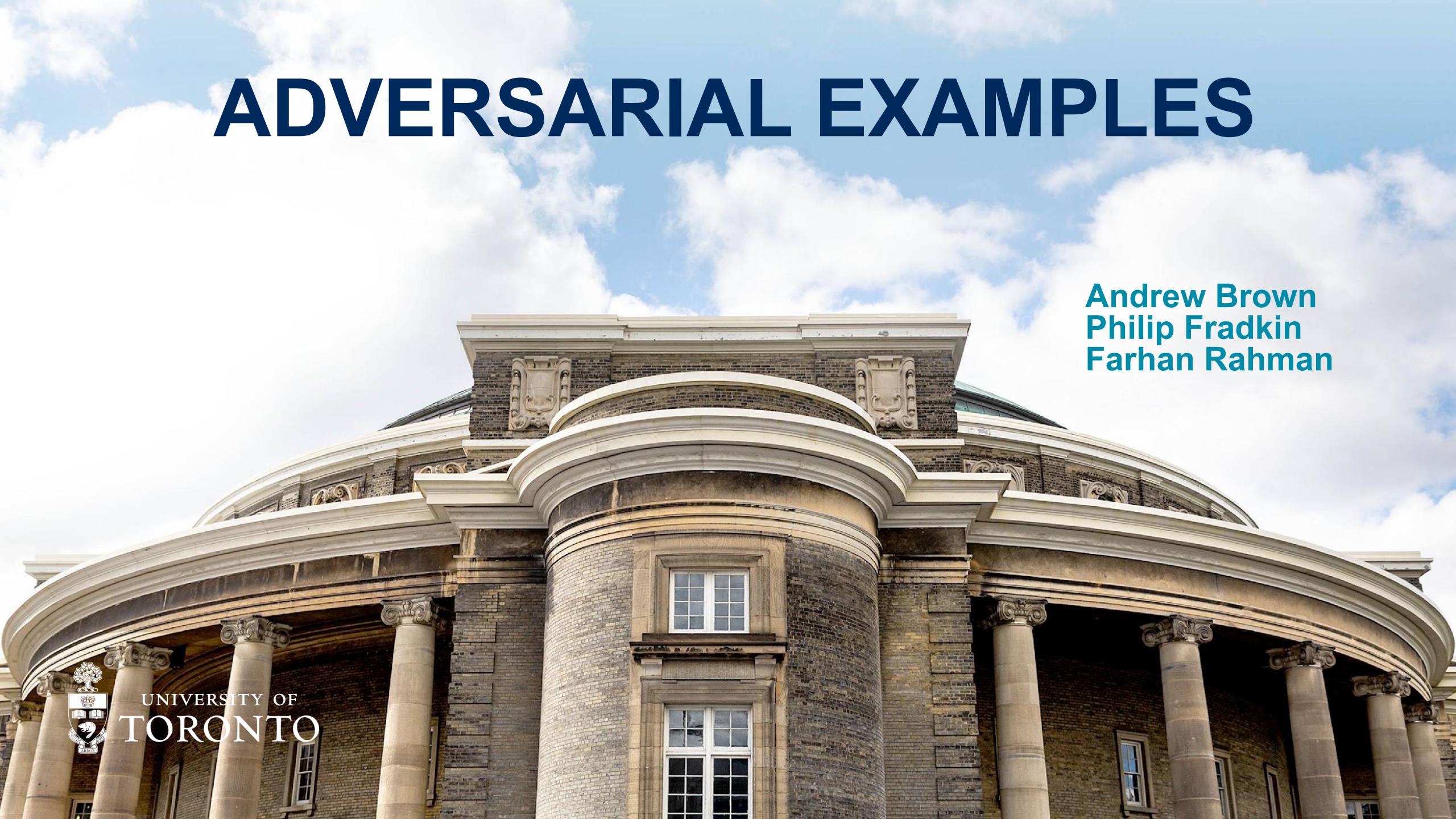
ADVERSARIAL EXAMPLES

Andrew Brown
Philip Fradkin
Farhan Rahman



Outline

1. Introduce attacks on neural networks and importance of the subject

2. Andrew is going to talk about: *Intriguing Properties of Neural Networks*

3. Farhan will talk about broader application of attacks: *Practical Black-Box Attacks against Machine Learning*

4. Phil is going to talk about defenses against attacks: *Certified Adversarial Robustness via Randomized Smoothing*

Why do we need to study adversarial attacks:

- Computer vision is used everywhere!
- Facial Recognition
- Self Driving Cars (change speed limit sign)
- Biometric Recognition
- Text Applications too! (Think CNNs for Sentiment Analysis or Text classification)
- Ad-blockers can incorrectly classified ads
- Spam Classifiers can incorrectly identify malicious mails

Classic Attack: Fast gradient sign method

$\boldsymbol{x}$

"panda"
57.7% confidence

$+ .007 \times$

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"nematode"
8.2% confidence

$=$

$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"gibbon"
99.3 % confidence

Extremely simple to implement and cheap to compute!

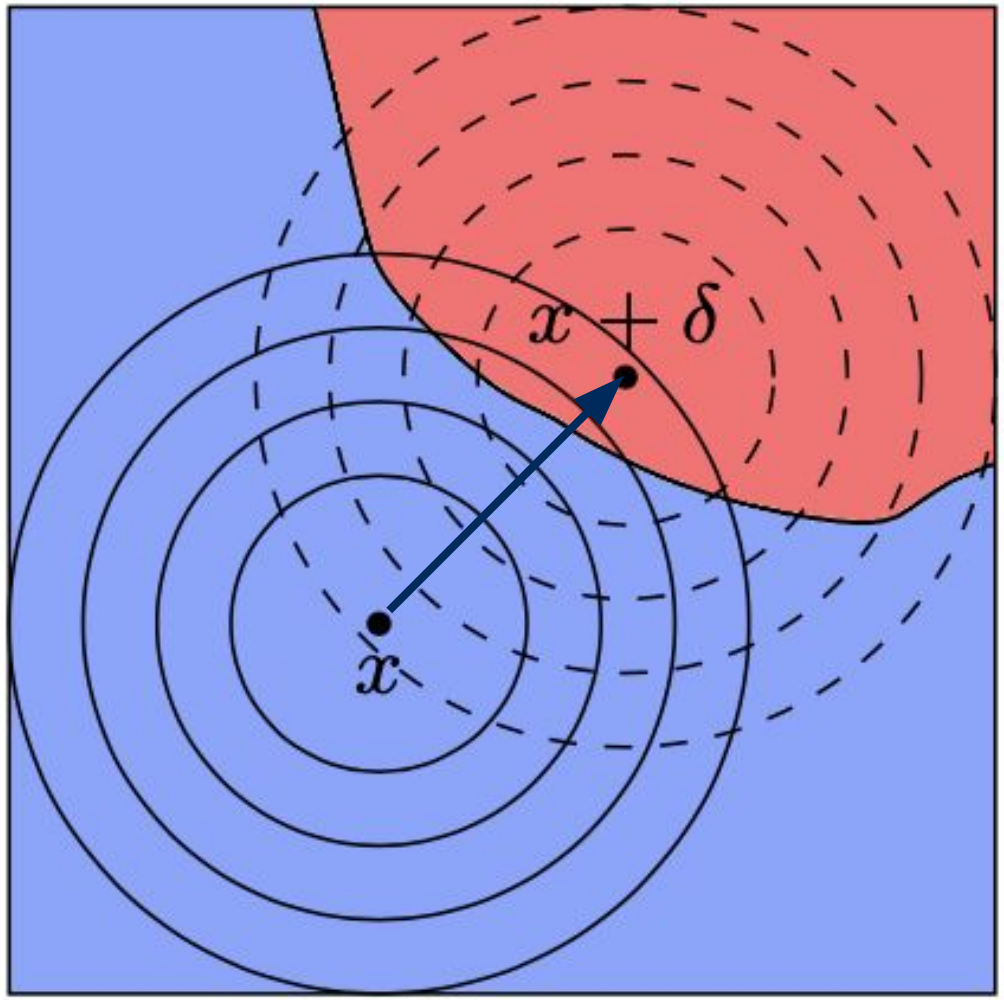
$$adv\_x = x + \epsilon * \text{sign}(\nabla_x J(\theta, x, y))$$

Maximizing the loss of the input image

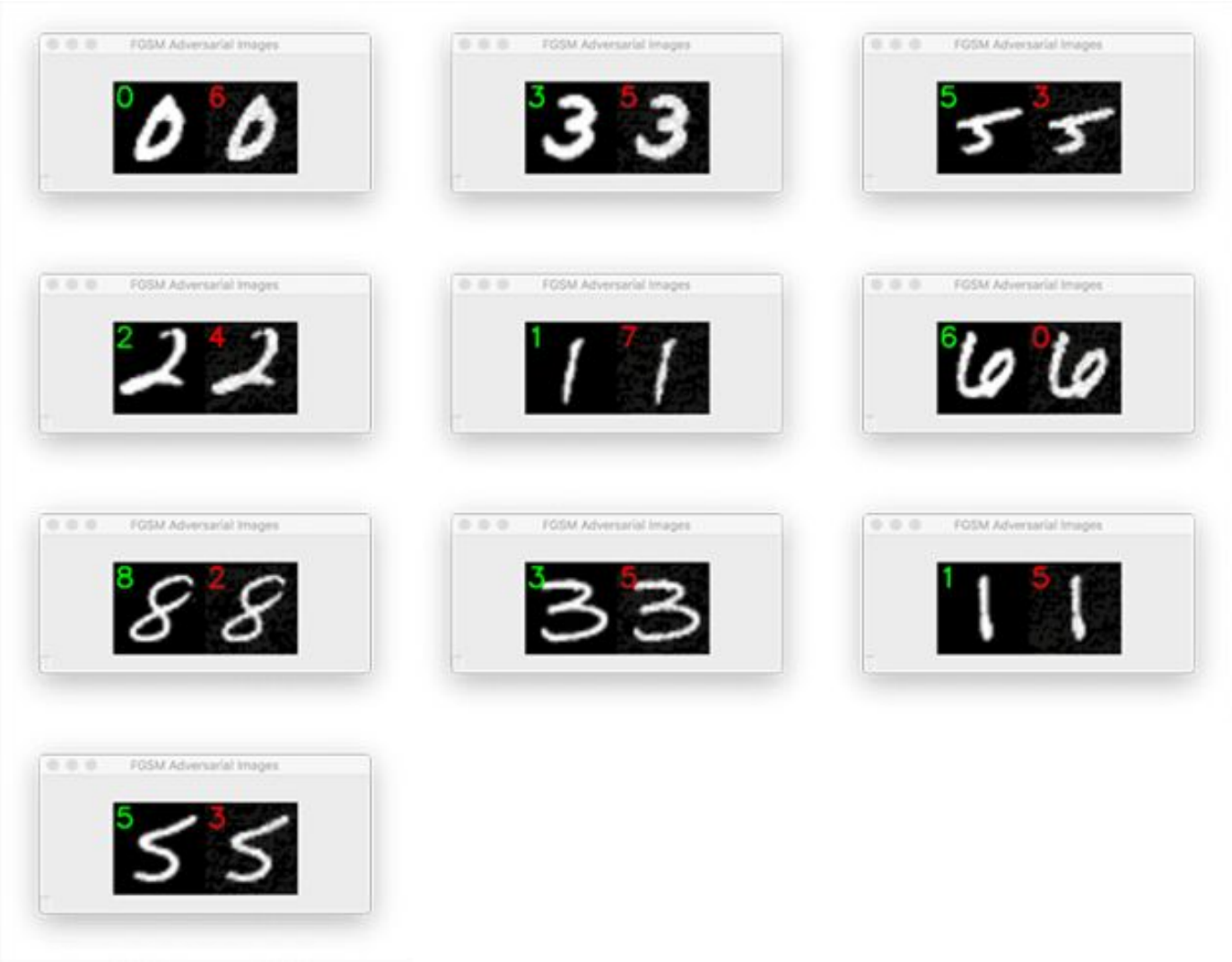
- x - original input
- Epsilon - tiny perturbation
- Sign: direction of gradient
- J - loss function computed over
  - θ - network
  - x input image
  - y - true label

Attack interpretation

- Movement within input space
- In FGSM determined by minimizing true class label probability
- The magnitude of the attack can be interpreted to be within a certain radius (an l2 ball)
- Trade Off between robustness against large l2 ball attacks vs classification accuracy



FGSM - changes indistinguishable to humans



Human adversarial examples

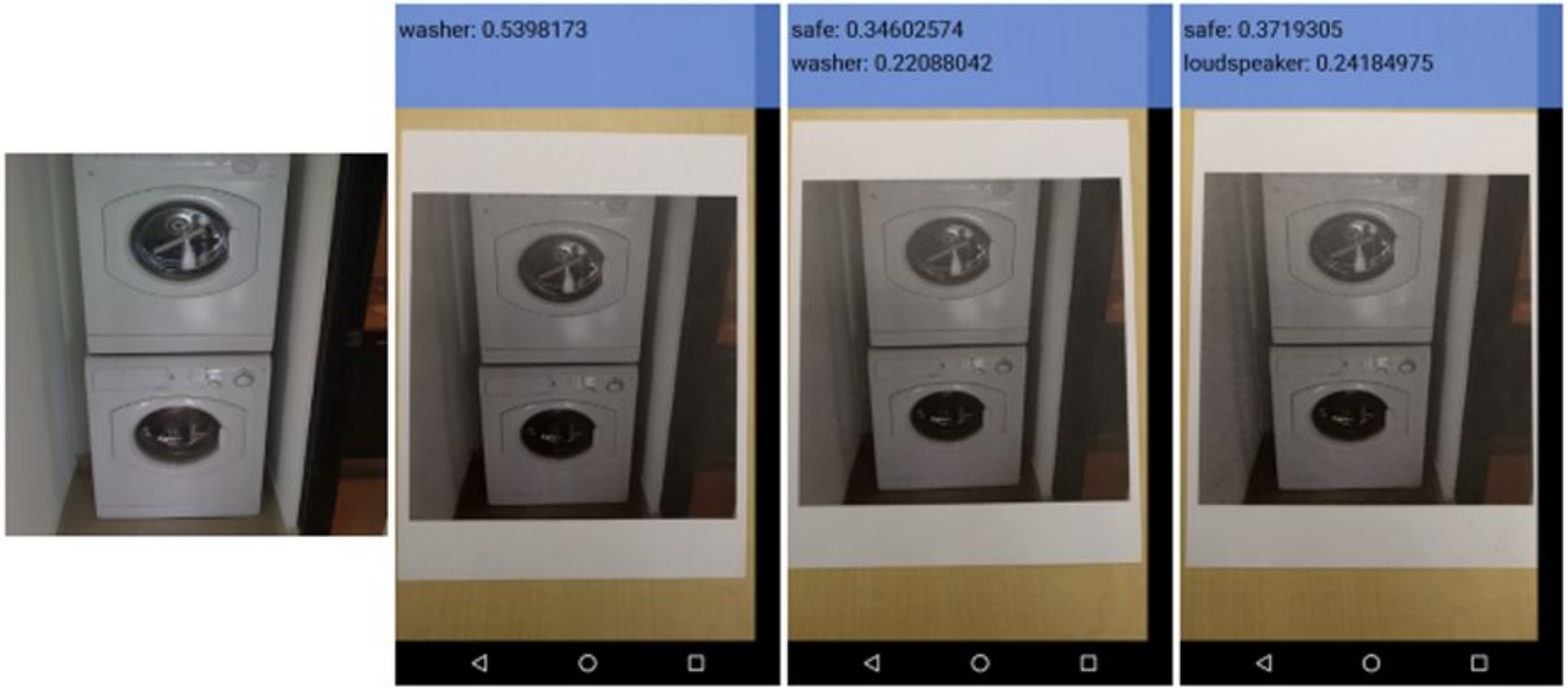
Human adversarial examples

Motivation for Adversarial Attacks

Adversaries have many reasons

- Fool real classifiers remotely hosted via API (Amazon, Google)
- Fool malware detection (spam/ham?)
- Show that machine learning systems are not robust

Adversarial Example in the Physical World



(a) Image from dataset (b) Clean image (c) Adv. image, $\epsilon = 4$ (d) Adv. image, $\epsilon = 8$

Kurakin et al, 2016

Impacts of Adversarial Examples

Many defences tried

- Adding noise at train time
- Dropout
- Ensembling
- Etc.

