DeepSearch: A Simple and Effective Blackbox Attack for Deep Neural Networks

Team11: Hay in Needlestack
Tung-Duong Mai
Minsoo Kang
Aitolkyn Baigutanova
Sanjarbek Rakhmonov

Outline

I - Problem

II - Methodology

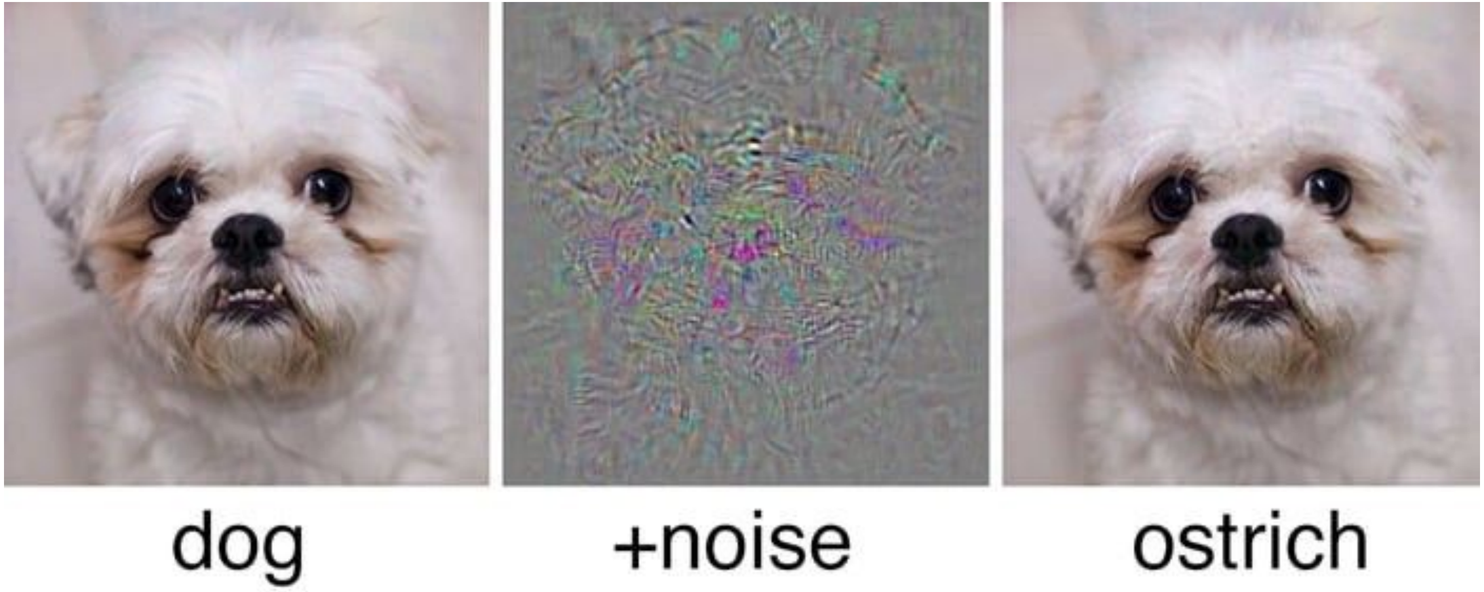
III - Evaluation

Paper Contribution

Devise a **black-box** **adversarial** attack that is **simple but effective** for image classifiers.

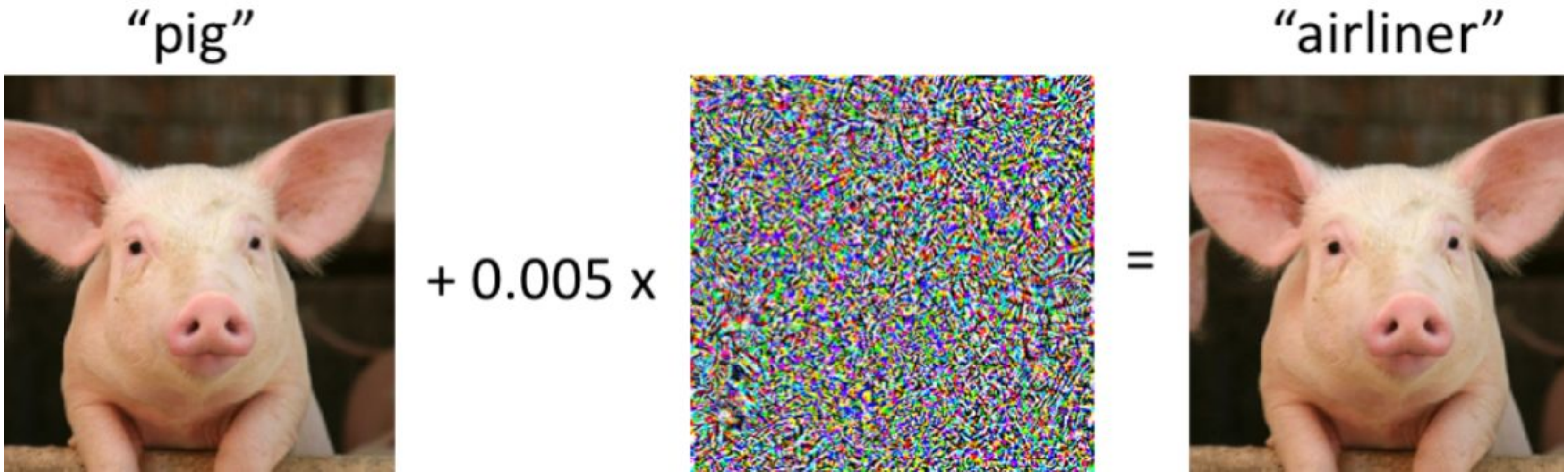
Adversarial attack

Adversarial attack is a machine learning technique attempting to fool models by supplying corrupted input.



Adversarial attack

By inserting a small noise to the original input, which is undetectable to humans, a different output is produced by the network.



Why is adversarial attack important?

(White image was possibly taken as open space.)

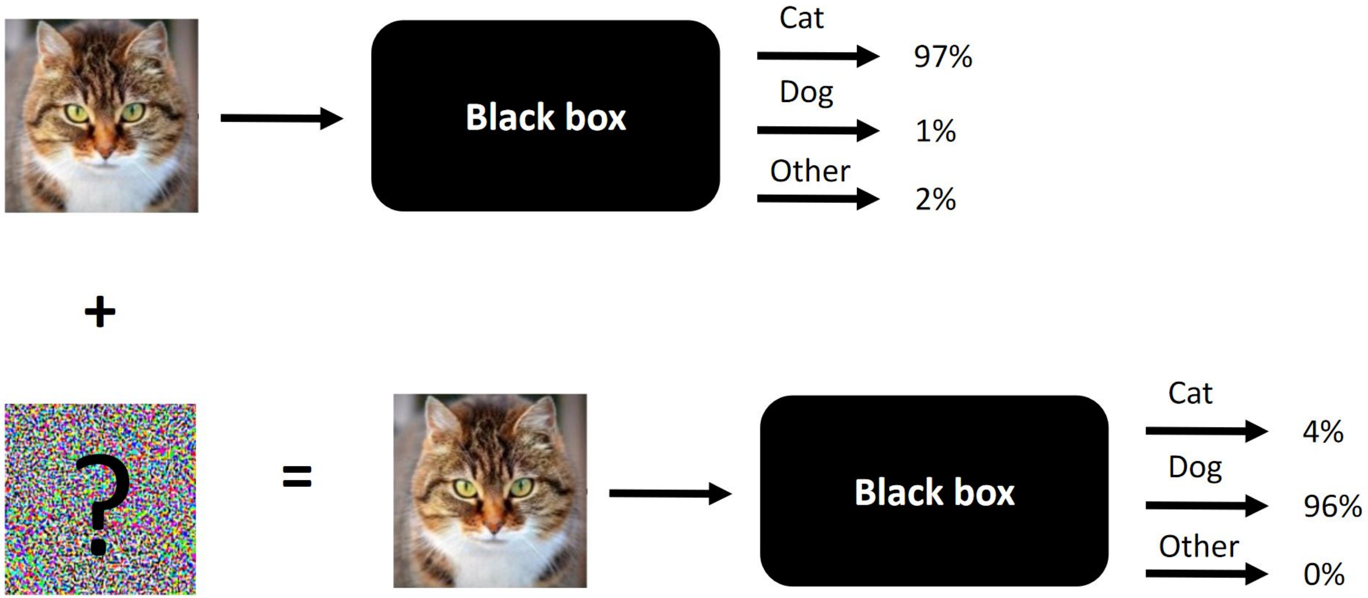
Definition: Adversarial example

1. Distance between input image x and perturbed image x' in R^n is sufficiently small (distance metric).

2. N(x) != N(x'), where N: $R^n \rightarrow C_m$ is classification into m classes

Blackbox

Attack done without knowing the internal structure of the model, which can only utilize the output as feedback, is called a **blackbox attack.**



