings against any unknown machine learning classifier
- Black-box attacks against classifiers hosted by Amazon and Google and achieve high misclassification rate, by training a logistic regression substitute model with only 800 queries

Interesting reading

- Mixup: Beyond Empirical Rish Minimization

$$\tilde{x} = \lambda x_i + (1-\lambda)x_j, \qquad \text{where } x_i, x_j \text{ are raw input vectors}$$
$$\tilde{y} = \lambda y_i + (1-\lambda)y_j, \qquad \text{where } y_i, y_j \text{ are one-hot label encodings}$$



  - Pros: improve the robustness of the networks
  - Cons: without guarantee for accuracy or robustness and not interpretable

Takeaways

- Different data augmentation can have opposite effects: increase attack transferability, or improve model robustness

Exploring the space of black-box attacks on deep neural networks

- Make queries to estimate gradient based on the output
- Need to know obtain the output of the logit layer
- Interesting point: simple feature reduction is efficient for query reduction

Query Based black-box attack

- Finite difference gradient estimation
  - Given *d*-dimensional vector *x*, we can make 2*d* queries to estimate the gradient as below

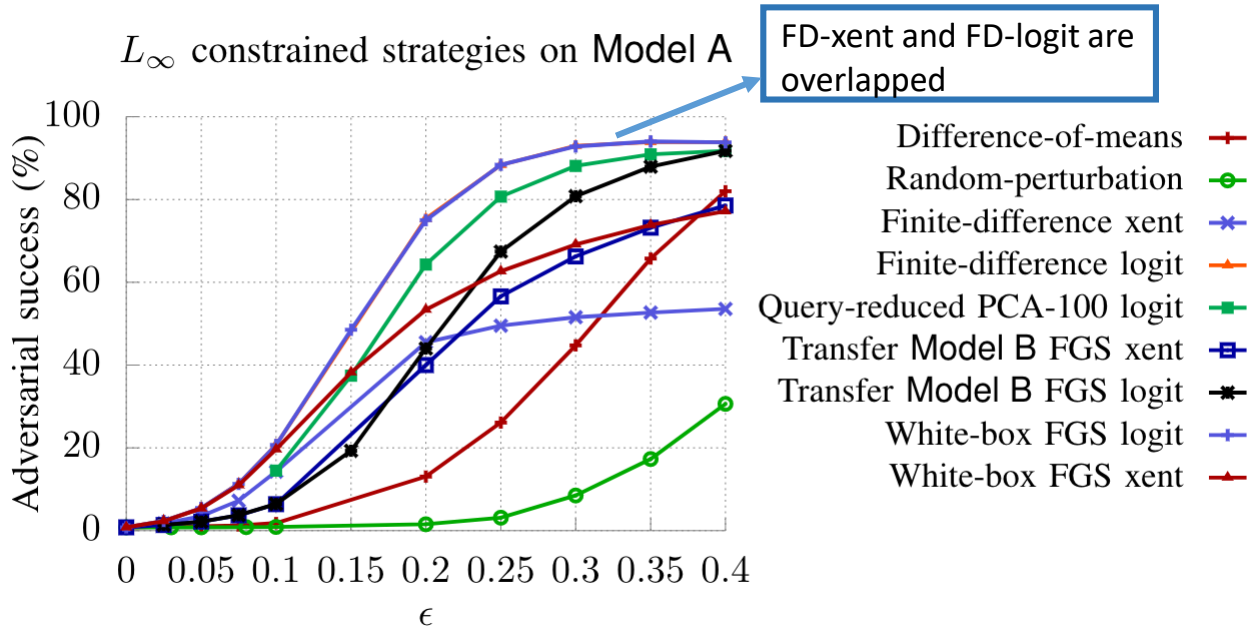
$$\mathrm{FD}_{\mathbf{x}}(g(\mathbf{x}), \delta) = \begin{bmatrix} \frac{g(\mathbf{x}+\delta\mathbf{e}_1)-g(\mathbf{x}-\delta\mathbf{e}_1)}{2\delta} \\ \vdots \\ \frac{g(\mathbf{x}+\delta\mathbf{e}_d)-g(\mathbf{x}-\delta\mathbf{e}_d)}{2\delta} \end{bmatrix} \qquad \hat{g}_i := \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_i} \approx \frac{f(\mathbf{x}+h\mathbf{e}_i) - f(\mathbf{x}-h\mathbf{e}_i)}{2h}$$