Made a new problem to be solved -> robustness to adversarial examples

The original youtube video got taken down 3 days before our presentation

🤔🤔🤔

Intriguing Properties of Neural Networks (Andrew Brown)

Christian Szegedy
Google Inc.

Wojciech Zaremba
New York University

Ilya Sutskever
Google Inc.

Joan Bruna
New York University

Dumitru Erhan
Google Inc.

Ian Goodfellow
University of Montreal

Rob Fergus
New York University
Facebook Inc.

Submitted in 2013, revised in 2014

Motivation

- Earlier papers suggest certain neurons in a DNN respond to certain features
  - This is **empirically** correct
- Original purpose was to perturb images in order to change classification
  - Different classification = different perceptual visual?

- We wish to investigate the two statements
- Context: A very early paper in Adversarial Example Research (2013/14)

Introduction

Authors find two properties of Deep Neural Networks (DNNs)

1. Semantic meaning within individual units and random linear combinations of high-level units
2. **Performance of DNNs after small non-random imperceptible perturbations**

A small non-random imperceptible perturbation = Adversarial Example!

Terminology

Unit - Neuron

$\phi(x)$ - activation values of some layer

Distortion - Adversarial Perturbation (measured as standard deviation of pixels)

FC - Fully connected NN with one or more hidden layers and a softmax classifier

AE - Classifier trained on top of autoencoder

AlexNet - Convolutional NN designed by Alex Krizhevsky

QuocNet - Unsupervised network with 1 billion learnable parameters

1. Semantics

1. Semantics

Claim: No distinction between individual high-level units and random linear combinations of high-level units

Experiment: Look at input images which maximize activation of units in some layer

$$x' = \arg\max_{x \in \mathcal{I}} \langle \phi(x), e_i \rangle$$

vs.

$$x' = \arg\max_{x \in \mathcal{I}} \langle \phi(x), v \rangle$$

Where, ei = vector associated with i-th unit (individual unit) and v = random vector

Compare **x` (images)** from individual units vs random linear combinations

1. Semantics Result

(a) Unit sensitive to white flowers.

(b) Unit sensitive to postures.

Images stimulating single neuron most

(a) Direction sensitive to white, spread flowers.

(b) Direction sensitive to white dogs.

Images stimulating a random linear combination of units most

Discussion - Semantics

Both image sets (and many more experiments) are semantically related, regardless of individual neurons or random linear combination of neurons

Is it incorrect to conclude neural networks disentangle images into features across ***individual*** neurons?

How is the network using features of the image?

- Non-local Generalization?
- Local generalization?

2. Adversarial Examples

2. Imperceptible Perturbations (Adversarial Examples)

Claim:

- Many classifiers will misclassify an image after applying an imperceptible perturbation
- The same perturbation can cause a different classifier trained on different data to misclassify the same input -> **transferability**
  - Different data still means from same distribution (eg. partitioning MNIST)
  - When discovered, the authors did not know what adversarial examples were yet
  - Not considered with any application (possible attacks!)

2. Adversarial Examples

Experiment:

- Minimize $\|r\|_2$ subject to:
  1. $f(x+r) = l$
  2. $x + r \in [0,1]^m$

An Optimization problem!

Where f = classifier, x = image, r = perturbation, and l = different label from original

Informally, x+r is the closest image to x that is classified as l by f

- Change the image as little as possible

2. Adversarial Examples

Experiment cont.: Minimizer is denoted as D(x, l)

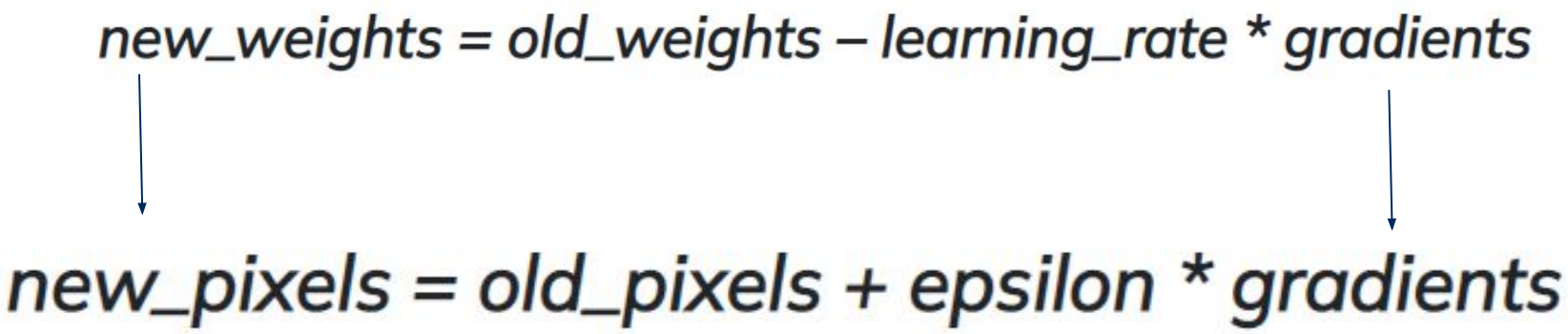
Formally stated,

$$\text{Minimize } c|r| + \text{loss}_f(x + r, l) \text{ subject to } x + r \in [0, 1]^m$$

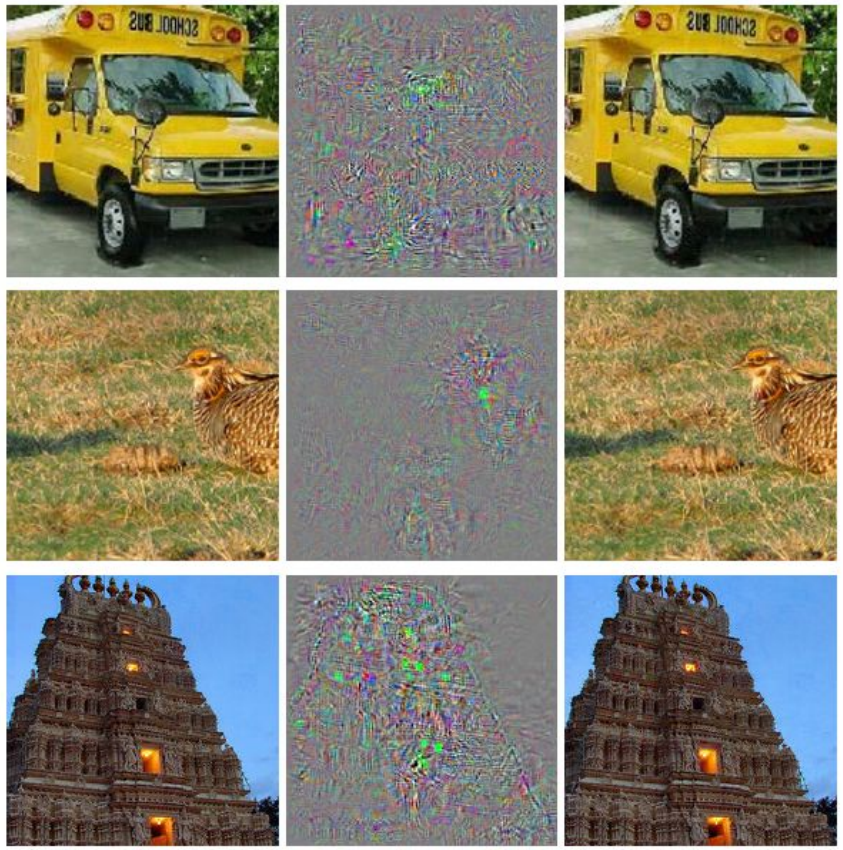
- Using line-search we find the minimum c > 0 s.t the classifier f classifies as target label l
- Now known as FGSM

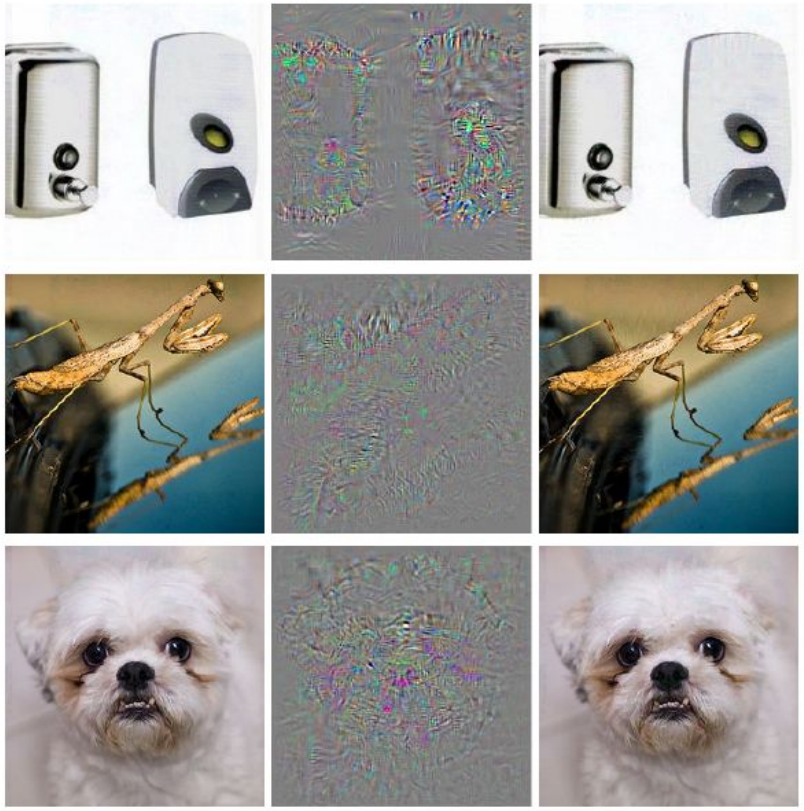
2. Adversarial Examples

FGSM Explanation



2. Adversarial Examples Results - AlexNet trained on ImageNet





2. Adversarial Examples Results - QuocNet (binary car classifier)

(a)

(b)

2. Adversarial Examples on MNIST

Model Name	Description	Training error	Test error	Av. min. distortion
FC10(10^{-4})	Softmax with $\lambda = 10^{-4}$	6.7%	7.4%	0.062
FC10(10^{-2})	Softmax with $\lambda = 10^{-2}$	10%	9.4%	0.1
FC10(1)	Softmax with $\lambda = 1$	21.2%	20%	0.14
FC100-100-10	Sigmoid network $\lambda = 10^{-5}, 10^{-5}, 10^{-6}$	0%	1.64%	0.058
FC200-200-10	Sigmoid network $\lambda = 10^{-5}, 10^{-5}, 10^{-6}$	0%	1.54%	0.065
AE400-10	Autoencoder with Softmax $\lambda = 10^{-6}$	0.57%	1.9%	0.086

Table 1: Tests of the generalization of adversarial instances on MNIST.

2. Adversarial Examples on MNIST

Model Name	Description	Training error	Test error	Av. min. distortion
FC10(10^{-4})	Softmax with $\lambda = 10^{-4}$	6.7%	7.4%	0.062
FC10(10^{-2})	Softmax with $\lambda = 10^{-2}$	10%	9.4%	0.1
FC10(1)	Softmax with $\lambda = 1$	21.2%	20%	0.14
FC100-100-10	Sigmoid network $\lambda = 10^{-5}, 10^{-5}, 10^{-6}$	0%	1.64%	0.058
FC200-200-10	Sigmoid network $\lambda = 10^{-5}, 10^{-5}, 10^{-6}$	0%	1.54%	0.065
AE400-10	Autoencoder with Softmax $\lambda = 10^{-6}$	0.57%	1.9%	0.086

Table 1: Tests of the generalization of adversarial instances on MNIST.