Approach: simple & effective

Simple: search-based

Effective:

- High attack success rate
- Small number of queries → hierarchical grouping
- Small distortion from original input

Approach: design

Feedback-directed fuzzing:

*Mutate pixels to find adversarial inputs

Iterative refinement:

*Refinement to reduce L-distance of an adversarial input

Query reduction:

*Hierarchical grouping for simultaneous fuzzing

Methodology

Main Algorithm

Improvements - Refinement

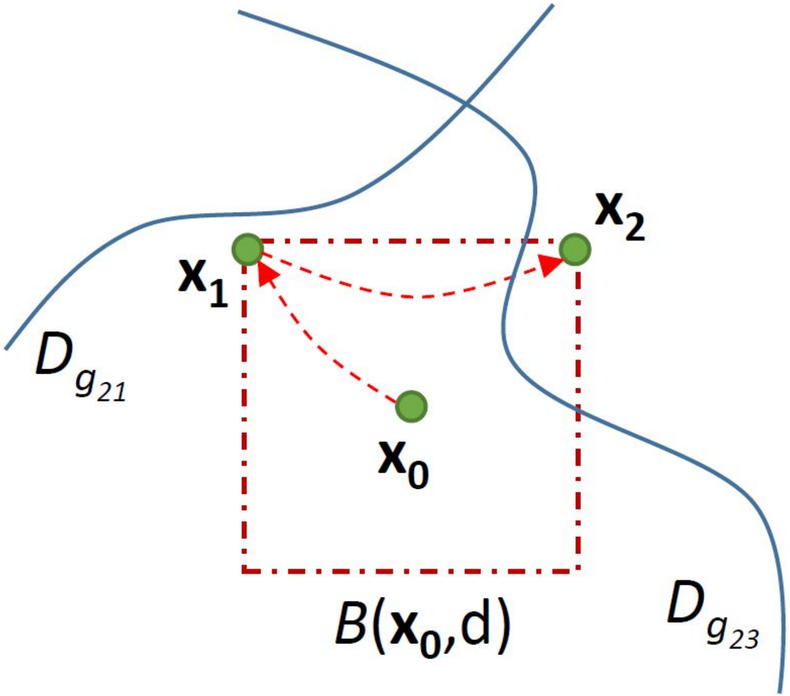
Improvements - Grouping

1. Introduce 2D illustration (Fig 1)
2. Explain what is adversarial in the context (x_0 => x'_0)
3. Main algorithm (Binary)
   A. Exploratory step + choosing upper or lower bound
   B. For linear case, no iteration is needed.
   C. For non-linear case, the process is iterated to approximate. (The iterative result still stays inside the boundary of B(x, d)
4. Multiclass
   A. Multi class is combinations of binary

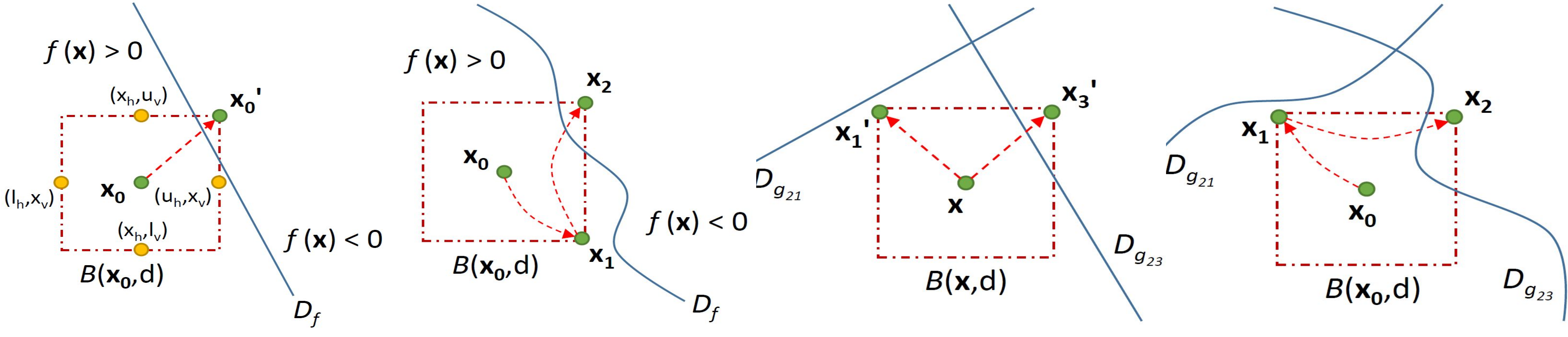
Algorithm: Before we go in...