  - An example of approximate FGS with finite difference

$$x_{adv} = \mathbf{x} + \epsilon \cdot \mathrm{sign}\left(\mathrm{FD}_{\mathbf{x}}(\ell_f(\mathbf{x}, y), \delta)\right)$$

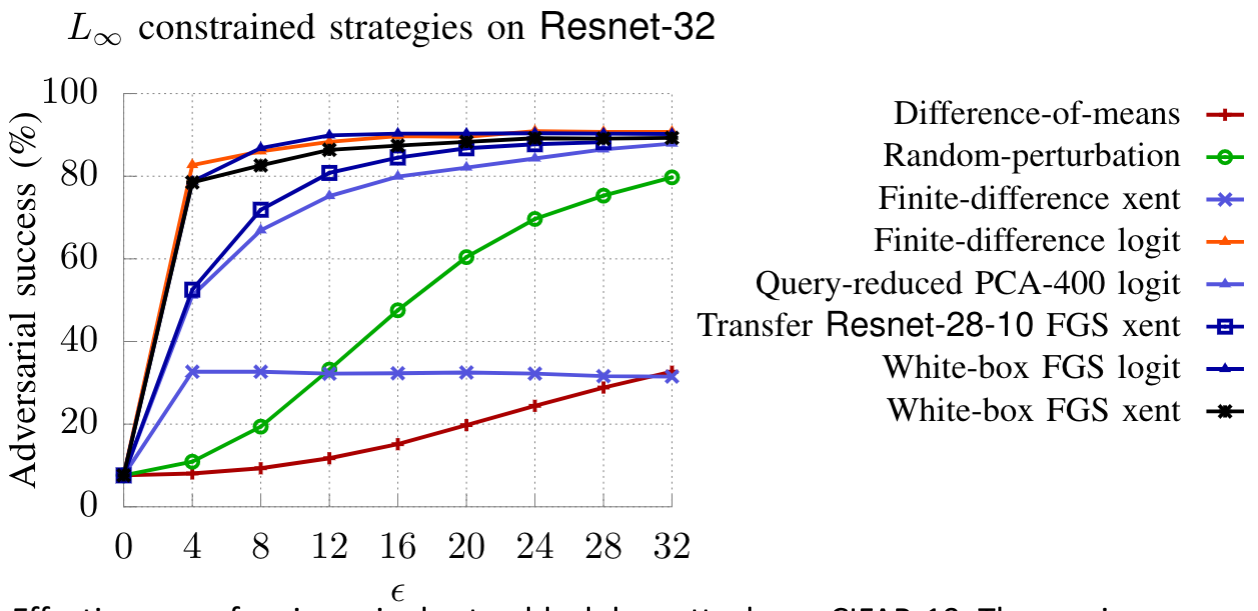
- Query reduced gradient estimation
  - Random grouping
  - PCA

Similarly, we can also approximate for logit-based loss by making 2d queries



Effectiveness of various single step black-box attacks on MNIST. The y-axis represents the variation in adversarial success as ϵ increases.

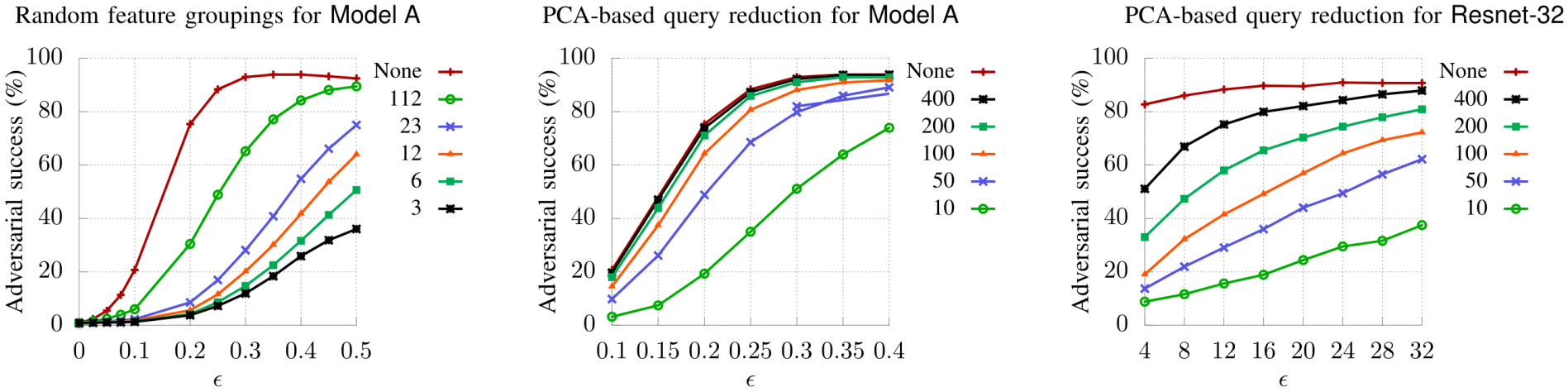
Finite Differences method outperform other black-box attacks and achieves similar attach success rate with the white-box attack