Observation: with higher λ (regularization) more minimum perturbation is required at a cost of higher training/testing error

2. Adversarial Examples - Transferability

Adversarial Examples transfer on same models = 100% (diagonal)
Cross-model generalization - Attacks transfer to different models **sometimes*

Transferred to these models

Original Models

	FC10(10^{-4})	FC10(10^{-2})	FC10(1)	FC100-100-10	FC200-200-10	AE400-10	Av. distortion
FC10(10^{-4})	100%	11.7%	22.7%	2%	3.9%	2.7%	0.062
FC10(10^{-2})	87.1%	100%	35.2%	35.9%	27.3%	9.8%	0.1
FC10(1)	71.9%	76.2%	100%	48.1%	47%	34.4%	0.14
FC100-100-10	28.9%	13.7%	21.1%	100%	6.6%	2%	0.058
FC200-200-10	38.2%	14%	23.8%	20.3%	100%	2.7%	0.065
AE400-10	23.4%	16%	24.8%	9.4%	6.6%	100%	0.086
Gaussian noise, stddev=0.1	5.0%	10.1%	18.3%	0%	0%	0.8%	0.1
Gaussian noise, stddev=0.3	15.6%	11.3%	22.7%	5%	4.3%	3.1%	0.3

**All trained on same MNIST data

2. Adversarial Examples - Transferability

Adversarial Examples transfer on same models = 100% (diagonal)
Cross-model generalization - Attacks transfer to different models **sometimes*

Transferred to these models

Original Models

	FC10(10^{-4})	FC10(10^{-2})	FC10(1)	FC100-100-10	FC200-200-10	AE400-10	Av. distortion
FC10(10^{-4})	100%	11.7%	22.7%	2%	3.9%	2.7%	0.062
FC10(10^{-2})	87.1%	100%	35.2%	35.9%	27.3%	9.8%	0.1
FC10(1)	71.9%	76.2%	100%	48.1%	47%	34.4%	0.14
FC100-100-10	28.9%	13.7%	21.1%	100%	6.6%	2%	0.058
FC200-200-10	38.2%	14%	23.8%	20.3%	100%	2.7%	0.065
AE400-10	23.4%	16%	24.8%	9.4%	6.6%	100%	0.086
Gaussian noise, stddev=0.1	5.0%	10.1%	18.3%	0%	0%	0.8%	0.1
Gaussian noise, stddev=0.3	15.6%	11.3%	22.7%	5%	4.3%	3.1%	0.3

**All trained on same MNIST data

2. Adversarial Example with Different Training Data

Partition MNIST into halves (P1 and P2) and train

		Transfer Adversarial Examples to these models		
		FC100-100-10	FC123-456-10	FC100-100-10'
Trained on P1	Distorted for FC100-100-10 (av. stddev=0.062)	100%	26.2%	5.9%
	Distorted for FC123-456-10 (av. stddev=0.059)	6.25%	100%	5.1%
Trained on P2	Distorted for FC100-100-10' (av. stddev=0.058)	8.2%	8.2%	100%
	Gaussian noise with stddev=0.06	2.2%	2.6%	2.4%

Discussion - Adversarial Examples

1. Box-Constrained L-BFGS can reliably find adversarial examples for DNNs
   - These adversarial examples are sometimes (ImageNet) indistinguishable to the human eye

2. Are these supervised models “learning?” -> Local generalization

3. Auto-Encoders perform better against transferability of adversarial examples, was this a fair experiment?

4. What is the cause of transferability? The same adversarial example is often misclassified by a variety of different classifiers

Criticisms of the Work

1. Activation of neurons for semantic meaning is only lightly touched
2. Should evaluate ‘likelihood’ of adversarial examples they generate
    a. Hypothesis: higher dimensionality = higher density of adversarial examples
    b. It seems intuitive that slight perturbations at the input layer can massively change activations at the final layer (in large architectures)
3. Authors should’ve compared other kinds of classifiers as a control (SVMs, Random Forest)
    a. Blind Spots could be a function of the data and not classifiers
    b. Especially other unsupervised techniques**

Future Work (in 2014)

1. How come Adversarial Examples transfer/generalize to other models (different architecture and training data)

2. The suggested cause of Adversarial Examples
   - Extreme Nonlinearity of DNNs?
   - Insufficient Model Averaging?
   - Insufficient regularization?

3. Auto-Encoders are more resilient to adversarial examples -> more unsupervised techniques?

4. Discovered adversarial examples, we need to define some attacks/defences!

Conclusion

- DNNs have counter-intuitive properties
  - Semantic meaning of individual units vs random linear combination of units
  - The existence of adversarial examples
- Adversarial Examples show DNNs are not robust
  - More research towards understanding the probability of an adversarial examples occurring

The follow up to this paper - Explaining and Harnessing Adversarial Examples (2015)

Practical Black-Box Attacks against Machine Learning

Papernot et al., 2016

Spot the difference?

Classified as Macaw

Classified as Dragonfly

How these attacks are carried out?

Previous works consider:

(1). Knowledge of the model architecture with parameters.

(2). Adversary able to collect surrogate dataset.

(3). Can modify inputs to observe the output.

Current Study:

(1). The adversary has no knowledge on the model internals

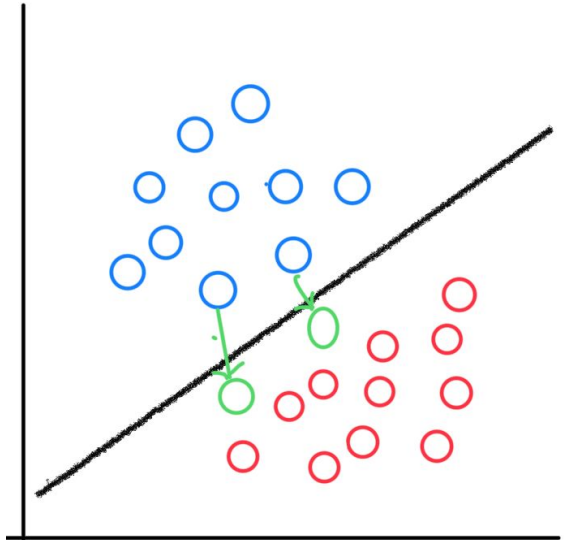
(2). Only access to limited dataset.

(3). The adversary has the capability to only observe the label of the target output.

How a Black Box setting is successful?

The paper exploits the act that adversarial examples transfer well between neural classifiers.

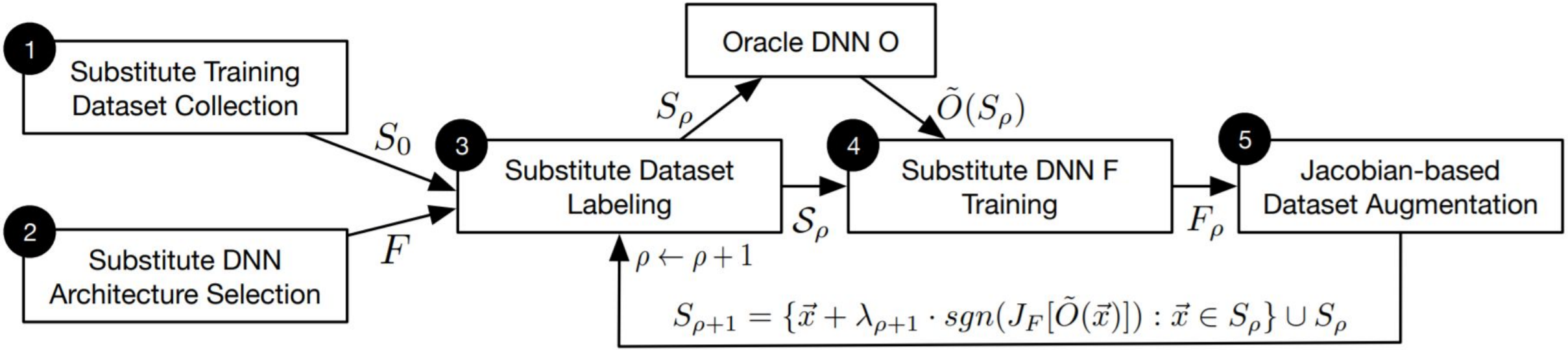
Query to build a substitute model



Crafting adversarial samples

Strategy for attacks:

Goal: Find the direction of varying output



Substitute DNN Training:

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

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

Why do we need a Querying Heuristic ?

- One possible approach is to make infinite number of queries to obtain the oracle's output.
- A DNN with M input component, each taking discrete values among a set of K possible values. The possible number of inputs to be queried is K^M causing intractability.
- Large number of queries renders the adversarial behavior easy to detect.
- Alternative is to randomly selecting additional points to queried.
- But using Gaussian noise to select points, the model was not able to learn.

Jacobian Based Augmentation:

Generate synthetic training inputs is based on identifying directions in which the model's output is varying, around an initial set of training points.

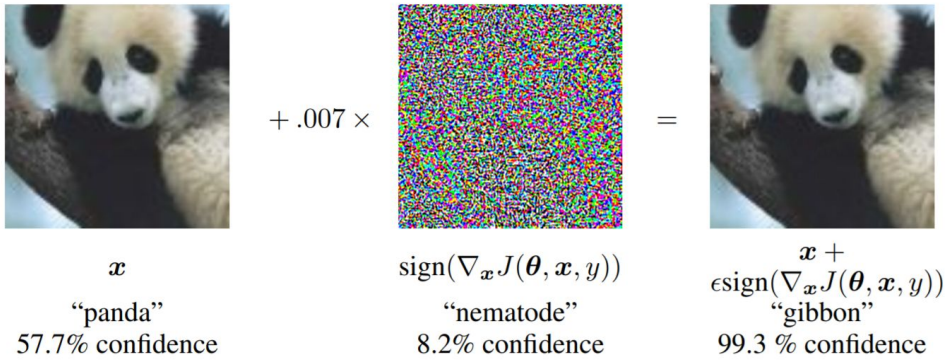
We need more input-output pair.

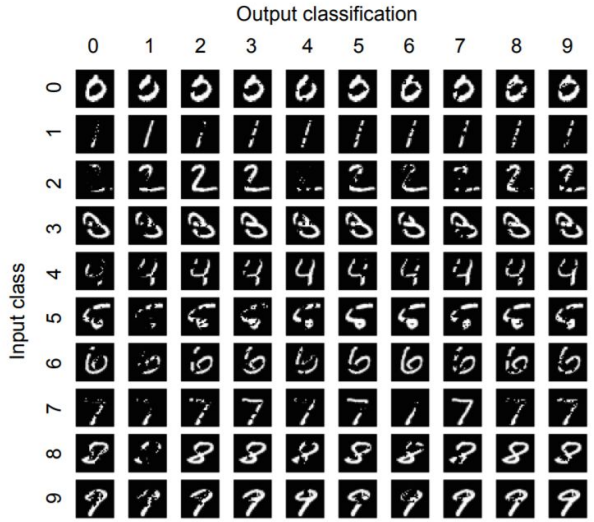
Direction identified by the substitute DNN's Jacobian matrix J_F evaluated at several input points.

$$\text{sgn}\left(J_F(\vec{x})[\tilde{O}(\vec{x})]\right)$$

$$X+ \lambda \cdot \text{sgn}\left(J_F(\vec{x})[\tilde{O}(\vec{x})]\right)$$

Generating Adversarial Examples:





$$\delta_{\vec{x}} = \varepsilon \operatorname{sgn}(\nabla_{\vec{x}} c(F, \vec{x}, y))$$

$$S(\vec{x}, t)[i] = \begin{cases} 0 \text{ if } \frac{\partial F_t}{\partial \vec{x}_i}(\vec{x}) < 0 \text{ or } \sum_{j \neq t} \frac{\partial F_j}{\partial \vec{x}_i}(\vec{x}) > 0 \\ \frac{\partial F_t}{\partial \vec{x}_i}(\vec{x}) \left| \sum_{j \neq t} \frac{\partial F_j}{\partial \vec{x}_i}(\vec{x}) \right| \text{ otherwise} \end{cases}$$

The Goodfellow algorithm is well suited for fast crafting of many adversarial samples with relatively large perturbations thus potentially easier to detect.

The Papernot algorithm reduces perturbations at the expense of a greater computing cost.

Generating Adversarial Examples:Papernot et al. algorithm

Adversary Goal: Misclassify a sample X, s.t target class t ≠ label(X).

$$S(\mathbf{X}, t)[i] = \begin{cases} 0 \text{ if } \frac{\partial \mathbf{F}_t(\mathbf{X})}{\partial \mathbf{X}_i} < 0 \text{ or } \sum_{j \neq t} \frac{\partial \mathbf{F}_j(\mathbf{X})}{\partial \mathbf{X}_i} > 0 \\ \left(\frac{\partial \mathbf{F}_t(\mathbf{X})}{\partial \mathbf{X}_i}\right) \left| \sum_{j \neq t} \frac{\partial \mathbf{F}_j(\mathbf{X})}{\partial \mathbf{X}_i} \right| \text{ otherwise} \end{cases}$$

i : input feature,
rejects input components with a negative target derivative or an overall positive derivative on other classes

The product on the second line allows us to consider all other forward derivative components together in such a way that we can easily compare S(X, t)[i] for all input features

Papernot et al. algorithm:Cont.

Algorithm 1 Crafting adversarial samples

$\mathbf{X}$ is the benign sample, $\mathbf{Y}^*$ is the target network output, $\mathbf{F}$ is the function learned by the network during training, Υ is the maximum distortion, and θ is the change made to features. This algorithm is applied to a specific DNN in Algorithm 2.

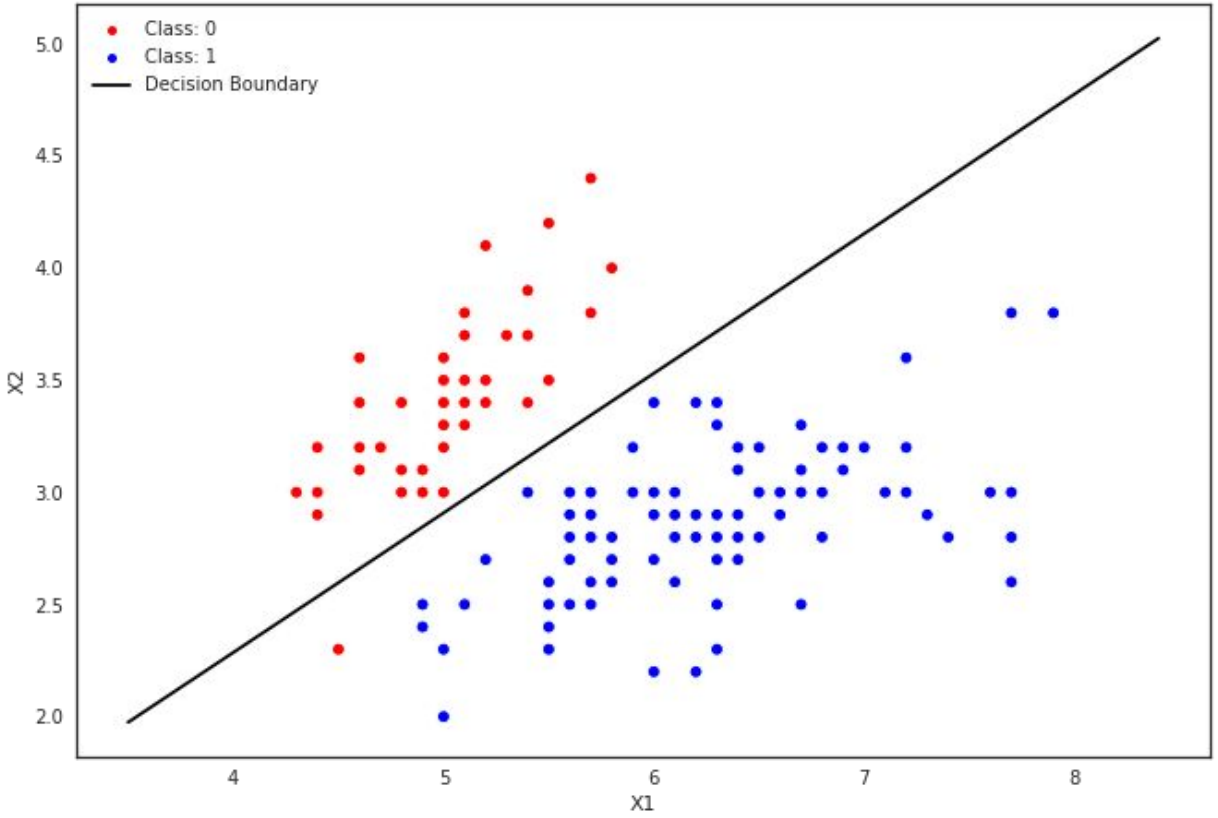
Input: $\mathbf{X}$, $\mathbf{Y}^*$, $\mathbf{F}$, Υ, θ