1. Input 
2. Mutation / Modification
3. Classifier
4. Adversarial
5. Distortion Boundary

Simplified in illustration



Algorithm: Simple to Generalization

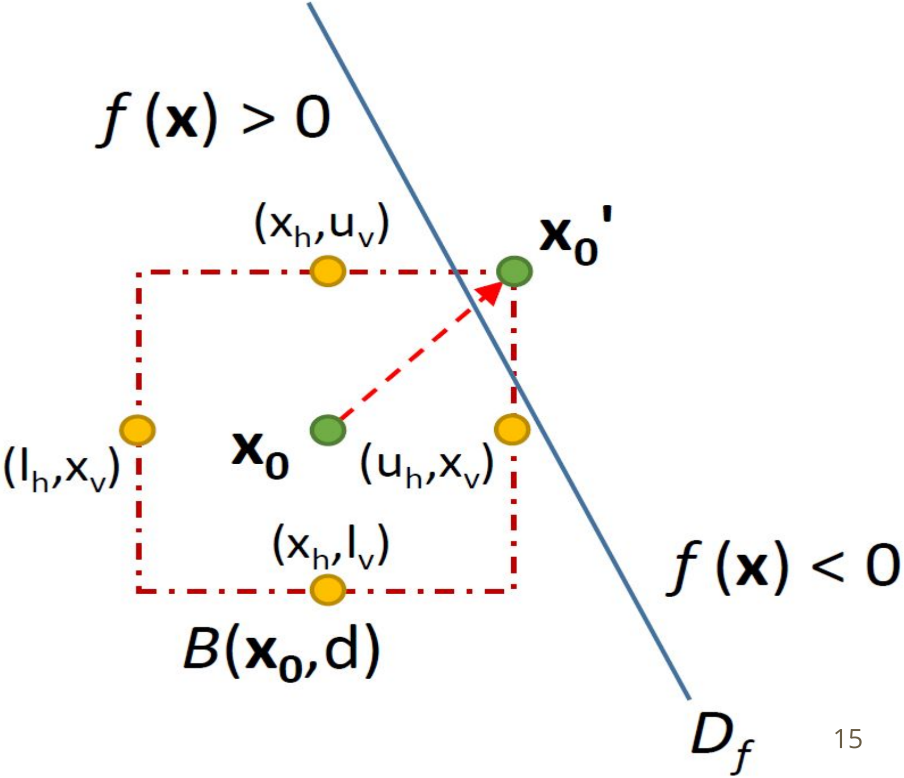


LInear Binary ▶️ Non-Linear Binary ▶️ Non-Linear Multi ▶️ Non-Linear Multi

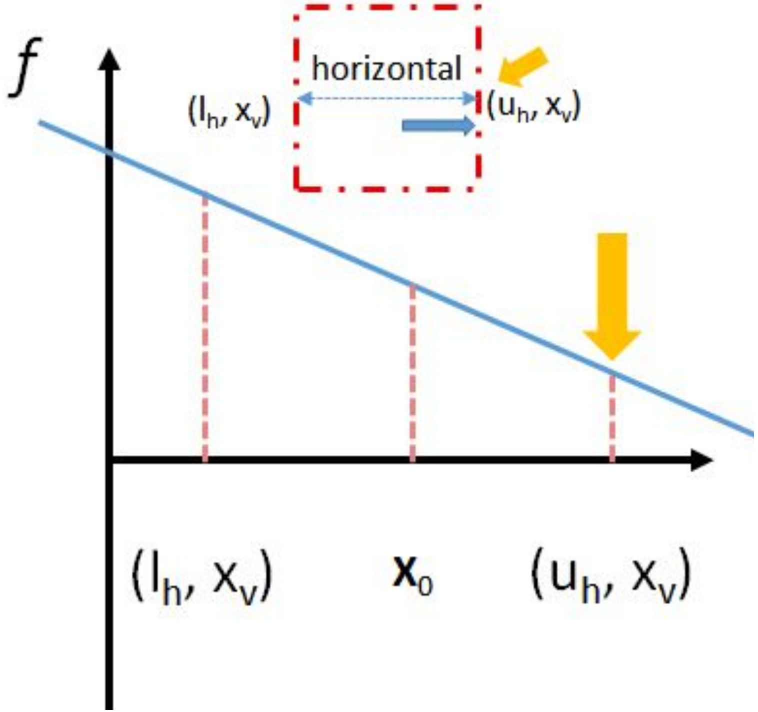
Simple → **General**

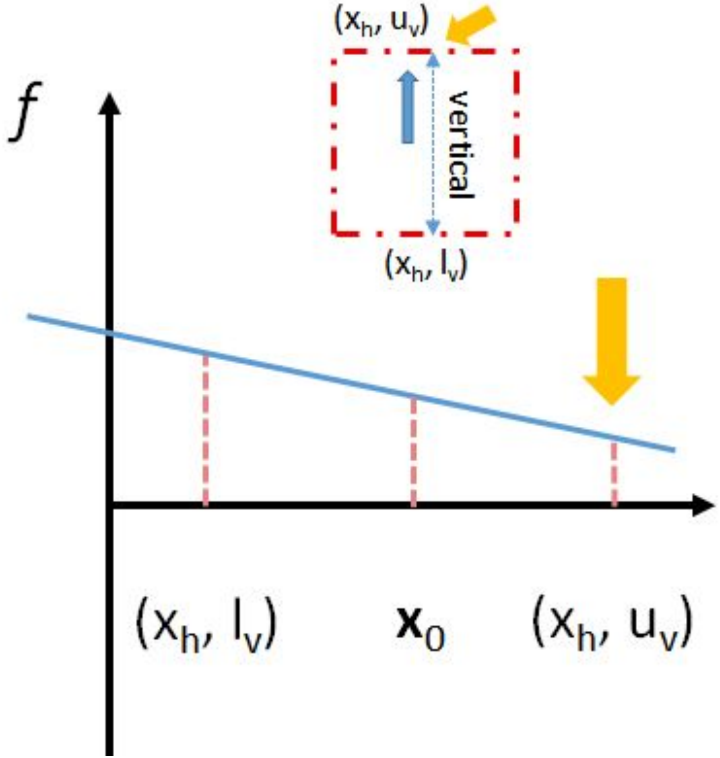
Algorithm: Basis - linear binary

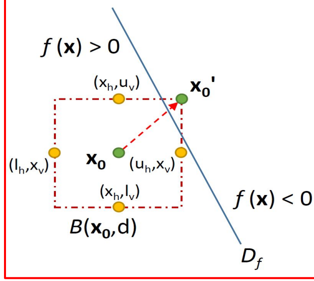
1. Classification Boundary f(input) = 0
2. Fuzz between upper bound and lower bound