Effectiveness of various single step black-box attacks on CIFAR-10. The y-axis represents the variation in adversarial success as ϵ increases.

Finite Differences method outperform other black-box attacks and achieves similar attach success rate with the white-box attack

Gradient Estimation Attack with Query Reduction



Adversarial success rates for Gradient Estimation attacks with query reduction on Model A (MNIST) and Resnet-32 (CIFAR-10).

Finite Differences method with query reduction perform approximately similar with the gradient estimation black-box attack

Black-box Attack Clarifai

Original image, classified as “drug” with a confidence of 0.99

Adversarial example, classified as “safe” with a confidence of 0.96

The Gradient Estimation black-box attack on Clarifai’s Content Moderation Model

Takeaways

- Without relying on transferability, it is also possible to conduct black-box attacks
- Gradient estimation is accurate based on finite difference method
- It is possible to reduce the number of queries and still obtain good gradient approximation

Similar work

- ZOO: zeroth order optimization based black-box attacks to deep neural networks without training substitute models
  - Estimate gradient based on queries
  - Also need to access the logit layer results
  - Need to make large amount of queries
  - Difference: apply optimization based attack with the estimated gradient

Interesting reading

- Our transferability proof?
- The Space of Transferable Adversarial Examples
  - Adversarial examples span a continuous subspace of large (~25) dimensionality
  - For two different models, a significant fraction of their subspaces is shared, thus enabling transferability
  - Empirically show similarity of different models' decision boundaries: boundaries are actually close in arbitrary directions, whether adversarial or benign

If two models achieve low error for some task while also exhibiting low robustness to adversarial examples, adversarial examples crafted on one model transfer to the other.

Related reading

- Adversarial Learning
  - For linear classifier with binary features, it is possible prove efficiency for black-box attack
  - What's ACRE $(1+\epsilon)$ - learnable?
  - How to prove it?
  - Is it possible to apply it to DNNs? -- the next paper

Algorithm 3 FINDBOOLEANIMAC($\mathbf{x}^a, \mathbf{x}^-$)

$\mathbf{y} \leftarrow \mathbf{x}^-$
repeat
  $\mathbf{y}^{prev} \leftarrow \mathbf{y}$
  for all $f \in C_y$ **do**
    toggle f in $\mathbf{y}$
    if $c(\mathbf{y}) = 1$ **then**
      toggle f in $\mathbf{y}$
    end if
  end for
  for all $f_1 \in C_y; f_2 \in C_y; f_3 \notin C_y$ **do**
    toggle f_1, f_2, and f_3 in $\mathbf{y}$
    if $c(\mathbf{y}) = 1$ **then**
      toggle f_1, f_2, and f_3 in $\mathbf{y}$
    end if
  end for
until $\mathbf{y}^{prev} = \mathbf{y}$
return $\mathbf{y}$

Decision-Based Adversarial Attacks: Reliable Attacks Against Black-box Machine Learning Models

Existing Attacks & Defences

Gradient-based	‘modify input w.r.t. loss with the help of gradient’	mask the gradients!
Score-based	‘use predictions to numerically estimate the gradient’	add stochastic elements!
Transfer-based	‘attack one, attack the other with the help of training data’	robust training!

Existing Attacks & Defences

Gradient-based	‘modify input w.r.t. loss with the help of gradient’	mask the gradients!
Score-based	‘use predictions to numerically estimate the gradient’	add stochastic elements!
Transfer-based	‘attack one, attack the other with the help of training data’	robust training!