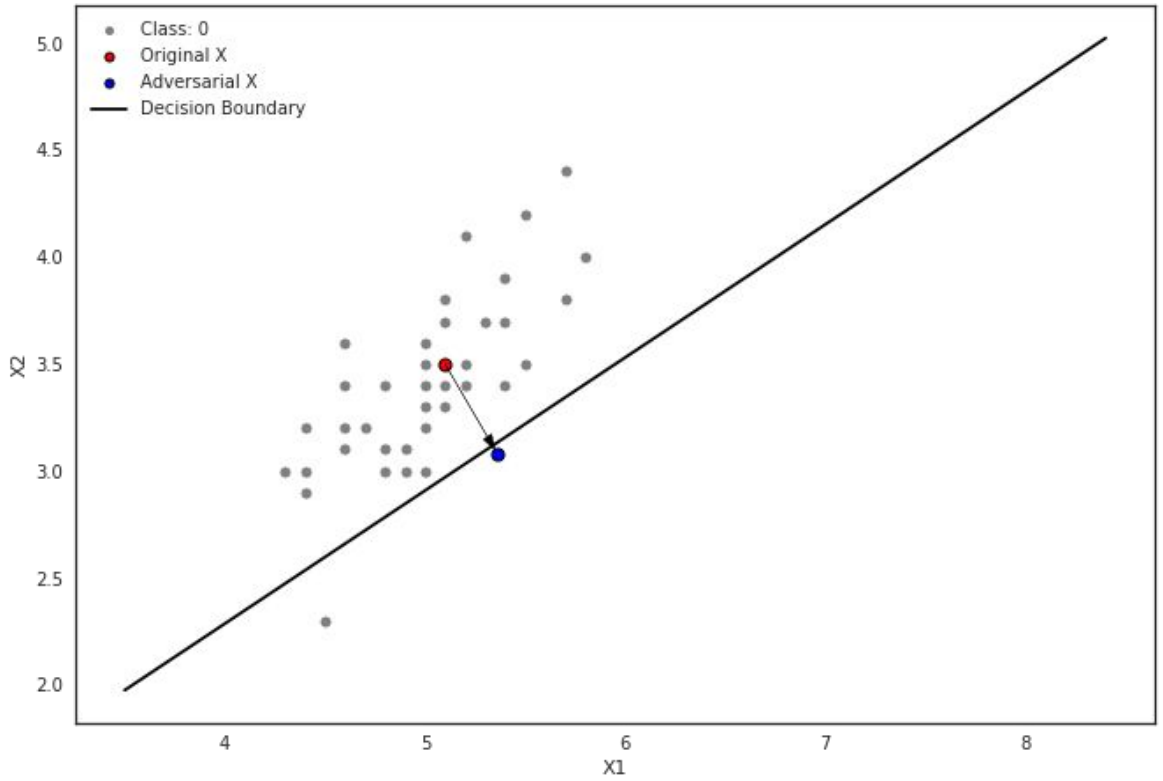
1: $\mathbf{X}^* \leftarrow \mathbf{X}$
2: $\Gamma = \{1 \ldots |\mathbf{X}|\}$
3: **while** $\mathbf{F}(\mathbf{X}^*) \neq \mathbf{Y}^*$ and $||\delta_{\mathbf{X}}|| < \Upsilon$ **do**
4: Compute forward derivative $\nabla \mathbf{F}(\mathbf{X}^*)$
5: $S = \texttt{saliency\_map}(\nabla \mathbf{F}(\mathbf{X}^*), \Gamma, \mathbf{Y}^*)$
6: Modify $\mathbf{X}^*_{i_{max}}$ by θ s.t. $i_{max} = \arg\max_i S(\mathbf{X}, \mathbf{Y}^*)[i]$
7: $\delta_{\mathbf{X}} \leftarrow \mathbf{X}^* - \mathbf{X}$
8: **end while**
9: **return** $\mathbf{X}^*$

Θ: the amount by which the selected feature is perturbed

γ: maximum number of iterations/maximum distortion allowed in a sample.

Generating Adversarial Example: Linear Regression



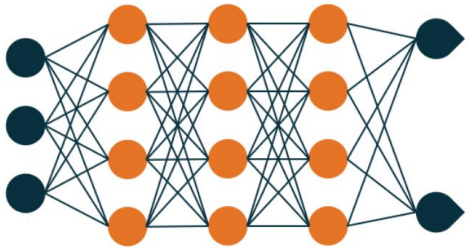


Attack against the MetaMind Oracle:

MNIST handwritten digit dataset(50,000 training samples)

MetaMind

Used API



Classifier with a 94.97% accuracy

adversary observes the oracle label

A : Substitute Architecture for training

\+ 150 samples from the MNIST test set

ID	In	Out	CM	CM	RL	RL	RL	S
A	784	10	32	64	200	200	-	10
B	3072	43	64	128	256	256	-	43
C	3072	43	32	64	200	200	-	43
D	3072	43	32	64	200	200	-	43
E	3072	43	64	64	200	200	100	43

Initial Substitute Training Sets

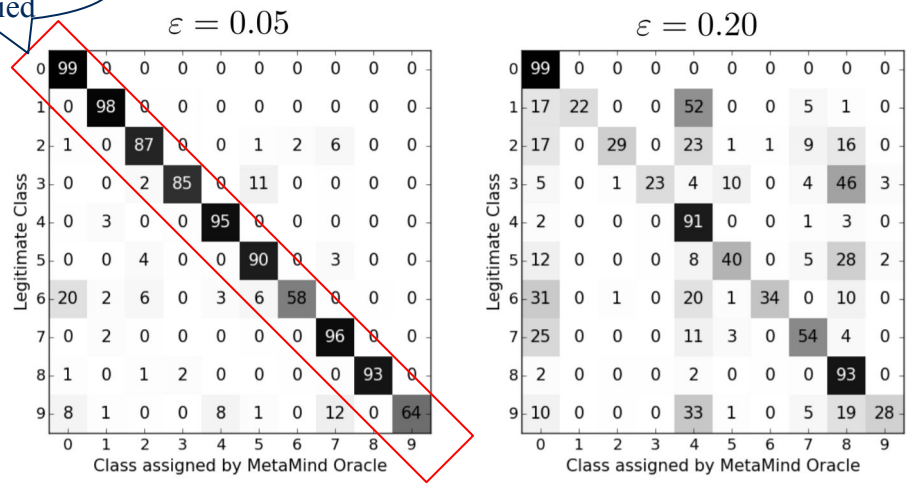
Jacobian based Augmentation to generate more additional synthetic data

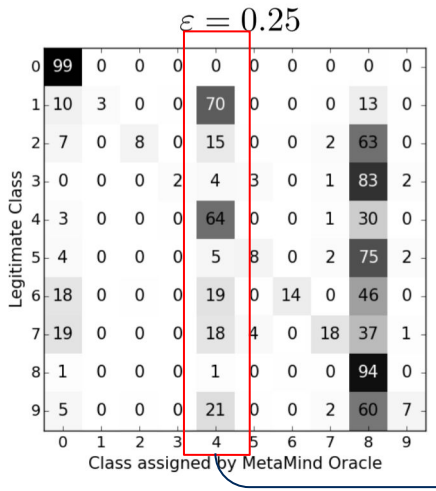
After 6 epochs of training the DNN, the accuracy of classification is 67% for the Handcrafted digits and for the MNIST digits the accuracy is 81%.

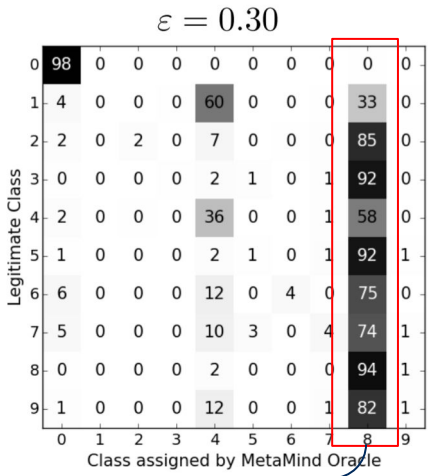
UNIVERSITY OF TORONTO

Attack against the MetaMind Oracle:Adversarial Sample Crafting

correctly classified

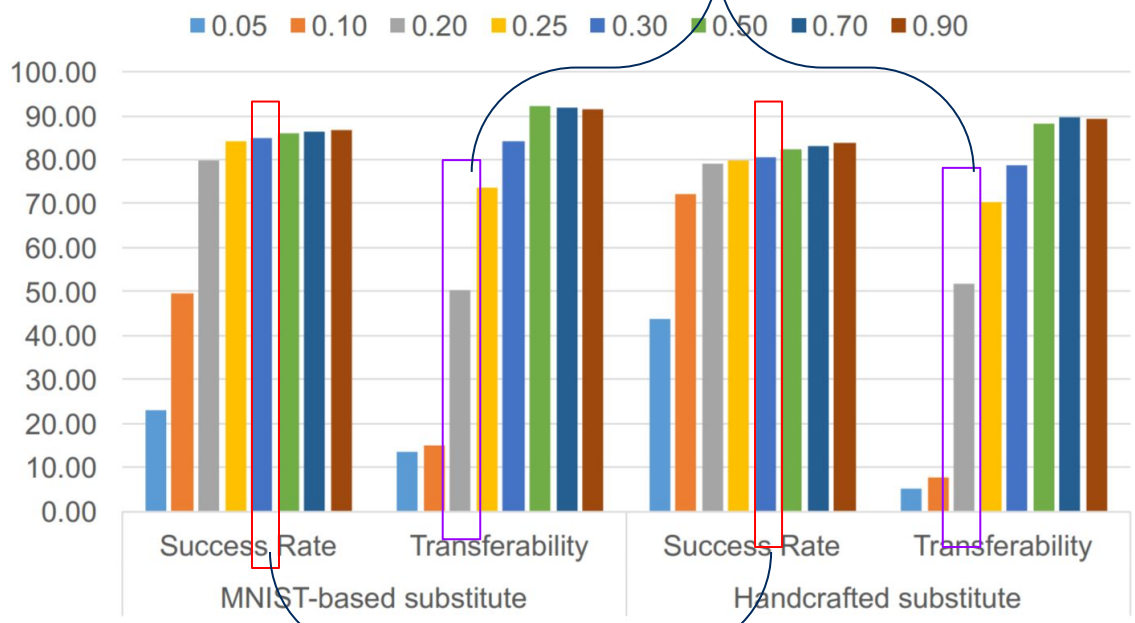






most samples being classified as 4s and 8s as ε increases

Handcrafted set by the adversary limits the transferability of adversarial samples for all model except for input variation = 0.2



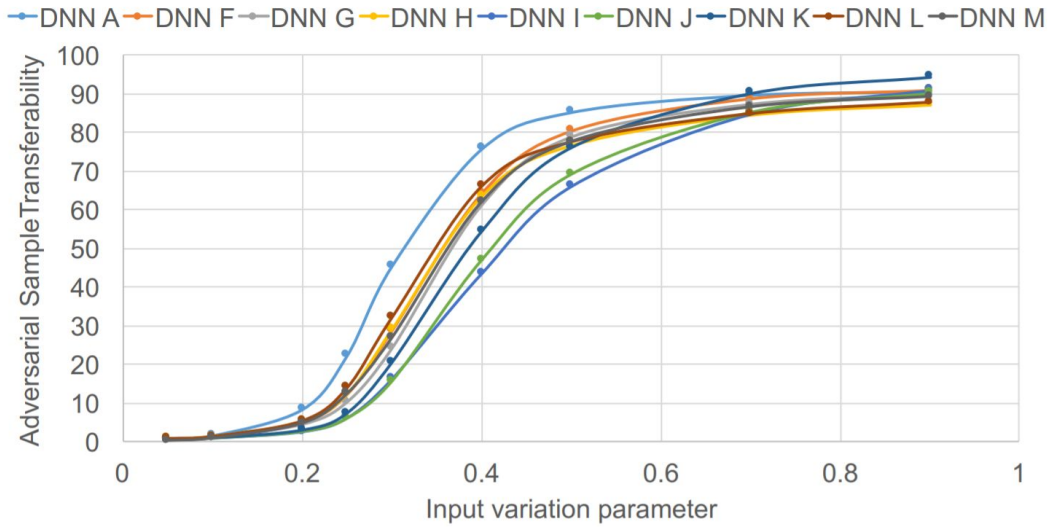
Attack strategy effectively damages the output integrity of the MetaMind oracle.