Algorithm: Basis - linear binary



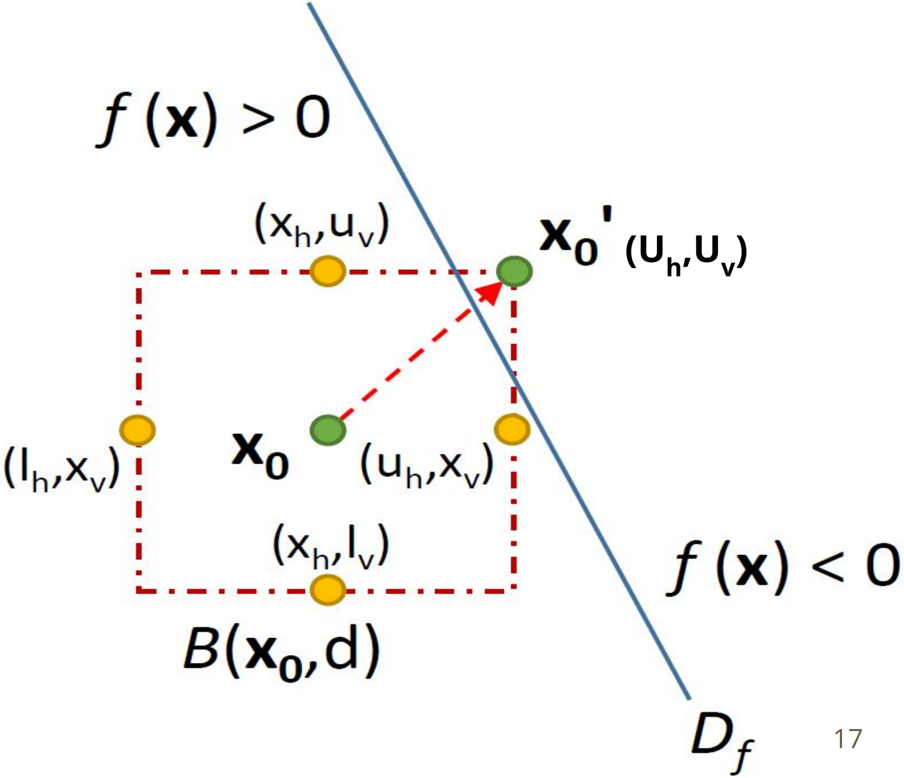




u_h and u_v selected

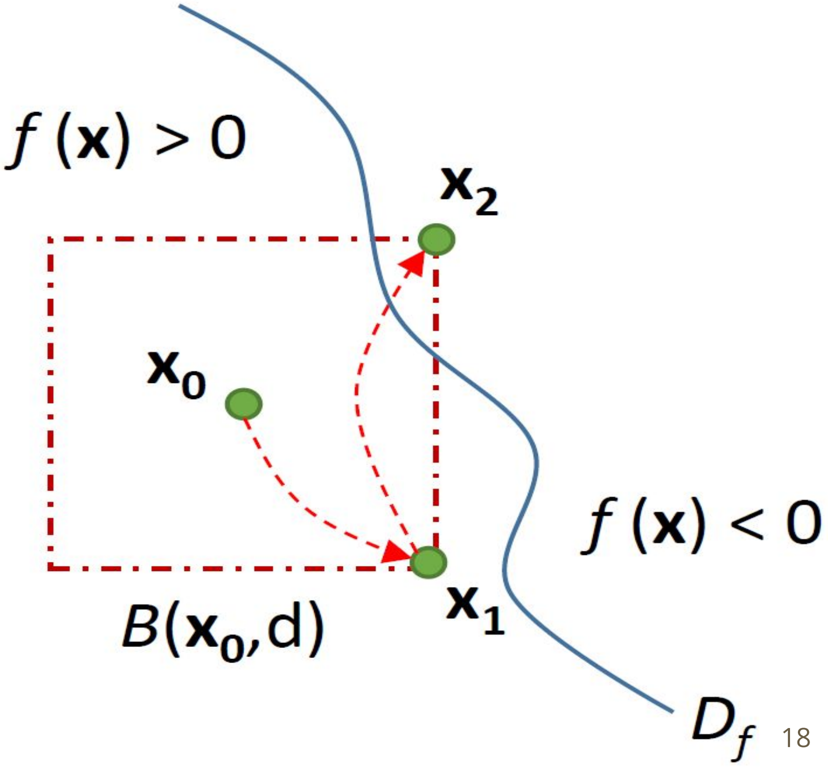
Algorithm: Basis - linear binary

1. Classification Boundary
   f(input) = 0
2. Fuzz between upper bound and lower bound

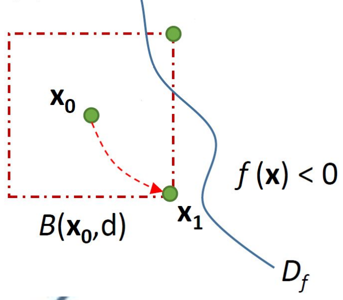


Algorithm: Approximation - nonlinear binary

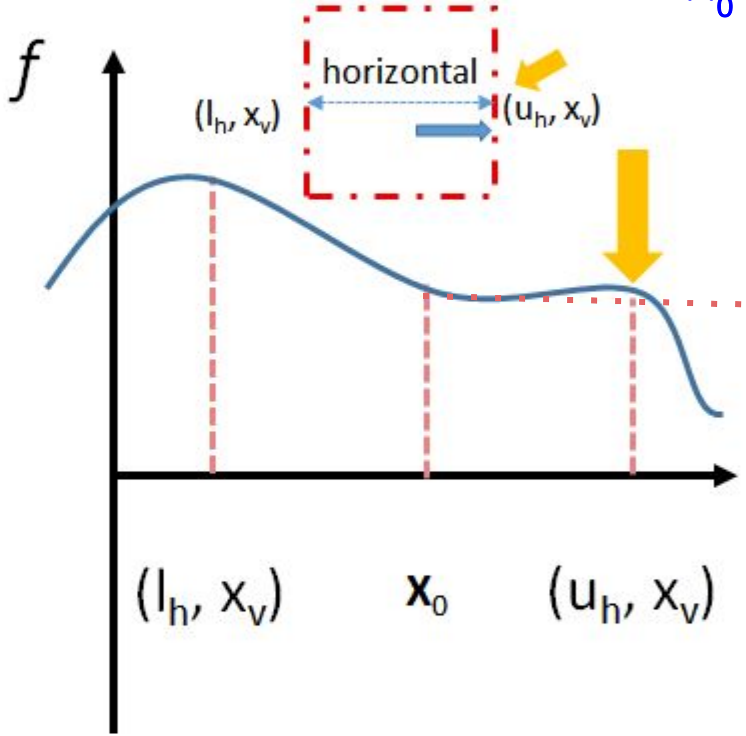
1. Same step as linear (Approximated)
2. Need for iteration (Minimum not guaranteed)
3. Still within original boundary