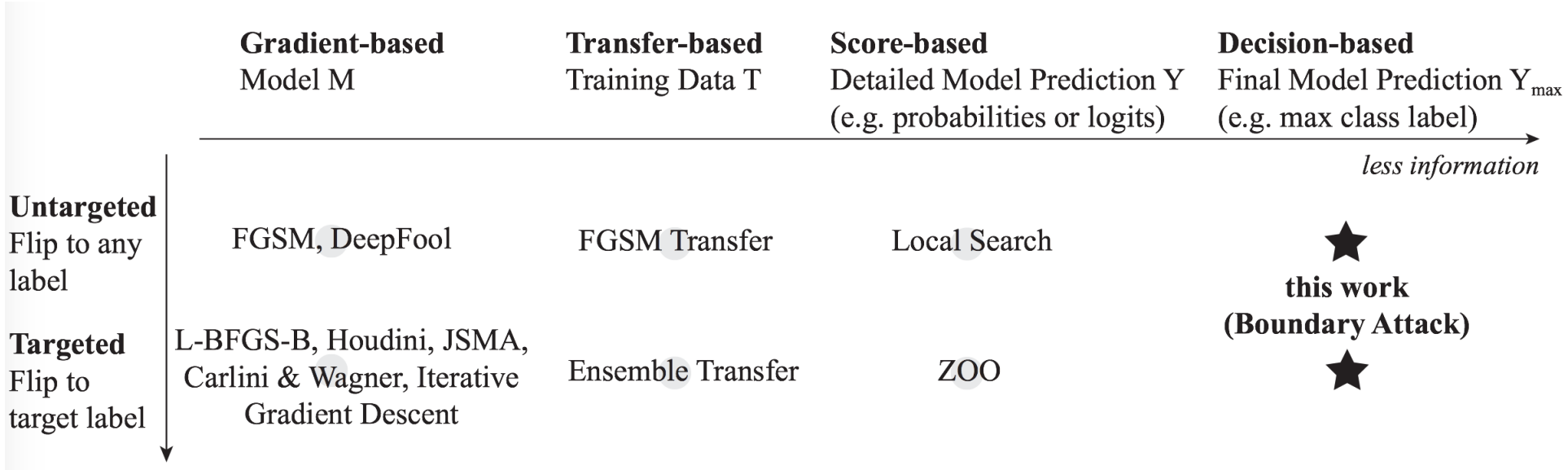
Easy to defend against!

Existing Attacks & Defences

Gradient-based	‘modify input w.r.t. loss with the help of **gradient**’	mask the gradients! **Easy to defend against!**
Score-based	**Hard to get!** ‘use **predictions** to numerically estimate the gradient’	add stochastic elements!
Transfer-based	‘attack one, attack the other with the help of **training data**’	robust training!

Boundary Attack

- Direct attacks that solely rely on the final decision of the model
- Starts from a large adversarial perturbation and then seeks to reduce the perturbation while staying adversarial