UNIVERSITY OF TORONTO

Discussion

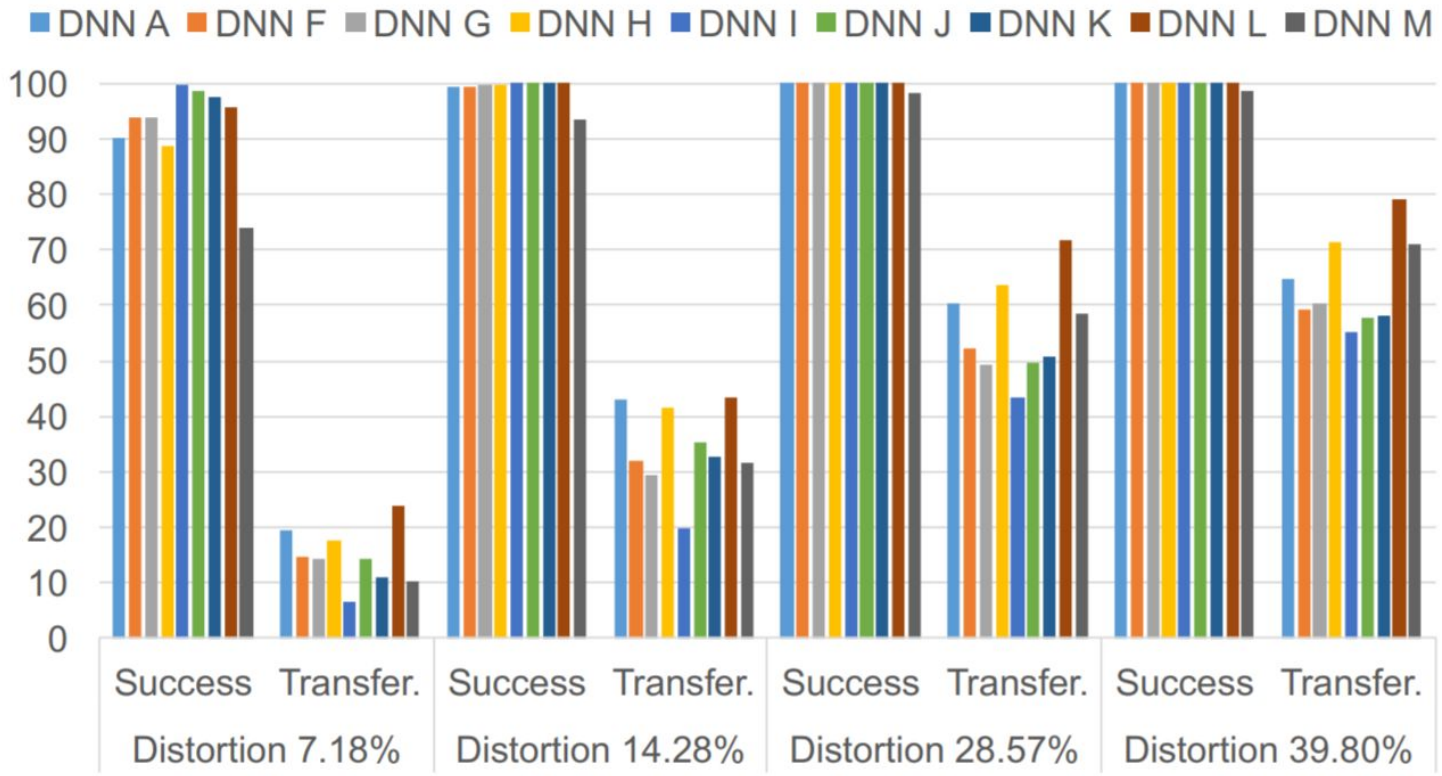
- How can substitute training be fine-tuned to improve adversarial sample transferability?
- For each adversarial sample crafting strategies, which parameters optimize transferability?

Adversarial Sample Crafting: Transferability and Success



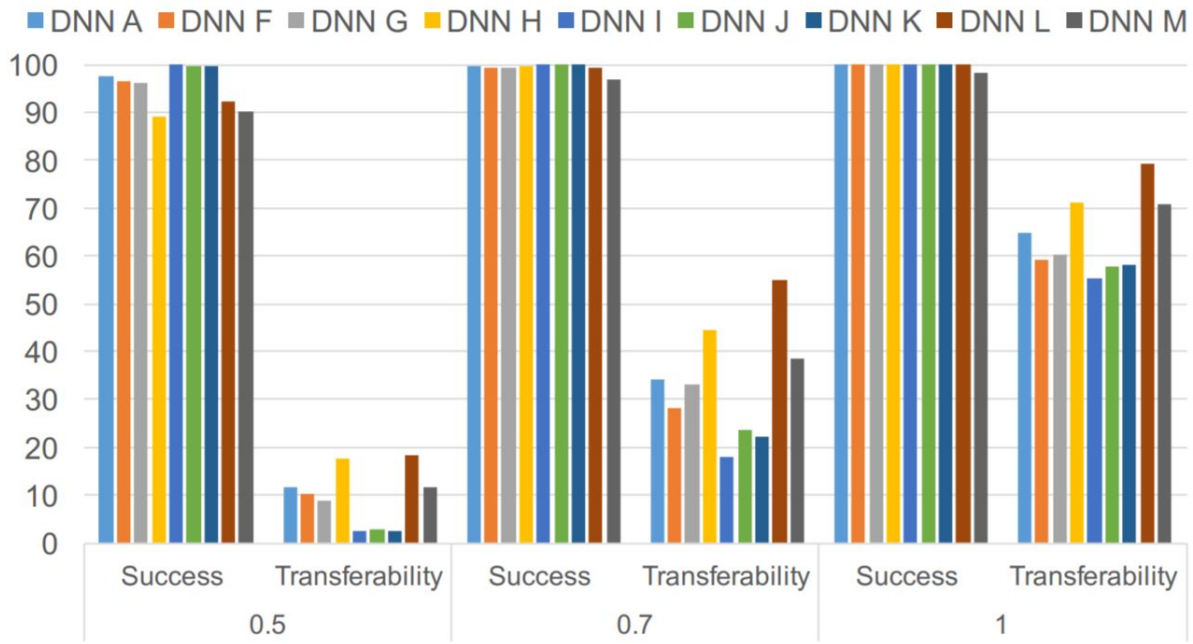
Impact of input variation ε in the Goodfellow crafting algorithm on the transferability of adversarial samples.

Adversarial Sample Crafting: Transferability and Success



Impact of the maximum distortion Y in the Papernot algorithm on success rate and transferability of adversarial samples.

Adversarial Sample Crafting: Transferability and Success



Impact of the input variation ε in the Papernot algorithm on the success rate and adversarial sample transferability computed for $\varepsilon \in \{0.5, 0.7, 1\}$ on DNNs from Table 1 with distortion Y = 39.80%

Attack Validation

The DNN misclassifies 84.24% of the adversarial inputs crafted

- Substitutes can also be learned with logistic regression.
- The attack generalizes to additional ML models

They misclassify adversarial examples at rates of 96.19% and 88.94%

The LR substitute works better for Amazon API as the model trained by Amazon is a multinomial logistic regression.

Defense Strategies

(1) Reactive: one seeks to detect adversarial examples.

(2) Proactive: one makes the model itself more robust.

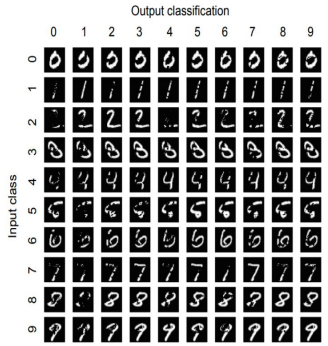
Gradient Masking:

- A model with no useful gradients.
- Even if the defender tries to prevent the direction in which the model is sensitive, there are other ways to find the directions.

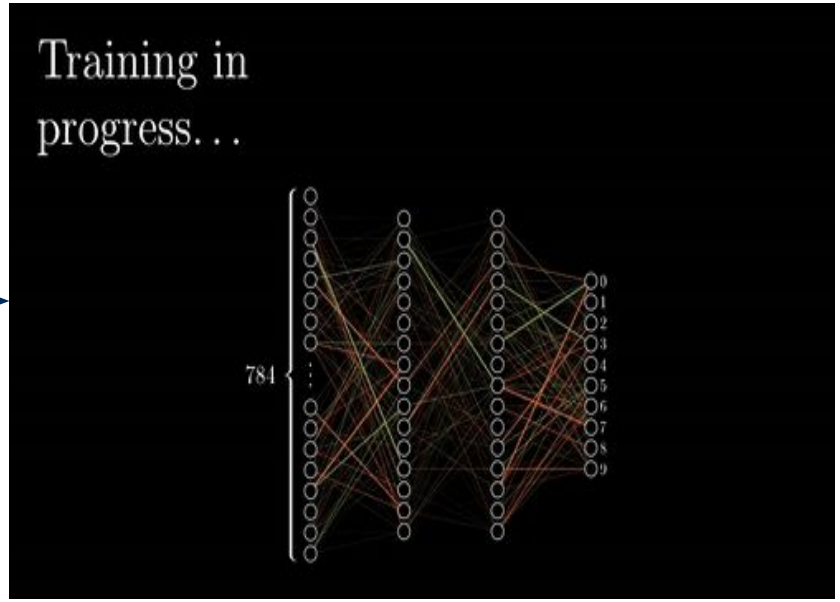
Defense Strategy:

- Adversarial training
- Defensive distillation for DNNs

Defense to attacks: Adversarial Training



Adversarial Examples



NN Training

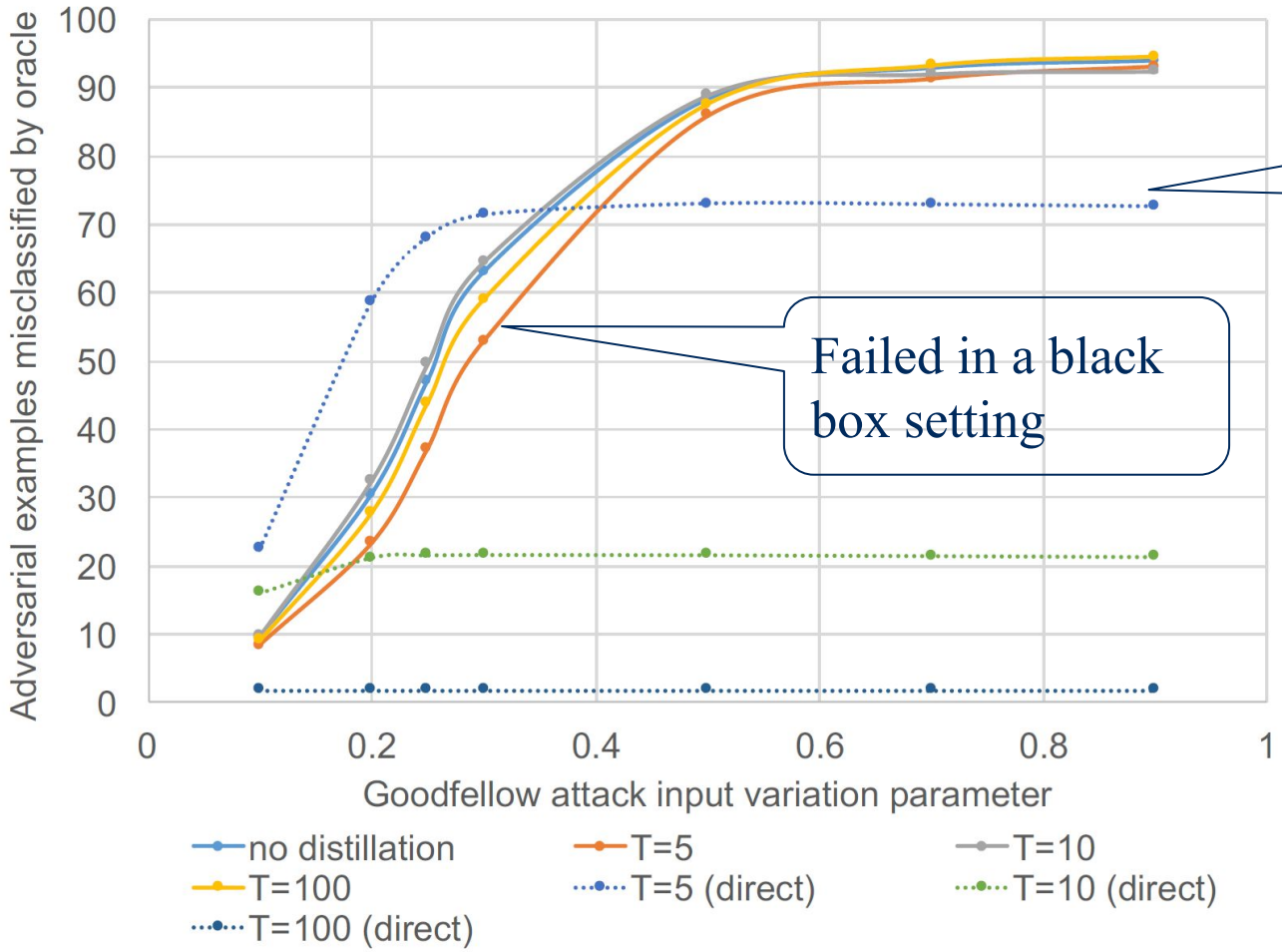
Including adversarial examples in training helps to decrease the misclassification rate to 8.75%

Defensive distillation

- Smooths the model's decision surface in adversarial directions exploited by the adversary.
- First model is trained with "hard" labels and then provides "soft" labels used to train the second model.
- More robust to attacks such as the fast gradient sign method or the Jacobian-based saliency map approach.

$$F(X) = \left[\frac{e^{z_i(X)/T}}{\sum_{l=0}^{N-1} e^{z_l(X)/T}} \right]_{i \in 0..N-1}$$

Performance of Defensive Distillation:



defensive distillation defends against in white-box settings

Failed in a black box setting

Failure of Defensive Distillation:

Distillation defends against attack by reducing the gradients in local neighbourhood of training points.

The substitute DNN is not distilled and has gradients required for the fast gradient sign method to be successful when computing adversarial examples.

But what if there were no gradients? What if an infinitesimal modification to the image caused no change in the output of the model ?

Gradient Masking

Many trivial ways to hid the gradient.

- Changing the model to return most likely class and not the probability.
- infinitesimal changes in the input will not change the output at all.

But are we making our model more robust ?

We are just giving the adversary fewer clues to figure out the holes in the model.

A method of attack is where an attacker can have a substitute model with a gradient, make adversarial examples for their model.

Limitation:

- The defender might increase the attacker's cost by training models with higher input dimensionality or modeling complexity leading to increased number of queries to train the substitute.
- What if the input to the target DNN is not an image data ? Will the attack strategy still be useful?
- The black box setting consider the attacks on test-time inputs, but there can be attacks which can be induced during the training phase.

Suggestion:

The paper discusses about the black box attack but it would have been better if the authors presented a comparative analysis using White Box Setting.

- Open question : Is there any defense strategy which can be useful against DNN attack ?

Difficulty Faced:

- When we have a model where $\vec{x^*} = \vec{x} + \arg\min\{\vec{z} : \tilde{O}(\vec{x} + \vec{z}) \neq \tilde{O}(\vec{x})\} = \vec{x} + \delta_{\vec{x}}$, we need to formalize the adversarial goal of finding a minimal perturbation that forces the targeted oracle to misclassify.
- A closed form solution cannot be found for a non-convex ML model (DNN).
- Previous adversarial attacks was based on approximating a solution using gradient-based optimization on functions defined by the DNN.

Improvement from previous works:

The study releases the two assumptions made :

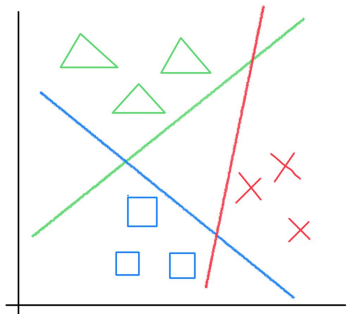
(a). Learning Substitute : gives benefit of having full access to the model and apply previous adversarial crafting methods.

(b). Replacing independently collected training set with a synthetic dataset constructed by the adversary with synthetic inputs and labeled by observing the DNN's output.