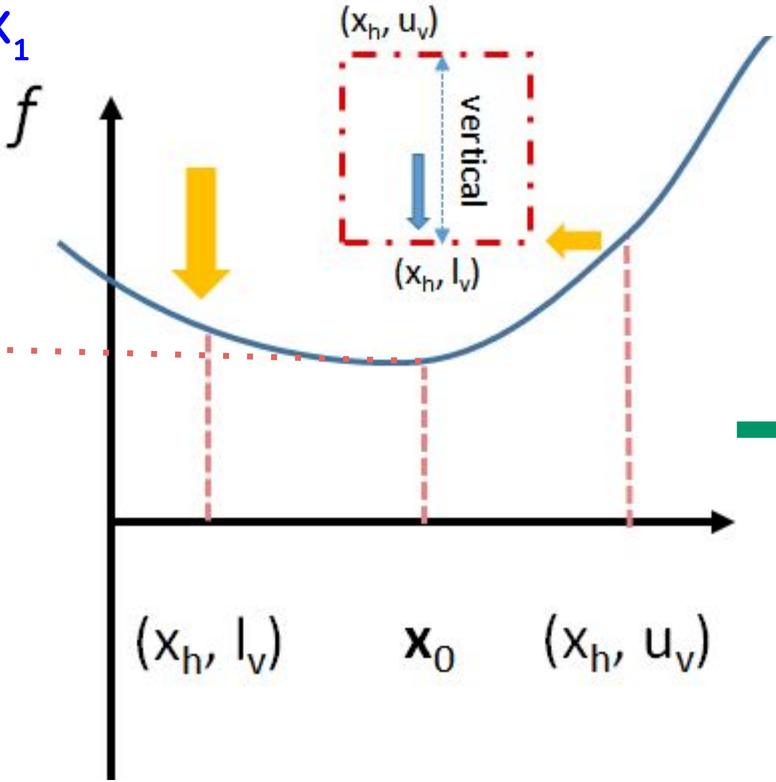


Algorithm: Approximation - nonlinear binary

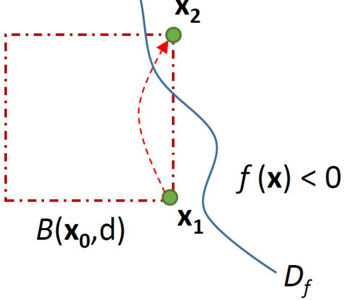


$x_0 \to x_1$



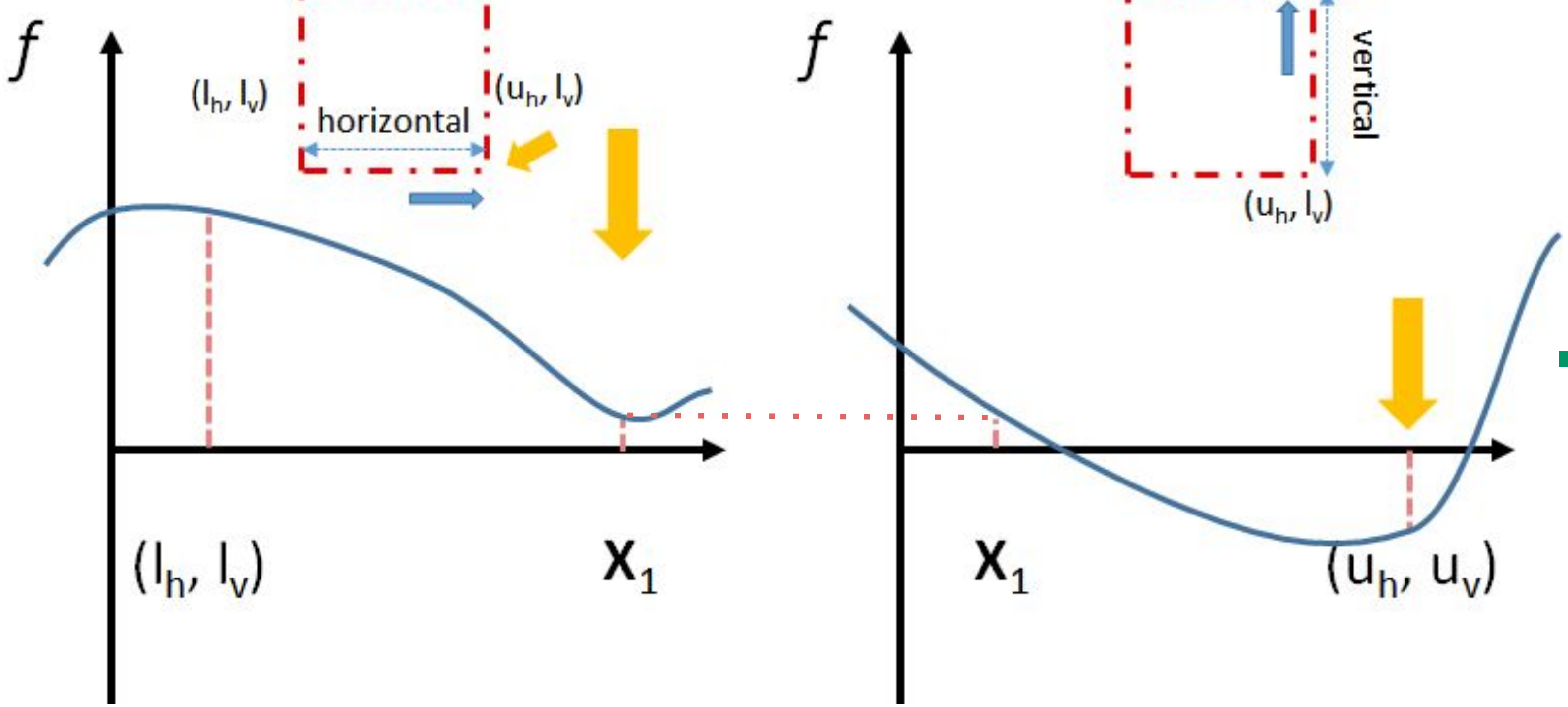


u_h and l_v selected



Algorithm: Approximation - nonlinear binary

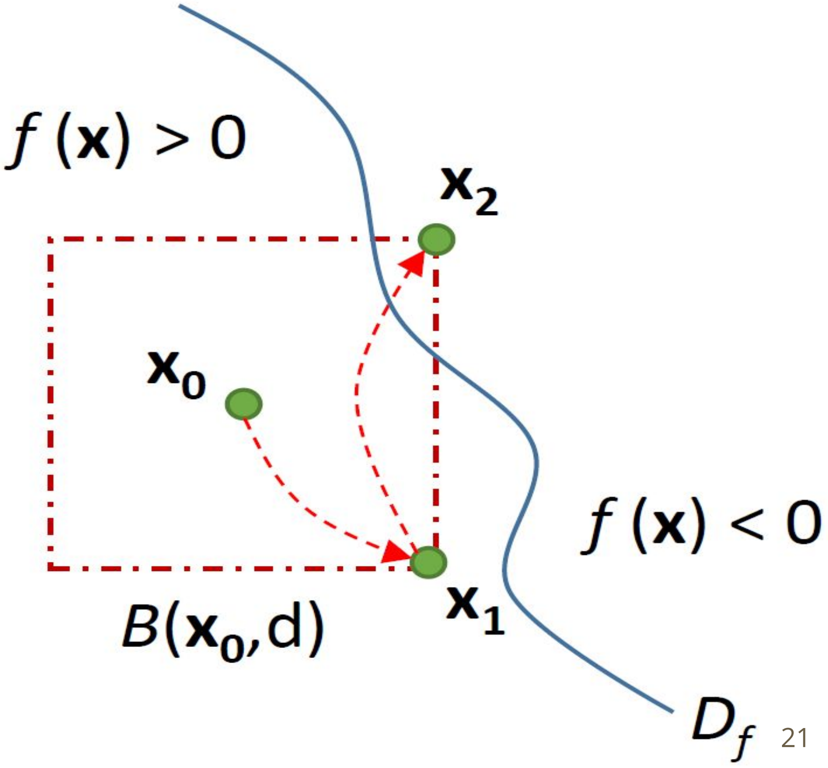
$X_1 \to X_2$



u_h and u_v selected

Algorithm: Approximation - nonlinear binary

1. Same step as linear (Approximated)
2. Need for iteration (Minimum not guaranteed)
3. Still within original boundary



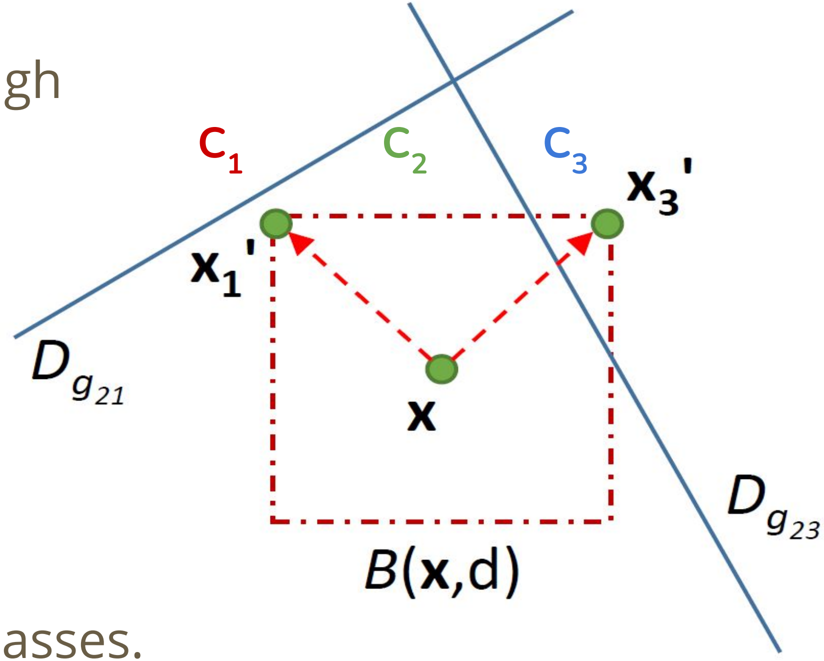
Algorithm: Multiclass - linear multiclass

1. Crossing one classifier is enough

2. Change in classifier function

   Classifier(f(input))
   to
   Classifier(f1(i), f2(i), ...)

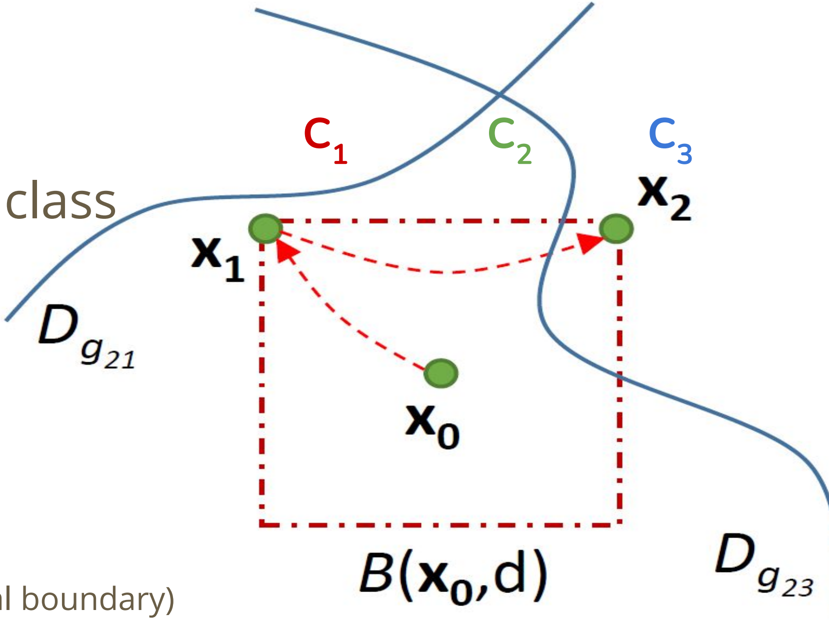
   Determined relative to each classes.



Algorithm: Changing classifiers targets

- nonlinear multiclass

1. Finalized generalization
2. Changing class target
   Lock on to the most probable class



Sum-up:

- Compare fitness at boundaries
- Choose closer one to adversarial
- Iterate until adversarial (within original boundary)

Algorithm: Back to practical (from simple example)

Illustration shows only for 2-pixel grayscale image.
(Two axis of modification)
This is the general form for **n**-pixel images with **j** classes. (**k** iterations)

Core of the algorithm

repeat

// for the classifier j that is most probable

$$r := \arg\min_j g_{ij}(\mathbf{x}_k)$$

$$\mathbf{x}_{k+1} := \text{ApproxMin}(\mathbf{x}_k, g_{ir}, (I_1, \ldots, I_n))$$

$$k := k + 1$$

until $\mathcal{N}_f(\mathbf{x}) \neq \mathcal{N}_f(\mathbf{x}_k)$, or $k = \text{MaxNum}$

'i' here is for current class and isn't related to the 'i' in the code on the right

I_n is range pixel_n can change in i.e, [lower_n, upper_n]

Function $\text{ApproxMin}(\mathbf{x}, f, (I_1, \ldots, I_n))$ **is**

$\mathbf{x}' := (0, \ldots, 0)$

foreach $1 \leq i \leq n$ **do**

if $f(\mathbf{x}[u_i/x_i]) > f(\mathbf{x}[l_i/x_i])$ **then**

$\mathbf{x}' := \mathbf{x}'[l_i/x'_i]$

else

$\mathbf{x}' := \mathbf{x}'[u_i/x'_i]$

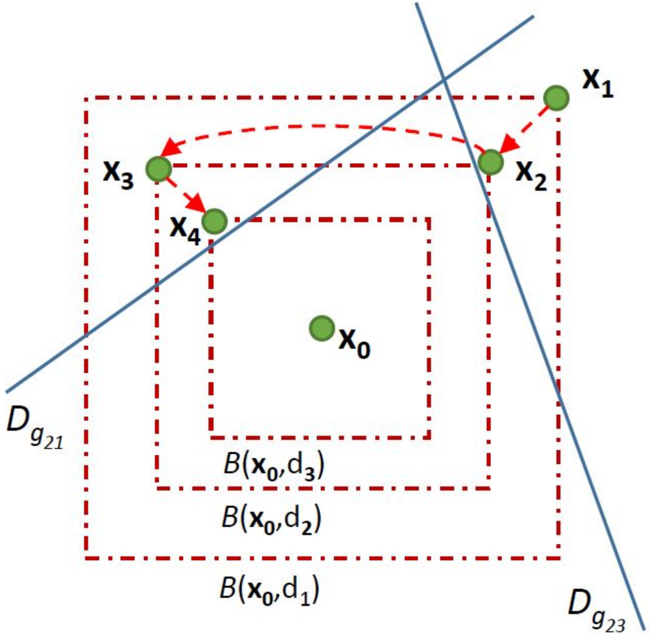
return $\mathbf{x}'$

Iterative refinement (Distortion reduction)

Why: We want the adversarial attack to be subtle, meaning the adversarial examples are close to the original input.