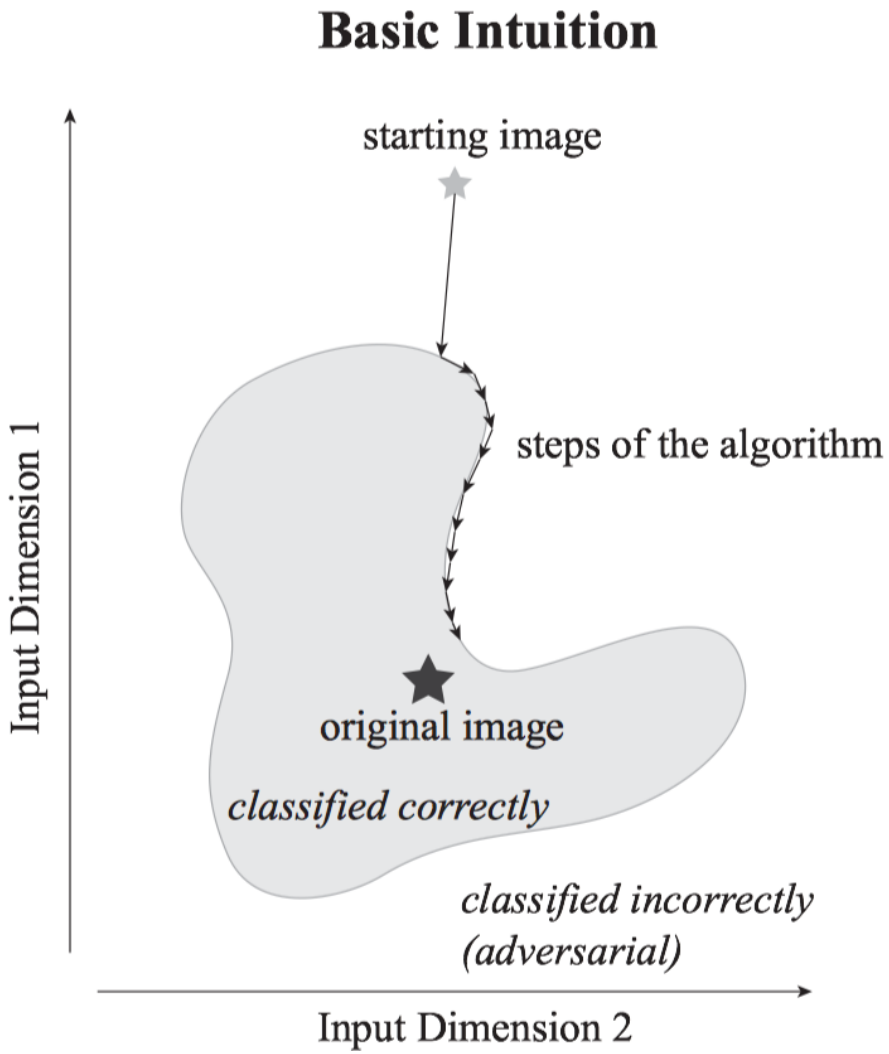


Boundary Attack

Data: original image $\mathbf{o}$, adversarial criterion $c(.)$, decision of model $d(.)$
Result: adversarial example $\tilde{\boldsymbol{o}}$ such that the distance $d(\boldsymbol{o}, \tilde{\boldsymbol{o}}) = \|\boldsymbol{o} - \tilde{\boldsymbol{o}}\|_2^2$ is minimized
initialization: $k = 0$, $\tilde{\boldsymbol{o}}^0 \sim \mathcal{U}(0, 1)$ s.t. $\tilde{\boldsymbol{o}}^0$ is adversarial;
while $k <$ *maximum number of steps* **do**
  draw random perturbation from proposal distribution $\boldsymbol{\eta}_k \sim \mathcal{P}(\tilde{\boldsymbol{o}}^{k-1})$;
  if $\tilde{\boldsymbol{o}}^{k-1} + \boldsymbol{\eta}_k$ *is adversarial* **then**
    set $\tilde{\boldsymbol{o}}^k = \tilde{\boldsymbol{o}}^{k-1} + \boldsymbol{\eta}_k$;
  else
    set $\tilde{\boldsymbol{o}}^k = \tilde{\boldsymbol{o}}^{k-1}$;
  end
  $k = k + 1$
end

Algorithm 1: Minimal version of the Boundary Attack.

Boundary Attack: Intuition



- After each step:
  - be an image!
  - don’t change too much!
  - be closer to original!

Boundary Attack

Data: original image $\mathbf{o}$, adversarial criterion $c(.)$, decision of model $d(.)$
Result: adversarial example $\tilde{\boldsymbol{o}}$ such that the distance $d(\boldsymbol{o}, \tilde{\boldsymbol{o}}) = \|\boldsymbol{o} - \tilde{\boldsymbol{o}}\|_2^2$ is minimized
initialization: $k = 0$, $\tilde{\boldsymbol{o}}^0 \sim \mathcal{U}(0, 1)$ s.t. $\tilde{\boldsymbol{o}}^0$ is adversarial;
while $k <$ *maximum number of steps* **do**
    draw random perturbation from proposal distribution $\boldsymbol{\eta}_k \sim \mathcal{P}(\tilde{\boldsymbol{o}}^{k-1})$;
    if $\tilde{\boldsymbol{o}}^{k-1} + \boldsymbol{\eta}_k$ *is adversarial* **then**
        set $\tilde{\boldsymbol{o}}^k = \tilde{\boldsymbol{o}}^{k-1} + \boldsymbol{\eta}_k$;
    else
        set $\tilde{\boldsymbol{o}}^k = \tilde{\boldsymbol{o}}^{k-1}$;
    end
    $k = k + 1$
end

Algorithm 1: Minimal version of the Boundary Attack.