Outline:

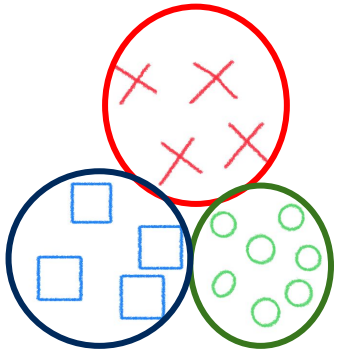
- Malicious inputs modified to generate erroneous model outputs while appearing unmodified to human observers.
- All existing work requires the knowledge of the model internals or its training data.
- First work to induce attacks in the DNN remotely and without the knowledge of the model internals.
- Performed real world attacks on a DNN hosted by MetaMind and also attack against models hosted by Amazon and Google.

Threat Model:(TO DO)



Targeted Model

The attacker targets the DNN. The DNN can be used to classify the handwritten digits or images of traffic signal.



Adversarial Capabilities

$$\tilde{O}(\vec{x}) = \arg \max_{j \in 0..N-1} O_j(\vec{x})$$

The oracle O, is the targeted DNN. The output label O˜(~x) is the index of the class assigned the largest probability by the DNN.

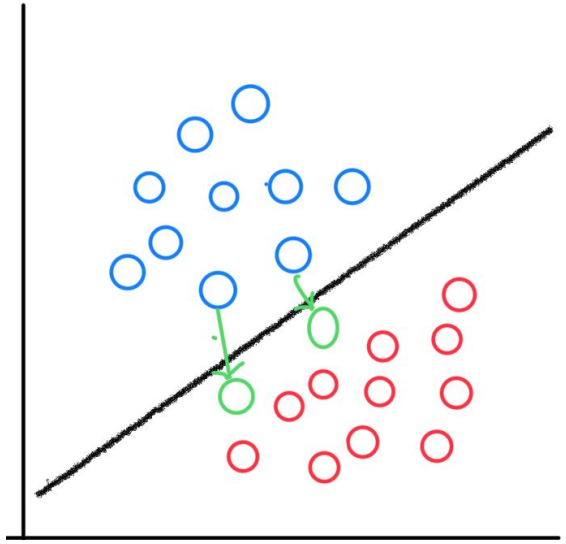
Adversarial Goal

$$\vec{x^*} = \vec{x} + \arg\min\{\vec{z} : \tilde{O}(\vec{x} + \vec{z}) \neq \tilde{O}(\vec{x})\} = \vec{x} + \delta_{\vec{x}}$$

To produce a minimally altered version of any input ~x, named adversarial sample, and denoted x~*, misclassified by oracle O

Overall Attack Strategy

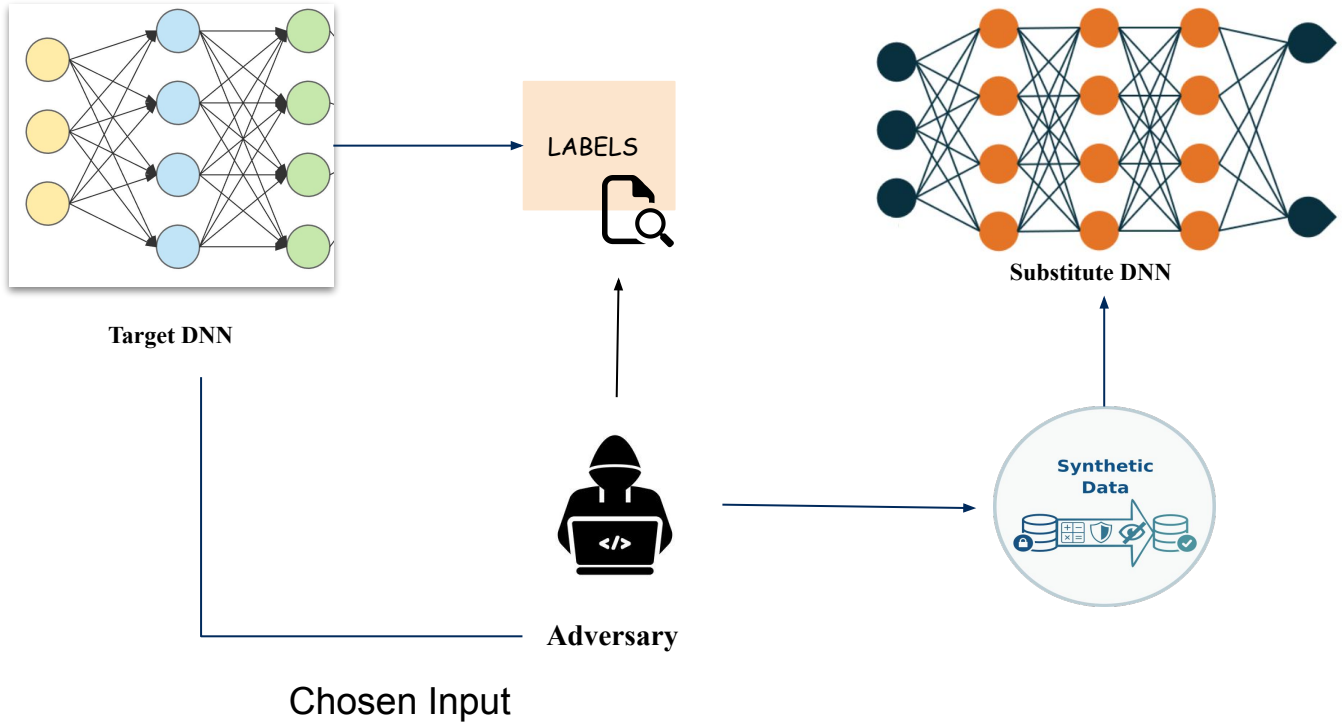
Query to build a substitute model



Crafting adversarial samples

The Black Box Attack:

- Only access labels
- Limited label queries
- Scales to various ML classifiers



Certified Adversarial Robustness via Randomized Smoothing

Cohen et. al. 2019
Presented by Philip Fradkin

Types of defenses against adversarial attacks

Exact certified defenses:

- Classifier g, input x, radius r
- Can guarantee whether there is a perturbation within a certain radius r to misclassify x

Utilize techniques:

- Satisfiability Modulo Theories
- Mixed integer linear programming

Issues:

- No method scales to 100,000 activation networks

Types of defenses against adversarial attacks

Exact certified defenses:

- Classifier g, input x, radius r
- Can guarantee whether there is a perturbation within a certain radius r to misclassify x

Utilize techniques:

- Satisfiability Modulo Theories
- Mixed integer linear programming

Issues:

- No method scales to 100,000 activation networks

Conservative Certified defenses:

- Certify that no perturbation exists or decline to make a prediction

Utilize Techniques:

- Local smoothness within vicinity of x
- Step through network approximating perturbation magnitude

Issues:

- Assume architecture
- Numerically unfeasible for big models

Types of defenses against adversarial attacks

Empirical defenses:

- Usually involves altering training process

Utilize techniques

- Add adversarial samples to training set

Issues:

- No theoretical guarantees
- More sophisticated attacks are able to break the defense

Conservative Certified defenses:

- Certify that no perturbation exists or decline to make a prediction

Utilize Techniques:

- Local smoothness within vicinity of x
- Step through network approximating perturbation magnitude

Issues:

- Assume architecture
- Numerically unfeasible for big models

Lipschitz constant

Intuitively: Small change in input corresponds to a small change in the output

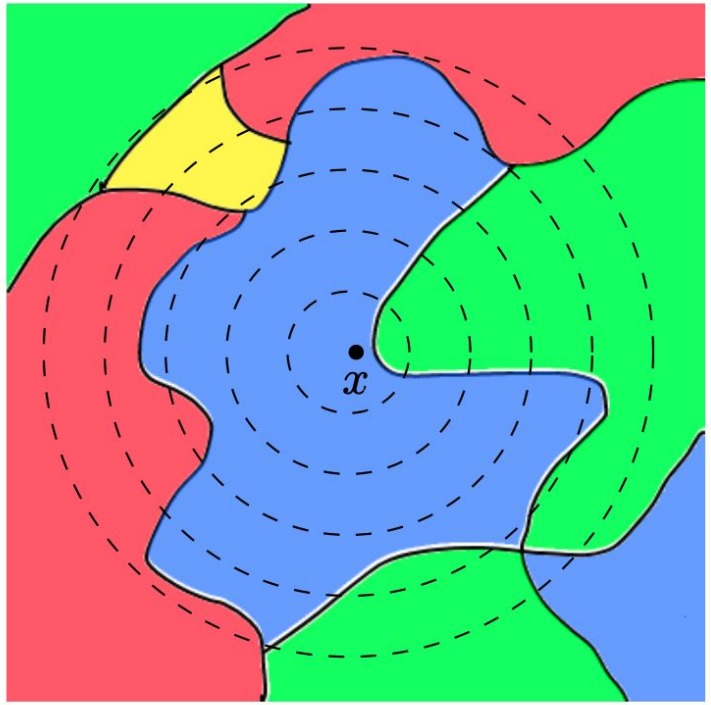
- Small constant implies small perturbations in inputs leads to small changes in output
- In linear models it is bounded by slope
- $K \|x_1 - x_2\| \geq \|f(x_1) - f(x_2)\|$
- Neural networks have large Lipschitz constants
- What if instead of redefining the function (network) we redefine what the output of this function looks like?

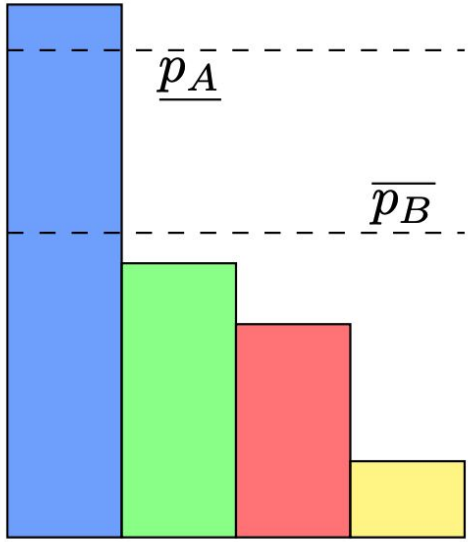
Gaussian smoothing

$$g(x) = \arg\max_{c \in \mathcal{Y}} \mathbb{P}(f(x+\varepsilon) = c)$$

where $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$

- Stochastic method
- No need to change the neural network
- Sampling within an l2 ball of the sample by adding gaussian noise to the input
- Taking the maximum predicted class





Gaussian smoothing

$$g(x) = \arg\max_{c \in \mathcal{Y}} \mathbb{P}(f(x+\varepsilon) = c)$$

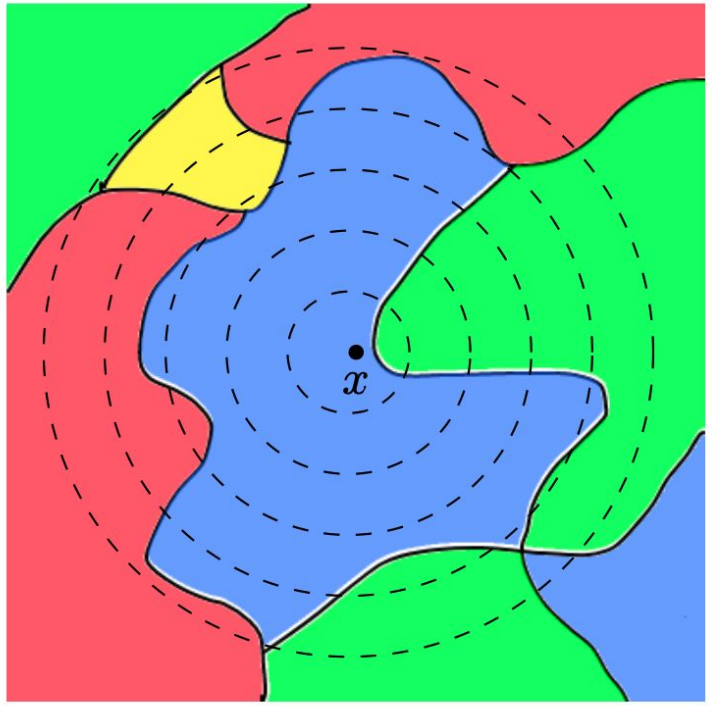
where $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$

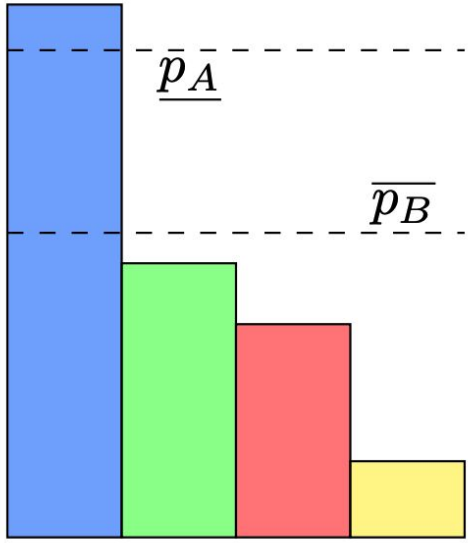
Pros:

- No assumptions of architecture
- Exact certified defense

Cons:

- Prediction is stochastic - have to resort to monte carlo
- Multiple forward passes increasing inference time (although still fastest certified defense)