How: Keep trying to reduce the step size while the output is still misclassified.

(Note that we use L∞ distance: Similarity is measured by the maximum difference of pixel value)



Iterative refinement (Distortion reduction)

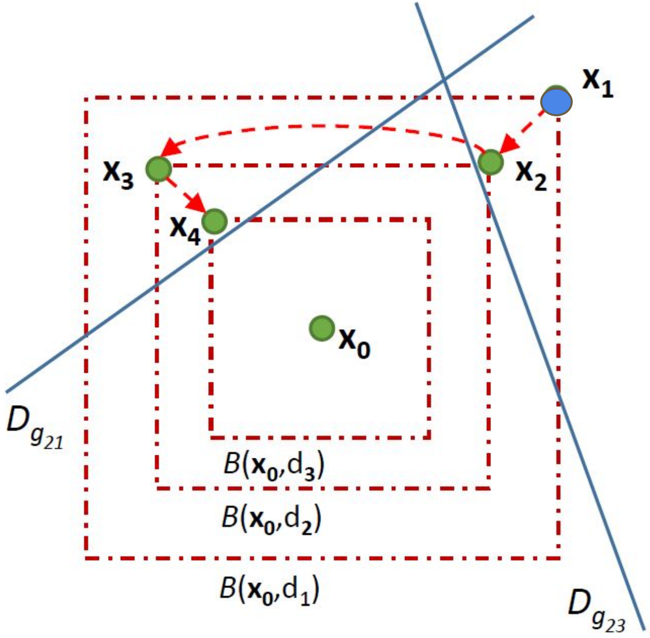
Algorithm 3: DeepSearch with iterative refinement.

Input: input $\mathbf{x} \in \mathbb{R}^n$, adversarial input $\mathbf{x}' \in \mathcal{B}(\mathbf{x}, d)$ $(d \in \mathbb{R})$, function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$

Output: an adversarial input $\mathbf{x}'' \in \mathcal{B}(\mathbf{x}, d')$ $(d' \leq d)$

Function `DS-Refinement`$(\mathbf{x}, \mathbf{x}', f)$ **is**
- **repeat**
  - $d := ||\mathbf{x} - \mathbf{x}'||_{L_\infty}$
  - apply bisect search to find the smallest distance $d' \leq d$ such that input $\text{Proj}(\mathcal{B}(\mathbf{x}, d'), \mathbf{x}')$ is an adversarial example.
  - choose an $\mathbf{x}_{new} \in \mathcal{B}(\mathbf{x}, d')$, from which to start a new search, e.g. $\mathbf{x}_{new} = \text{Proj}(\mathcal{B}(\mathbf{x}, d'), \mathbf{x}')$.
  - **if** f is binary **then**
    - $\mathbf{x}'' :=$ `DS-Binary`$(\mathbf{x}, \mathbf{x}_{new}, f, d')$
  - **else**
    - $\mathbf{x}'' :=$ `DS-Multiclass`$(\mathbf{x}, \mathbf{x}_{new}, f, d')$
  - **if** $\mathbf{x}''$ is an adversarial example **then**
    - $\mathbf{x}' := \mathbf{x}''$
  - **else**
    - $\mathbf{x}' := \text{Proj}(\mathcal{B}(\mathbf{x}, d'), \mathbf{x}')$
- **until** $\mathbf{x}''$ is not an adversarial example
- **return** $\mathbf{x}'$ and d'

Start



Iterative refinement (Distortion reduction)

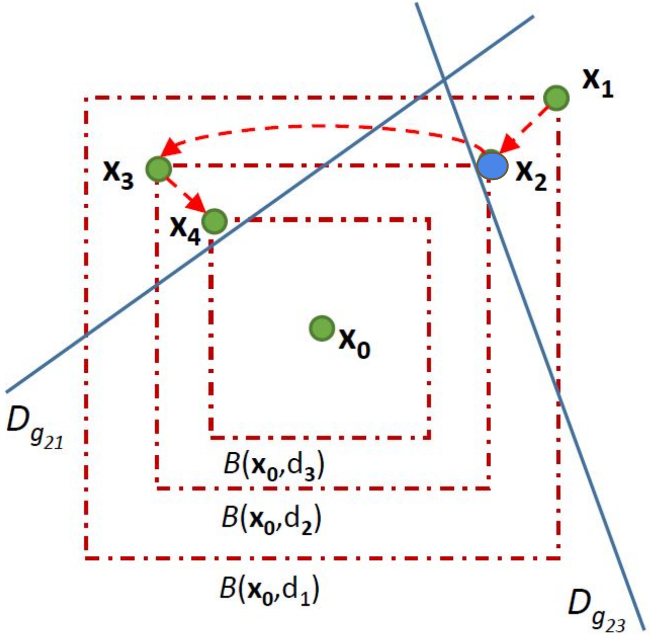
Algorithm 3: DeepSearch with iterative refinement.

Input: input $\mathbf{x} \in \mathbb{R}^n$, adversarial input $\mathbf{x}' \in \mathcal{B}(\mathbf{x}, d)$ $(d \in \mathbb{R})$,
function $f : \mathbb{R}^n \to \mathbb{R}^m$

Output: an adversarial input $\mathbf{x}'' \in \mathcal{B}(\mathbf{x}, d')$ $(d' \leq d)$

Function `DS-Refinement`($\mathbf{x}, \mathbf{x}', f$) **is**
- **repeat**
  - $d := ||\mathbf{x} - \mathbf{x}'||_{L_\infty}$
  - apply bisect search to find the smallest distance $d' \leq d$ such that input $\textsc{Proj}(\mathcal{B}(\mathbf{x}, d'), \mathbf{x}')$ is an adversarial example.
  - choose an $\mathbf{x}_{new} \in \mathcal{B}(\mathbf{x}, d')$, from which to start a new search, e.g. $\mathbf{x}_{new} = \textsc{Proj}(\mathcal{B}(\mathbf{x}, d'), \mathbf{x}')$.
  - **if** f is binary **then**
    - $\mathbf{x}'' :=$ `DS-Binary`($\mathbf{x}, \mathbf{x}_{new}, f, d'$)
  - **else**
    - $\mathbf{x}'' :=$ `DS-Multiclass`($\mathbf{x}, \mathbf{x}_{new}, f, d'$)
  - **if** $\mathbf{x}''$ is an adversarial example **then**
    - $\mathbf{x}' := \mathbf{x}''$
  - **else**
    - $\mathbf{x}' := \textsc{Proj}(\mathcal{B}(\mathbf{x}, d'), \mathbf{x}')$
- **until** $\mathbf{x}''$ is not an adversarial example
- **return** $\mathbf{x}'$ and d'

Refine



Iterative refinement (Distortion reduction)

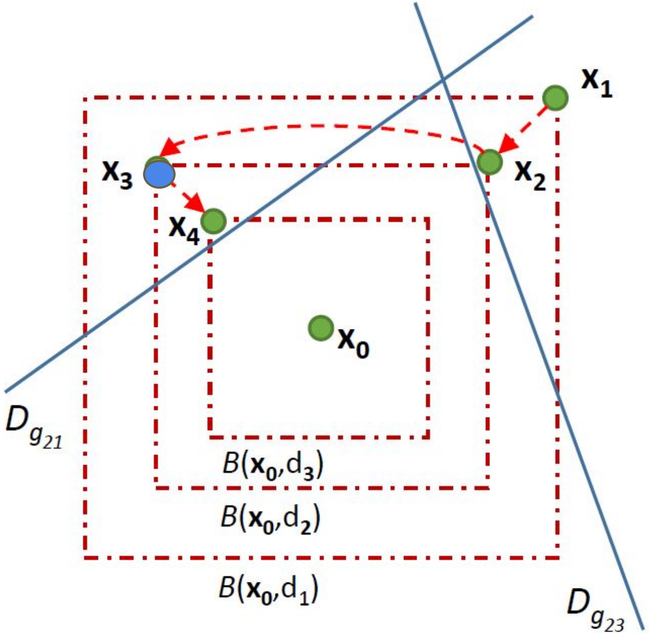
Algorithm 3: DeepSearch with iterative refinement.