Proposal Distribution

1. The perturbed sample lies within the input domain,

$$\tilde{o}_i^{k-1} + \eta_i^k \in [0, 255]. \tag{1}$$

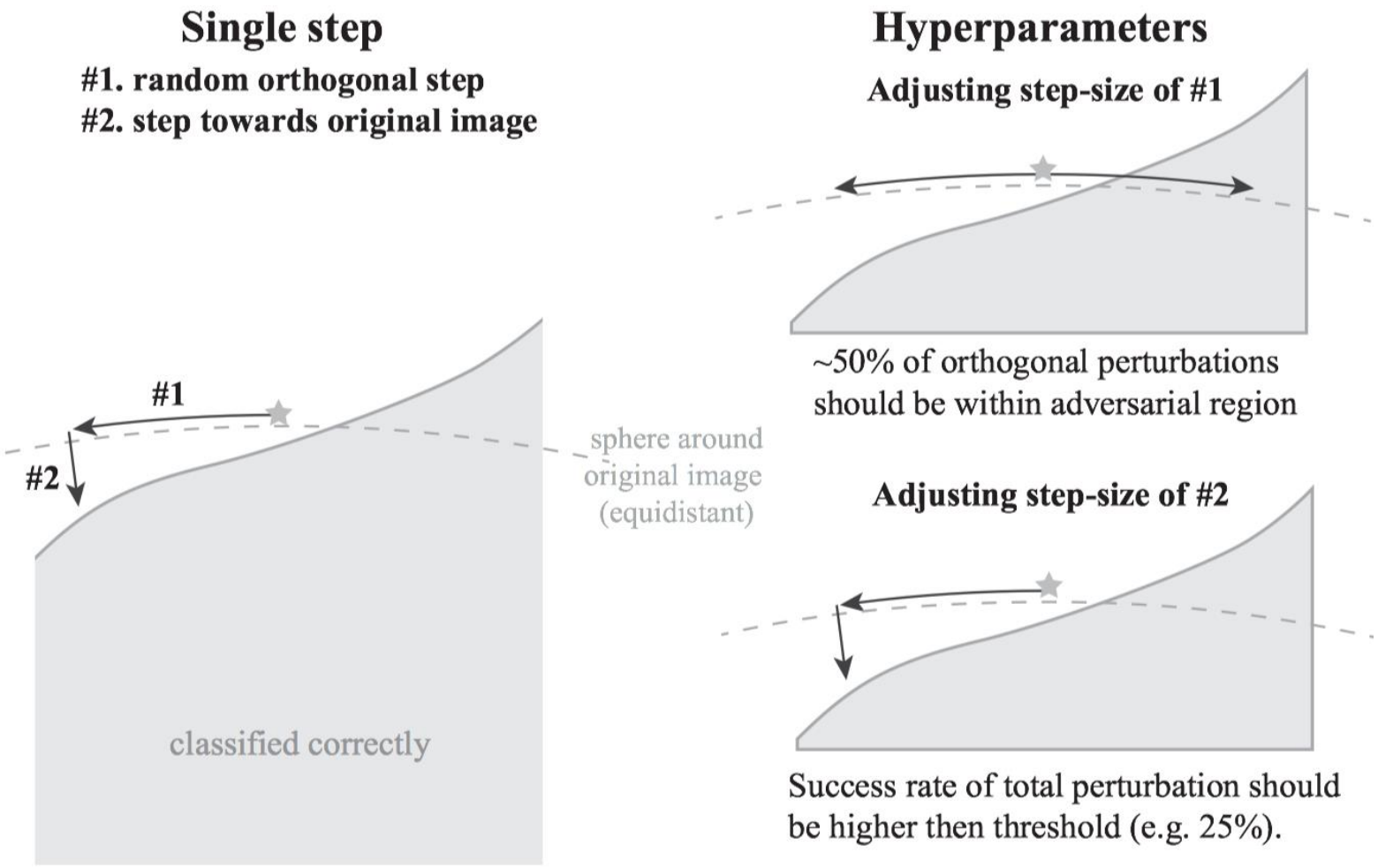
2. The perturbation has a relative size of δ,

$$\left\|\boldsymbol{\eta}^k\right\|_2 = \delta \cdot d(\mathbf{o}, \tilde{\mathbf{o}}^{\mathbf{k-1}}). \tag{2}$$