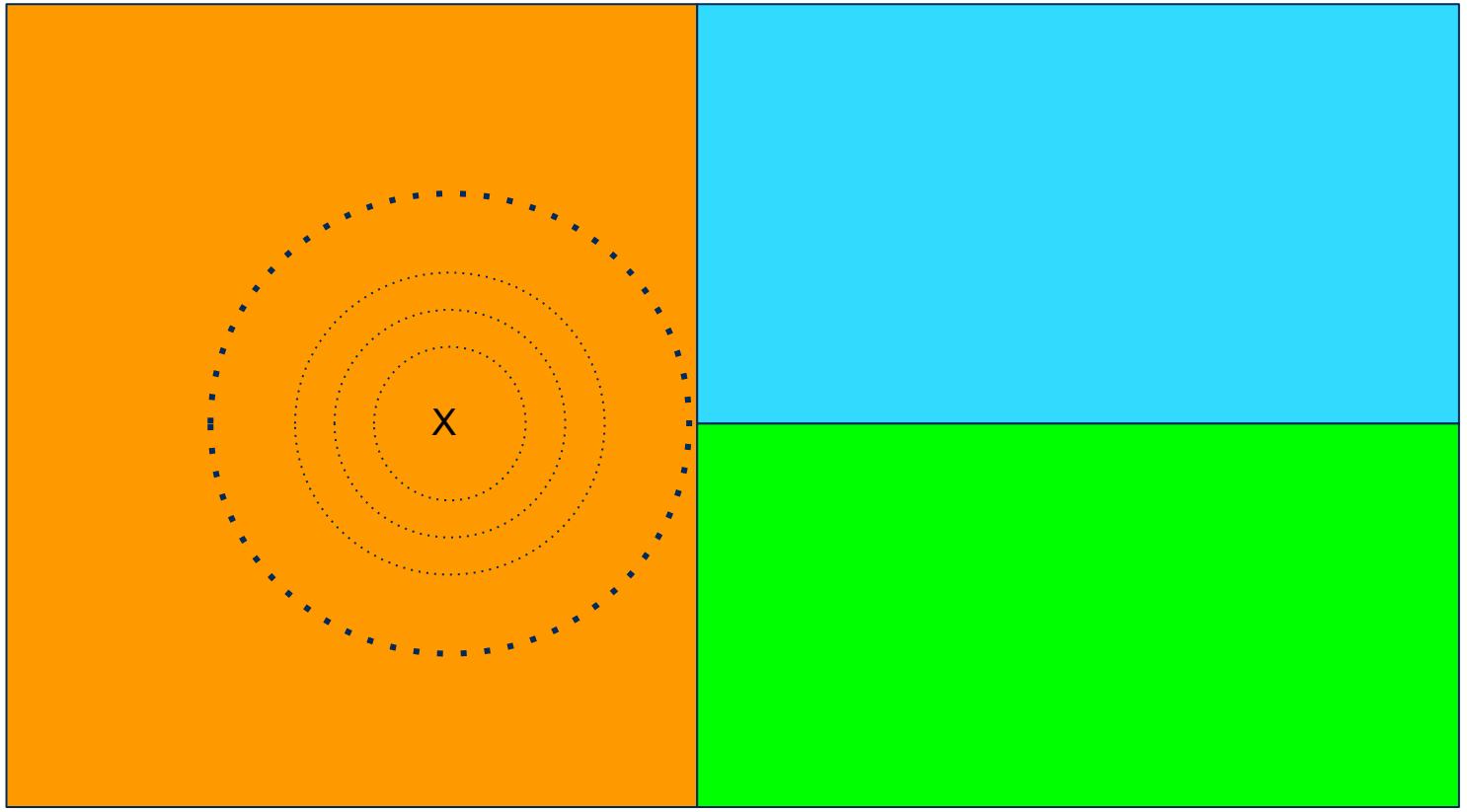




Example

Sample x - classified as

- orange 100%
- Teal 0%



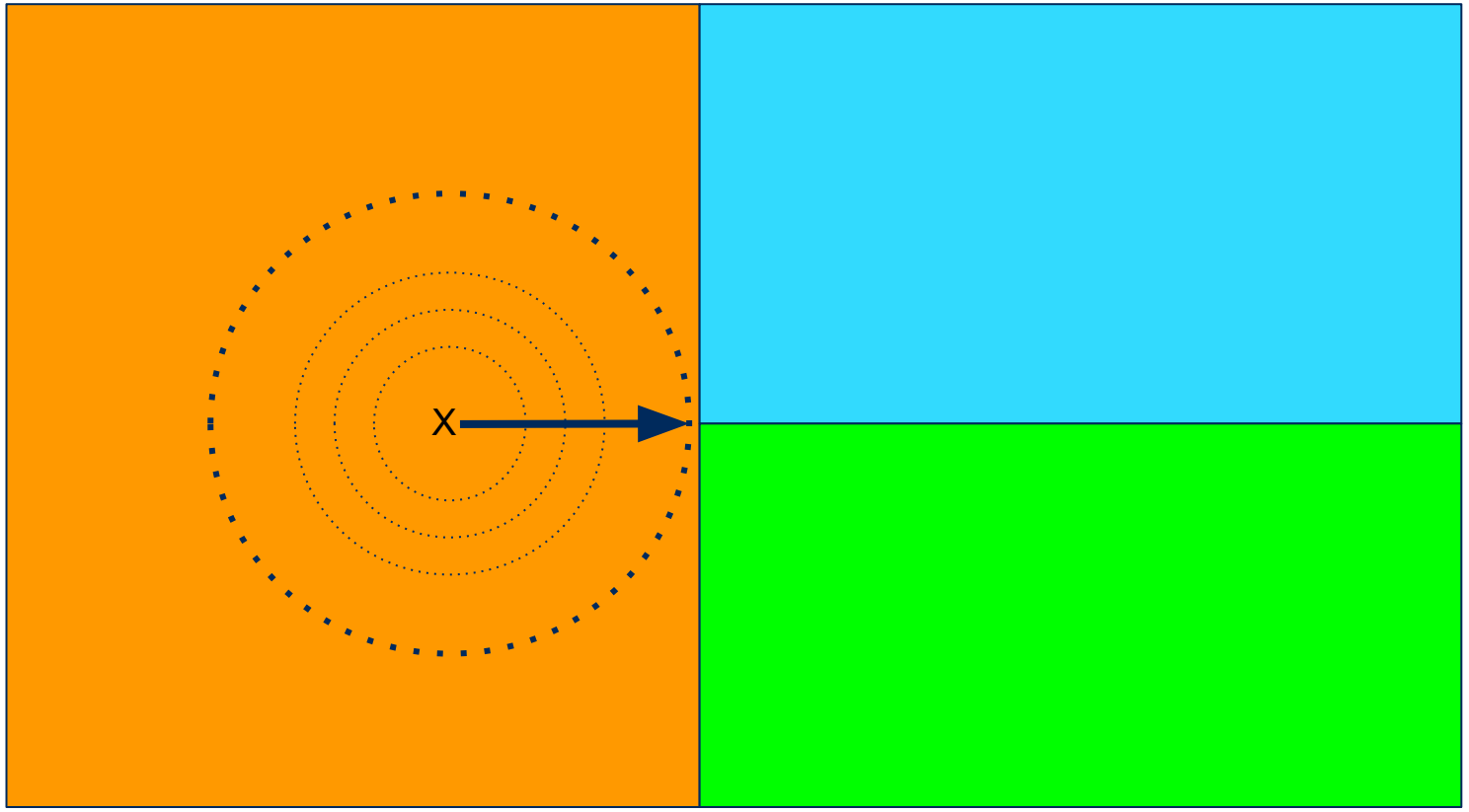
Example

Sample x - classified as

- orange 100%
- Teal 0%

Introduce Perturbation

Magnitude of L2 norm ball



Example

Sample x - classified as

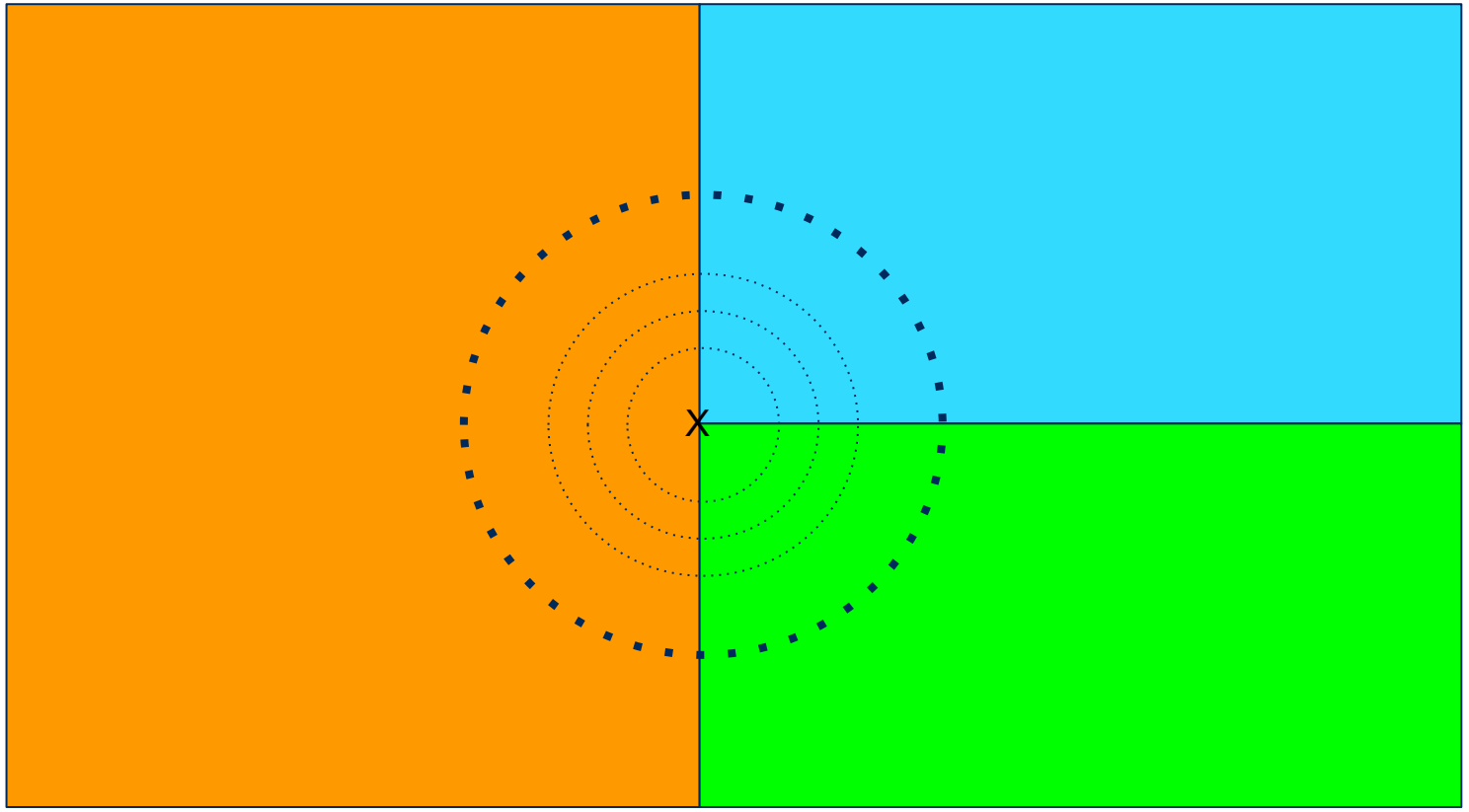
- orange 100%
- Teal 0%

Introduce Perturbation

Magnitude of L2 norm ball

Still know that at least

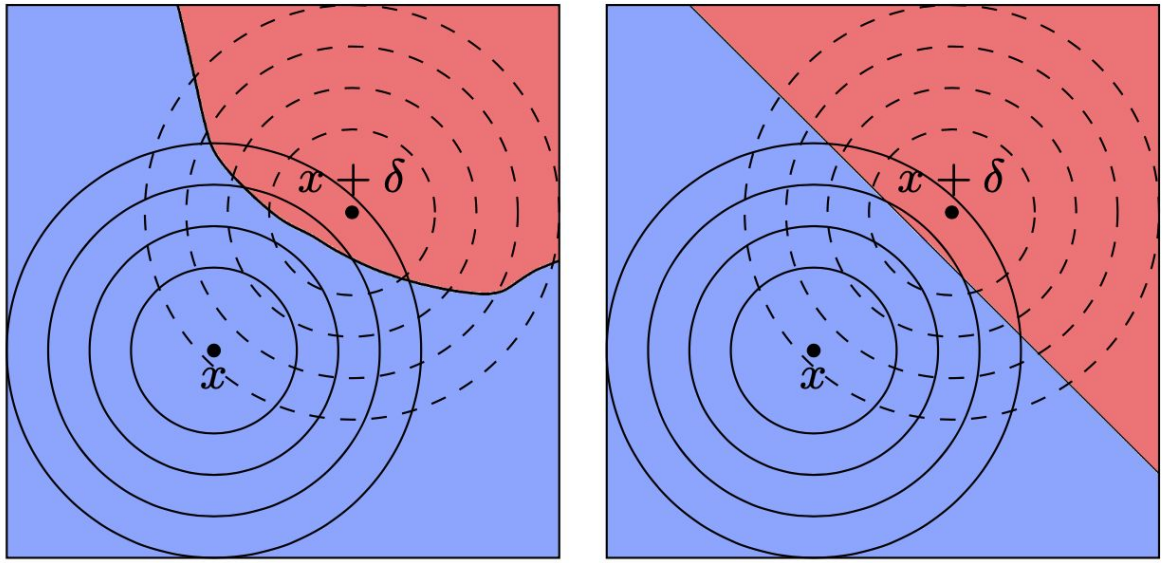
- 50% is orange
- Don’t know the other 50%



Robustness Guarantee Intuition

If inference with gaussian smoothing predicts 98% accuracy params: N(0, σ)

How will the prediction probability change if we shift the sample in the input space?



Robustness Guarantee Intuition

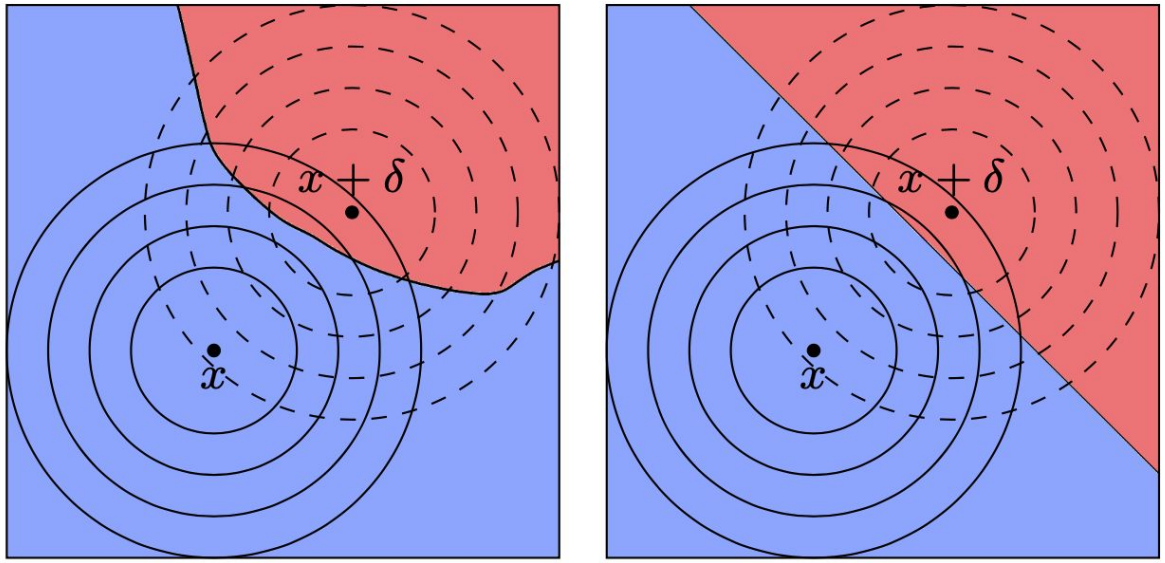
If inference with gaussian smoothing predicts 98% accuracy params: N(0, σ)

How will the prediction probability change if we shift the sample in the input space?

How much will the prediction probability shrink?