Input: input $\mathbf{x} \in \mathbb{R}^n$, adversarial input $\mathbf{x}' \in \mathcal{B}(\mathbf{x}, d)$ $(d \in \mathbb{R})$,
function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$

Output: an adversarial input $\mathbf{x}'' \in \mathcal{B}(\mathbf{x}, d')$ $(d' \leq d)$

Function `DS-Refinement`$(\mathbf{x}, \mathbf{x}', f)$ **is**
- **repeat**
  - $d := ||\mathbf{x} - \mathbf{x}'||_{L_\infty}$
  - apply bisect search to find the smallest distance $d' \leq d$ such that input $\text{Proj}(\mathcal{B}(\mathbf{x}, d'), \mathbf{x}')$ is an adversarial example.
  - choose an $\mathbf{x}_{new} \in \mathcal{B}(\mathbf{x}, d')$, from which to start a new search, e.g. $\mathbf{x}_{new} = \text{Proj}(\mathcal{B}(\mathbf{x}, d'), \mathbf{x}')$.
  - **if** f is binary **then**
    - $\mathbf{x}'' :=$ `DS-Binary`$(\mathbf{x}, \mathbf{x}_{new}, f, d')$
  - **else**
    - $\mathbf{x}'' :=$ `DS-Multiclass`$(\mathbf{x}, \mathbf{x}_{new}, f, d')$
  - **if** $\mathbf{x}''$ is an adversarial example **then**
    - $\mathbf{x}' := \mathbf{x}''$
  - **else**
    - $\mathbf{x}' := \text{Proj}(\mathcal{B}(\mathbf{x}, d'), \mathbf{x}')$
- **until** $\mathbf{x}''$ is not an adversarial example
- **return** $\mathbf{x}'$ and d'

Search



Iterative refinement (Distortion reduction)

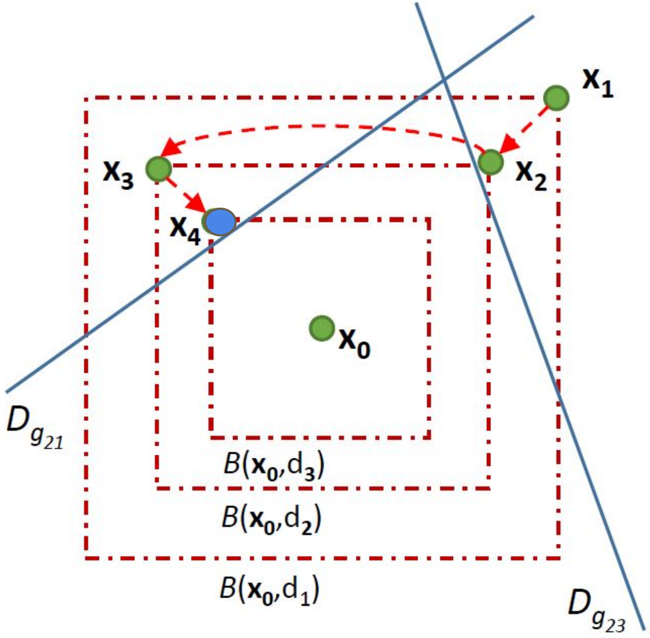
Algorithm 3: DeepSearch with iterative refinement.

Input: input $\mathbf{x} \in \mathbb{R}^n$, adversarial input $\mathbf{x}' \in \mathcal{B}(\mathbf{x}, d)$ $(d \in \mathbb{R})$, function $f : \mathbb{R}^n \to \mathbb{R}^m$

Output: an adversarial input $\mathbf{x}'' \in \mathcal{B}(\mathbf{x}, d')$ $(d' \leq d)$

Function `DS-Refinement`$(\mathbf{x}, \mathbf{x}', f)$ **is**
- **repeat**
  - $d := ||\mathbf{x} - \mathbf{x}'||_{L_\infty}$
  - apply bisect search to find the smallest distance $d' \leq d$ such that input $\text{Proj}(\mathcal{B}(\mathbf{x}, d'), \mathbf{x}')$ is an adversarial example.
  - choose an $\mathbf{x}_{new} \in \mathcal{B}(\mathbf{x}, d')$, from which to start a new search, e.g. $\mathbf{x}_{new} = \text{Proj}(\mathcal{B}(\mathbf{x}, d'), \mathbf{x}')$.
  - **if** f is binary **then**
    - $\mathbf{x}'' :=$ `DS-Binary`$(\mathbf{x}, \mathbf{x}_{new}, f, d')$
  - **else**
    - $\mathbf{x}'' :=$ `DS-Multiclass`$(\mathbf{x}, \mathbf{x}_{new}, f, d')$
  - **if** $\mathbf{x}''$ is an adversarial example **then**
    - $\mathbf{x}' := \mathbf{x}''$
  - **else**
    - $\mathbf{x}' := \text{Proj}(\mathcal{B}(\mathbf{x}, d'), \mathbf{x}')$
- **until** $\mathbf{x}''$ is not an adversarial example
- **return** $\mathbf{x}'$ and d'

Refine



Iterative refinement (Distortion reduction)

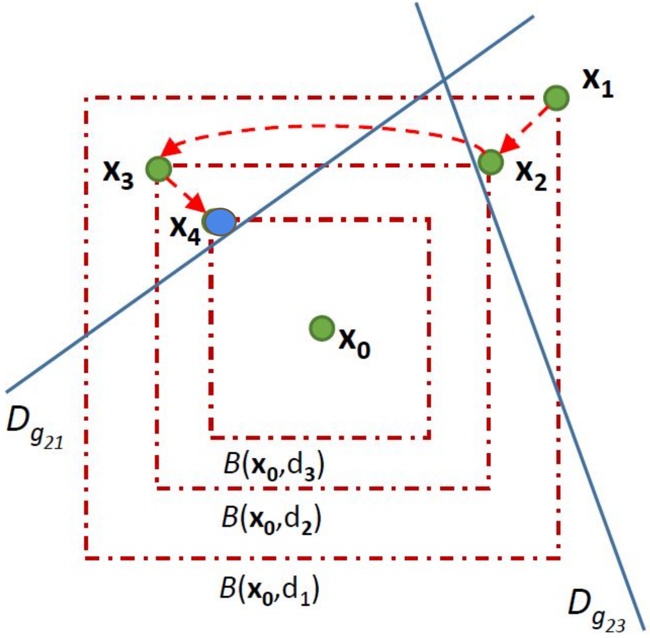
Algorithm 3: DeepSearch with iterative refinement.

Input: input $\mathbf{x} \in \mathbb{R}^n$, adversarial input $\mathbf{x}' \in \mathcal{B}(\mathbf{x}, d)$ $(d \in \mathbb{R})$,
function $f : \mathbb{R}^n \to \mathbb{R}^m$

Output: an adversarial input $\mathbf{x}'' \in \mathcal{B}(\mathbf{x}, d')$ $(d' \leq d)$

Function `DS-Refinement`$(\mathbf{x}, \mathbf{x}', f)$ **is**
- **repeat**
  - $d := ||\mathbf{x} - \mathbf{x}'||_{L_\infty}$
  - apply bisect search to find the smallest distance $d' \leq d$ such that input $\textsc{Proj}(\mathcal{B}(\mathbf{x}, d'), \mathbf{x}')$ is an adversarial example.
  - choose an $\mathbf{x}_{new} \in \mathcal{B}(\mathbf{x}, d')$, from which to start a new search, e.g. $\mathbf{x}_{new} = \textsc{Proj}(\mathcal{B}(\mathbf{x}, d'), \mathbf{x}')$.
  - **if** f is binary **then**
    - $\mathbf{x}'' :=$ `DS-Binary`$(\mathbf{x}, \mathbf{x}_{new}, f, d')$
  - **else**
    - $\mathbf{x}'' :=$ `DS-Multiclass`$(\mathbf{x}, \mathbf{x}_{new}, f, d')$
  - **if** $\mathbf{x}''$ is an adversarial example **then**
    - $\mathbf{x}' := \mathbf{x}''$
  - **else**
    - $\mathbf{x}' := \textsc{Proj}(\mathcal{B}(\mathbf{x}, d'), \mathbf{x}')$
- **until** $\mathbf{x}''$ is not an adversarial example
- **return** $\mathbf{x}'$ and d'