3. The perturbation reduces the distance of the perturbed image towards the original input by a relative amount ϵ,

$$d(\mathbf{o}, \tilde{\mathbf{o}}^{\mathbf{k-1}}) - d(\mathbf{o}, \tilde{\mathbf{o}}^{\mathbf{k-1}} + \boldsymbol{\eta}^k) = \epsilon \cdot d(\mathbf{o}, \tilde{\mathbf{o}}^{\mathbf{k-1}}). \tag{3}$$

Boundary Attack: one step

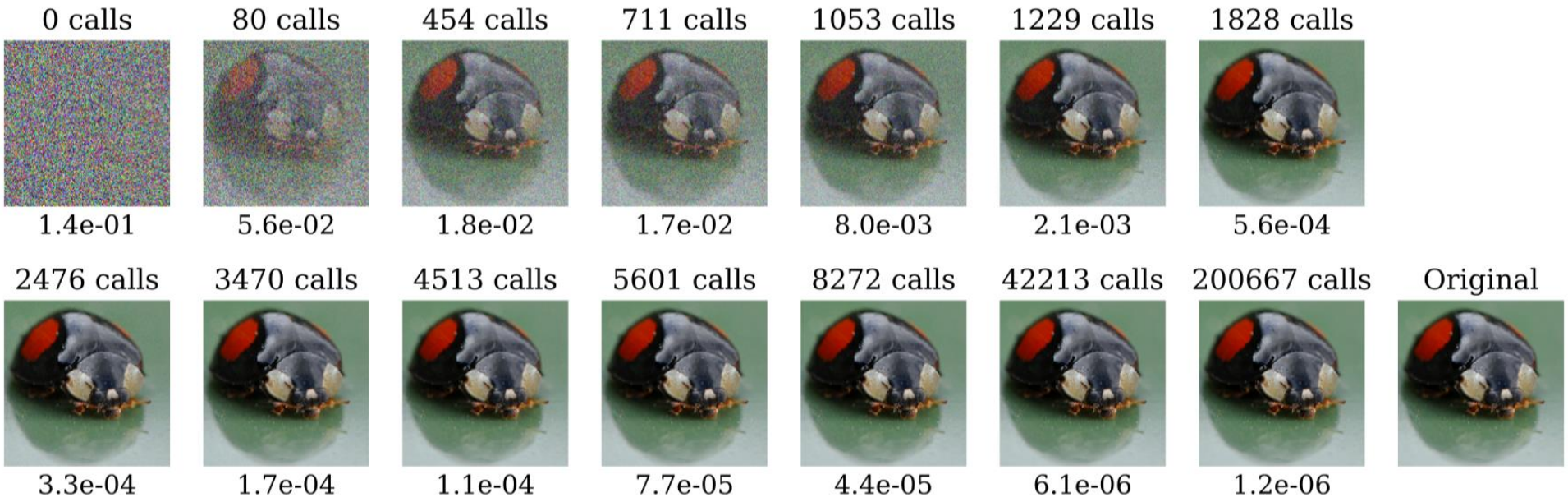


Evaluation Metric

$$\mathcal{S}_A(M) = \underset{i}{\text{median}} \left(\frac{1}{N} \left\| \boldsymbol{\eta}_{A,M}(\boldsymbol{o}_i) \right\|_2^2 \right)$$

$\boldsymbol{\eta}_{A,M}(\boldsymbol{o}_i) \in \mathbb{R}^N$ the adversarial perturbation that the attack A finds on model M for the i-th sample o_i .

Performance: Untargeted Attack



	Attack Type	MNIST	CIFAR	ImageNet		
				VGG-19	ResNet-50	Inception-v3
FGSM	gradient-based	4.2e-02	2.5e-05	1.0e-06	1.0e-06	9.7e-07
DeepFool	gradient-based	4.3e-03	5.8e-06	1.9e-07	7.5e-08	5.2e-08
Carlini & Wagner	gradient-based	2.2e-03	7.5e-06	5.7e-07	2.2e-07	7.6e-08
Boundary (ours)	decision-based	3.6e-03	5.6e-06	2.9e-07	1.0e-07	6.5e-08

Performance: Targeted Attack

	Attack Type	MNIST	CIFAR	VGG-19
Carlini & Wagner	gradient-based	4.8e-03	3.0e-05	5.7e-06
Boundary (ours)	decision-based	6.5e-03	3.3e-05	9.9e-06