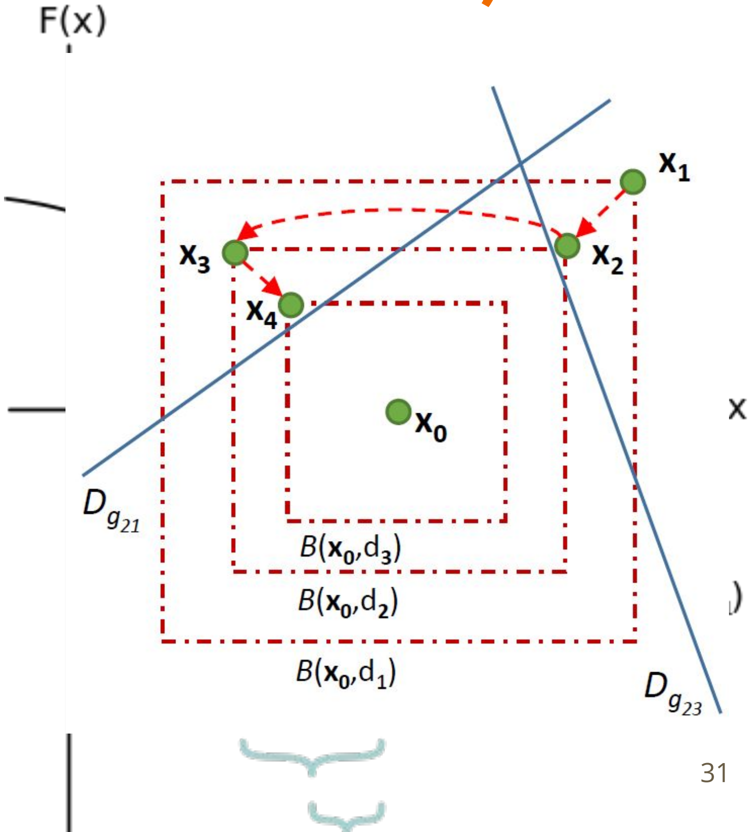
Search



Iterative refinement (Distortion reduction)

Refinement: Bisect search (binary search)

DeepSearch



Hierarchical grouping (Query reduction)

- Divide pixels of an input image in groups
- Mutate all pixels of a group in the same direction

Why? To reduce # of queries → 1 query/group

Hierarchical grouping (Query reduction)

1. *Initial grouping* → n/k² groups (k x k)

```
Function DS-Hierarchy(x, x_init, f, k, m) is
    G := Initial-Group({1, ..., n}, k) and x' := x_init
    repeat
        x' := DeepSearch(x, x', f, d, G)
        if 1 < k/m then
            G := Divide-Group(G, m)
            k := k/m
    until N_f(x) ≠ N_f(x'), or reached query budget L
    return x'
```

Hierarchical grouping (Query reduction)

2. Fuzzing → mutate all pixels in a group in the same direction

Function DS-Hierarchy($\mathbf{x}, \mathbf{x}_{init}, f, k, m$) **is**

  $\mathcal{G}$:= Initial-Group($\{1, \ldots, n\}, k$) and $\mathbf{x}' := \mathbf{x}_{init}$

  repeat

    $\boxed{\mathbf{x}' := \texttt{DeepSearch}(\mathbf{x}, \mathbf{x}', f, d, \mathcal{G})}$

    if $1 < k/m$ **then**

      $\mathcal{G}$:= Divide-Group($\mathcal{G}, m$)

      $k := k/m$

  until $\mathcal{N}_f(\mathbf{x}) \neq \mathcal{N}_f(\mathbf{x}')$, *or* reached query budget L

  return $\mathbf{x}'$

Hierarchical grouping (Query reduction)

3. Group splitting → if no adversarial examples then (k/m x k/m)

```
Function DS-Hierarchy(x, x_init, f, k, m) is
    G := Initial-Group({1, ..., n}, k) and x' := x_init
    repeat
        x' := DeepSearch(x, x', f, d, G)
        if 1 < k / m then
            G := Divide-Group(G, m)
            k := k / m
    until N_f(x) ≠ N_f(x'), or reached query budget L
    return x'
```

Evaluation

Experimental Evaluation

- Is DeepSearch effective in finding adversarial examples?
- Is DeepSearch effective in finding adversarial examples with low distortion?
- Is DeepSearch a query-efficient blackbox attack?
- Is the hierarchical grouping of DeepSearch effective in improving query efficiency?

Evaluation Setup

- **Datasets and network models**
  - Datasets: SVHN, CIFAR-10, ImageNet
  - Networks: ResNet w32-10(SVHN, CIFAR-10), Inception v3(ImageNet)
- **Existing approaches**
  - NES attack (QL-NES)
  - The Bandits attack
  - the Simple BlackBox Attack (SimBA)
  - Parsimonious blackbox attack

Metrics

- Success rate
  - Percentage of inputs for which effective adversarial examples are found
- Average distortion rate
  - Distance between original and adversarial input
- Average number of queries
  - Number of queries made to the network (because making queries might be costly, efficient approach should make less queries)

Results

Table 1: Results on SVHN networks.

Attack	Success rate	Avg. L_∞	Avg. L_2	Avg. queries	Med. queries
		Undefended network			
QL-NES	62.4%	2.58%	1.80%	2157	1700
Bandits	99.2%	3.43%	2.69%	762	573
SimBA	84.7%	4.65%	3.47%	1675	1430
Parsimonious	100%	4.59%	7.63%	337	231
DeepSearch	**100%**	**1.89%**	**3.17%**	**229**	**196**
		Defended network			
QL-NES	40.5%	4.10%	4.19%	5574	3900
Bandits	55.3%	4.38%	4.74%	2819	944
SimBA	65.9%	4.96%	3.95%	2687	2633
Parsimonious	78.9%	4.86%	8.08%	2174	423.5
DeepSearch	**83.1%**	**3.35%**	**5.58%**	**1808**	**458**

Table 2: Results on CIFAR-10 networks.

Attack	Success rate	Avg. L_∞	Avg. L_2	Avg. queries	Med. queries
		Undefended network			
QL-NES	52.8%	1.24%	0.99%	1360	1100
Bandits	92.6%	2.66%	2.34%	838	616
SimBA	71.6%	3.36%	2.19%	1311	1150
Parsimonious	100%	3.36%	6.36%	339	238.5
DeepSearch	**100%**	**1.64%**	**3.08%**	**247**	**196**
		Defended network			
QL-NES	30.1%	2.71%	3.09%	4408	3200
Bandits	39.2%	2.95%	4.39%	2952	1176
SimBA	41.2%	3.46%	4.50%	2425	2424
Parsimonious	47.4%	3.45%	6.61%	1228	366
DeepSearch	**47.7%**	**2.48%**	**4.70%**	**963**	**196**

Results

Table 3: Results on ImageNet undefended network.

Attack	Success rate	Avg. L_∞	Avg. L_2	Avg. queries	Med. queries
QL-NES	90.3%	1.83%	1.75%	2300	1800
Bandits	92.1%	2.15%	2.61%	930	496
SimBA	61%	3.15%	0.67%	4379	4103
Parsimonious	98.3%	3.16%	6.35%	660	241
DeepSearch	**99.3%**	**1.50%**	**3.05%**	**561**	**196**

Results

Table 4: Query reduction for SVHN and CIFAR-10. For each dataset, success rate (resp. average queries) is shown in the first (resp. second) row.

Dataset	1	2×2	4×4	8×8	16×16
Undefended network					
SVHN	100% 742	100% 300	100% 229	100% 238	100% 242
CIFAR-10	100% 462	100% 301	100% 247	100% 255	100% 259
Defended network					
SVHN	81.3% 3143	82.4% 2292	83.1% 1808	83.6% 1565	83.9% 1591
CIFAR-10	47.7% 2292	47.4% 1156	47.7% 963	47.6% 935	47.6% 946

Table 5: Query reduction for ImageNet. Success rate (resp. average queries) is shown in the first (resp. second) row.

Dataset	8×8	16×16	32×32	64×64	128×128
ImageNet	99.1% 765	99.1% 580	99.1% 533	99.4% 554	99.4% 580

Possible threats to validity

- Datasets and networks models
- Existing Approaches
- Fairness of comparison

Some comments

Pros:
- Simple
- Effective

Cons
- 'Block' grouping
- Requires probability output
- Non targeted attack

Thank you for your attention!