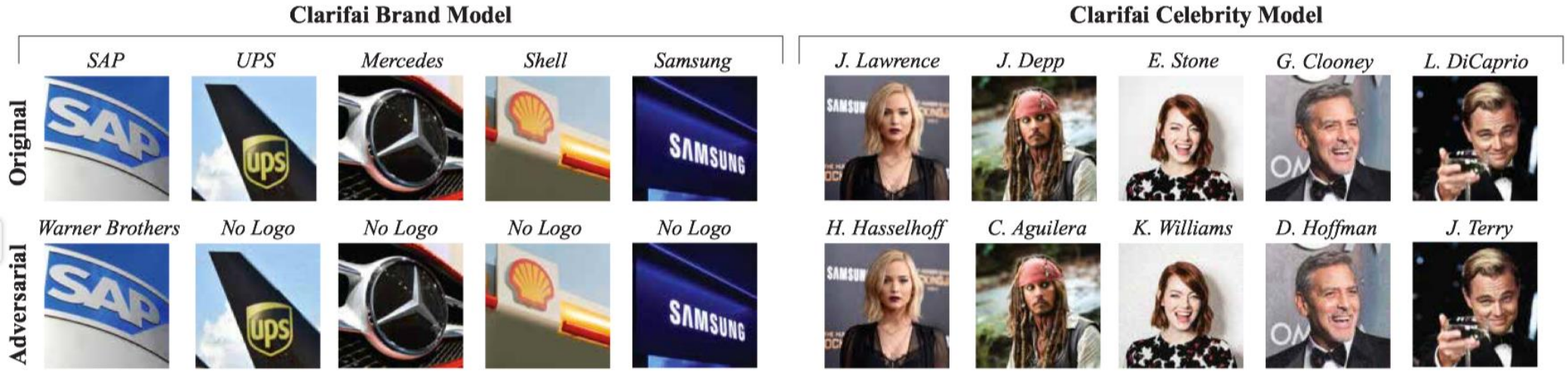
Attack Defensive Distillation

$$softmax(x, T)_i = \frac{e^{x_i/T}}{\sum_j e^{x_j/T}}$$

1. Train a teacher network as usual but with temperature T.
2. Train a distilled network—with the same architecture as the teacher—on the softmax outputs of the teacher. Both the distilled network and the teacher use temperature T.
3. Evaluate the distilled network at temperature $T = 1$ at test time.

		MNIST		CIFAR	
	Attack Type	standard	distilled	standard	distilled
FGSM	gradient-based	4.2e-02	fails	2.5e-05	fails
Boundary (ours)	decision-based	3.6e-03	4.2e-03	5.6e-06	1.3e-05

Real-world Application Attack



Takeaways

- Boundary Attack: decision-based black-box attack
- Drawback:
    - need many iterations to converge;
    - may trapped in local minimum.

Similar reading

- HopSkipJumpAttack: A Query-Efficient Decision-Based Attack

Algorithm 2 HopSkipJumpAttack

Require: Classifier C, a sample x, constraint ℓ_p, initial batch size B_0, iterations T.
Ensure: Perturbed image x_t.
  Set θ (Equation (17)).
  Initialize at $\tilde{x}_0$ with $\phi_{x^\star}(\tilde{x}_0) = 1$.
  Compute $d_0 = \|\tilde{x}_0 - x^\star\|_p$.
  for t in $1, 2, \ldots, T-1$ **do**
    (Boundary search)
    $x_t = \text{BIN-SEARCH}(\tilde{x}_{t-1}, x, \theta, \phi_{x^\star}, p)$
    (Gradient-direction estimation)
    Sample $B_t = B_0\sqrt{t}$ unit vectors $u_1, \ldots, u_{B_t}$.
    Set δ_t (Equation (17)).
    Compute $v_t(x_t, \delta_t)$ (Equation (14)).
    (Step size search)
    Initialize step size $\xi_t = \|x_t - x^\star\|_p/\sqrt{t}$.
    while $\phi_{x^\star}(x_t + \varepsilon_t v_t) = 0$ **do**
      $\xi_t \leftarrow \xi_t/2$.
    end while
    Set $\tilde{x}_t = x_t + \xi_t v_t$.
    Compute $d_t = \|\tilde{x}_t - x^\star\|_p$.
  end for
  Output $x_t = \text{BIN-SEARCH}(\tilde{x}_{t-1}, x, \theta, \phi_{x^\star}, p)$.