$||\delta||_2 < \sigma \Phi^{-1}(0.98)$

As long as perturbation is less than δ will predict blue class over 50% probability

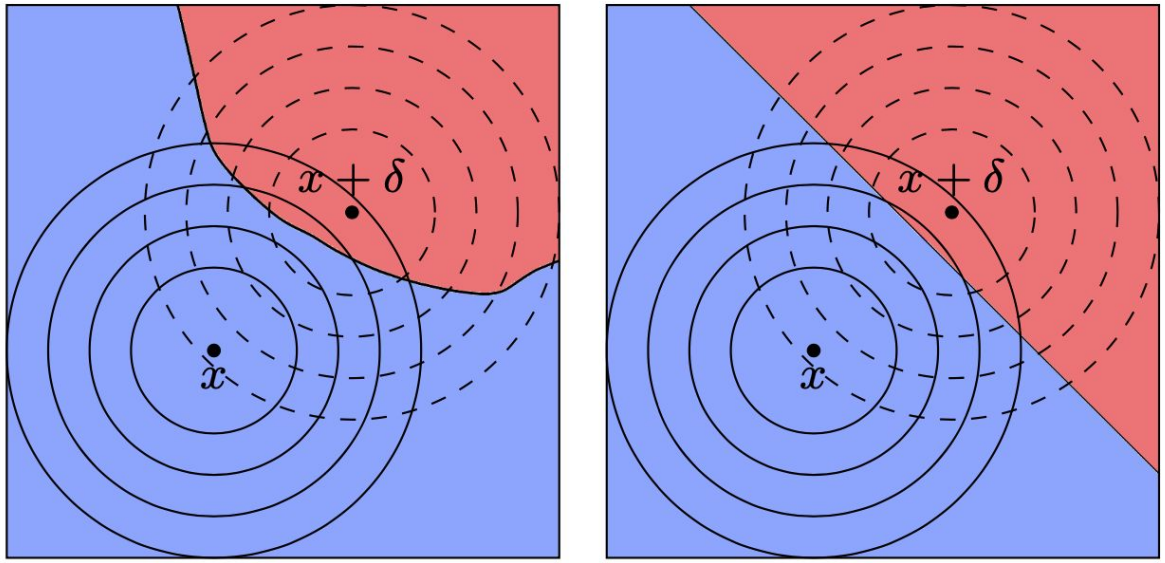


Robustness Guarantee

Assume: $p_a \in (½, 1]$

$$\mathbb{P}(f(x+\varepsilon) = c_A) \geq \underline{p_A}.$$

- Neyman-Pearson lemma: There exists a critical region C, which provides most statistical power to evaluate null vs alternative hypothesis
- f* is the worst possible classifier when decision boundary is normal to perturbation δ



Certification - extracting a probability

Train with gaussian augmentation with σ^2 variance

- n = 100,000 samples
- n_0 = 100 Monte Carlo
- α = 0.001

Estimate p_A with a high number of samples

LowerConfBound - binomial distribution confidence interval

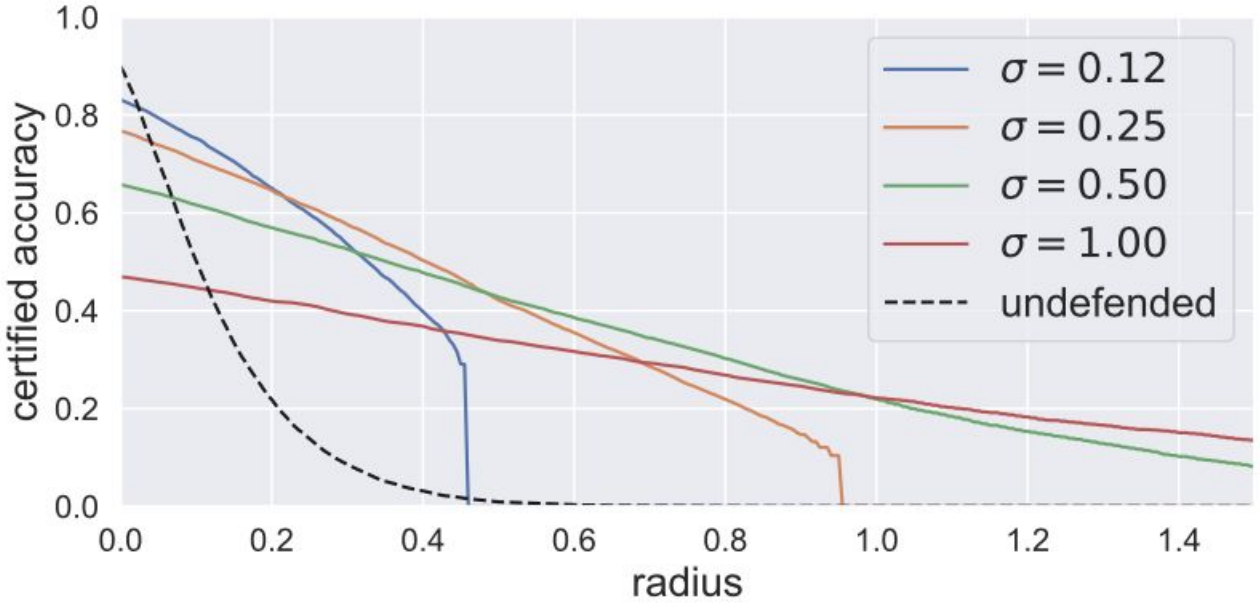
 # certify the robustness of g around x
function CERTIFY($f, \sigma, x, n_0, n, \alpha$)
counts0 ← SAMPLEUNDERNOISE(f, x, n_0, σ)
$\hat{c}_A$ ← top index in counts0
counts ← SAMPLEUNDERNOISE(f, x, n, σ)
$\underline{p_A}$ ← LOWERCONFBOUND(counts$[\hat{c}_A], n, 1-\alpha$)
if $\underline{p_A} > \frac{1}{2}$ **return** prediction $\hat{c}_A$ and radius $\sigma \Phi^{-1}(\underline{p_A})$
else return ABSTAIN

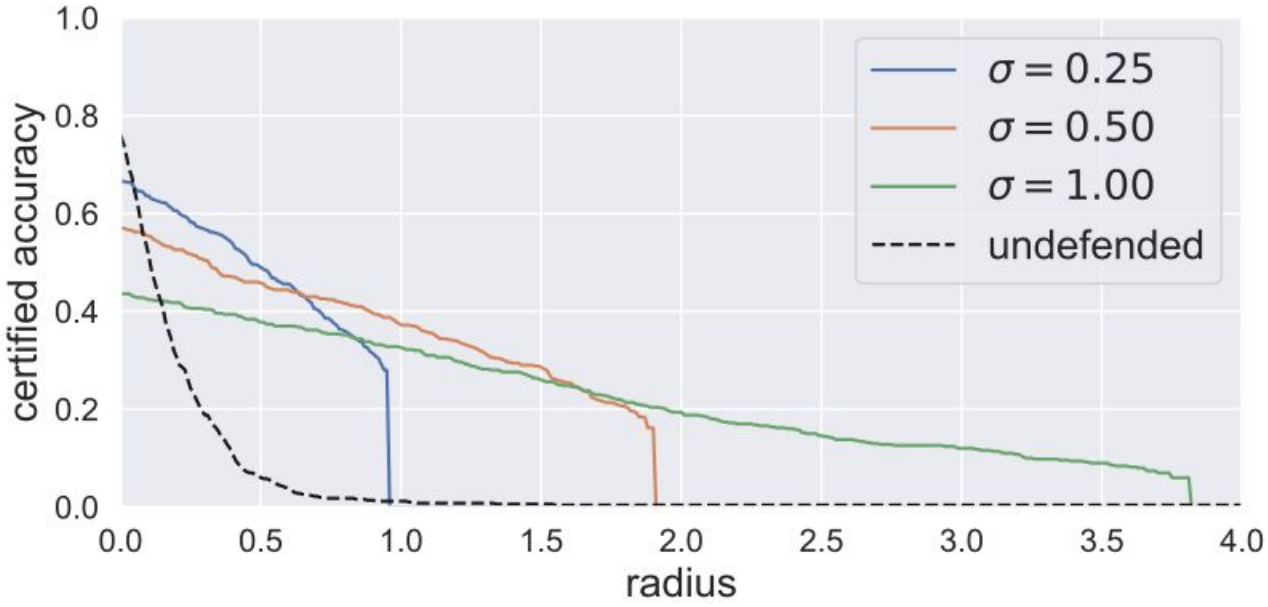
Robustness-Accuracy tradeoff

Plotted: Models trained at different noise levels

Higher noise (during training) lower accuracy

Higher noise (during training) higher radius of defense





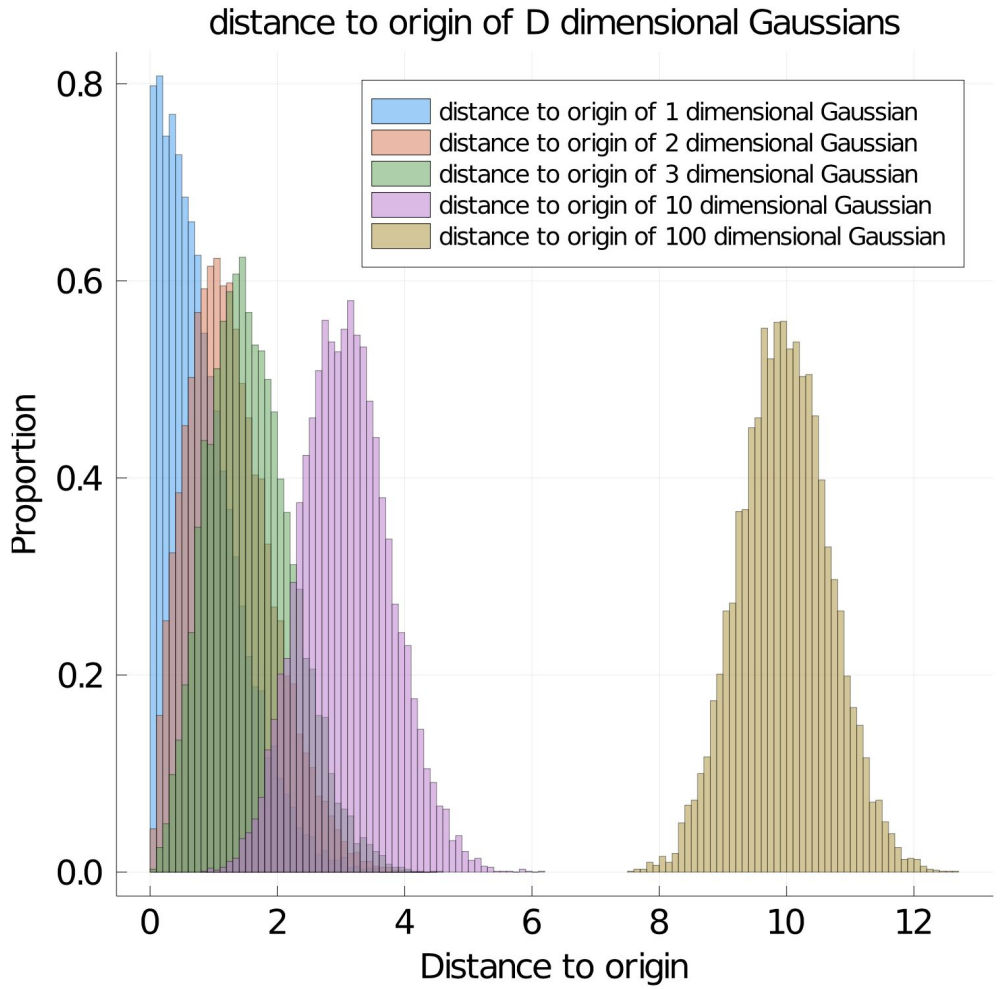
UNIVERSITY OF TORONTO

Shortcomings of the method

- Model trained without normal noise distribution augmentation would not work well

Shortcomings of the method

- Model trained without normal noise distribution augmentation would not work well
- Distance from origin changes based on number of dimensions
- Performance suffers
- To ensure reproducibility have to set seed but that makes it vulnerable to attacks
- Is there a point where the required number of samples needed to be drawn is no longer feasible due to the curse of dimensionality?



Conclusions

- Derived a theoretical bound on Gaussian smoothing for adversarial robustness
- Well designed set of empirical experiments
- Sneaky argument of no need to change architecture but in empirical experiments specific design choices are made
- “Lp norm balls are the equivalent of a fruit fly for biology” - Roger Grosse
- No real adversary is limited to an l_p norm ball
- Interesting follow up work on “Lipschitz constraints for neural networks”

Introduction

Outline:

- What are attacks on neural networks? (Phil)
  - Explain what FGSM is (https://arxiv.org/pdf/1412.6572.pdf)
- Motivating examples (We all contribute)
- Are there defenses against neural networks? (Farhan - section 6)
  - Adversarial training (https://arxiv.org/pdf/1412.6572.pdf)
- Why do the attacks happen? (Andrew)
- Impacts of the attacks (andrew)

Robustness Guarantee

Can show

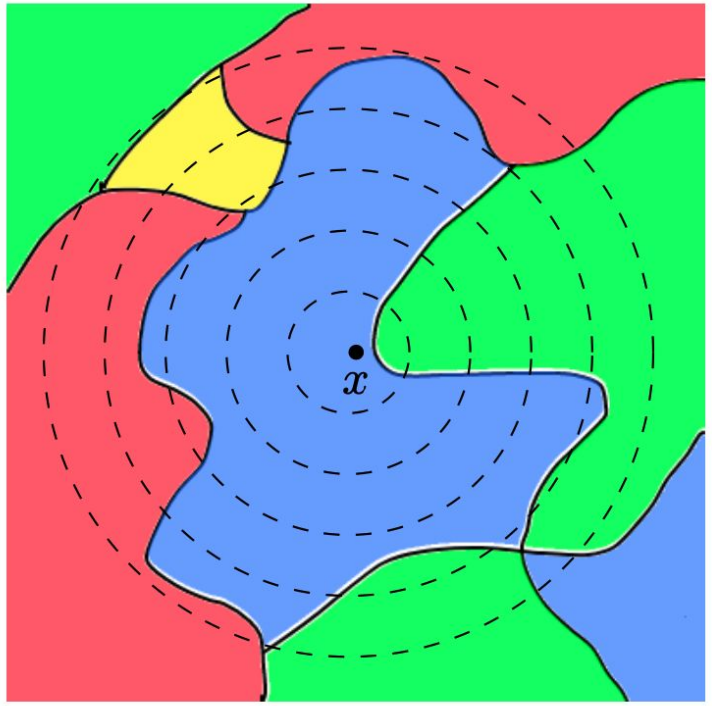
$$f^*(x') = \begin{cases} c_A & \text{if } \delta^T(x' - x) \leq \sigma \|\delta\|_2 \Phi^{-1}(\underline{p_A}) \\ c_B & \text{otherwise} \end{cases} \quad (4)$$

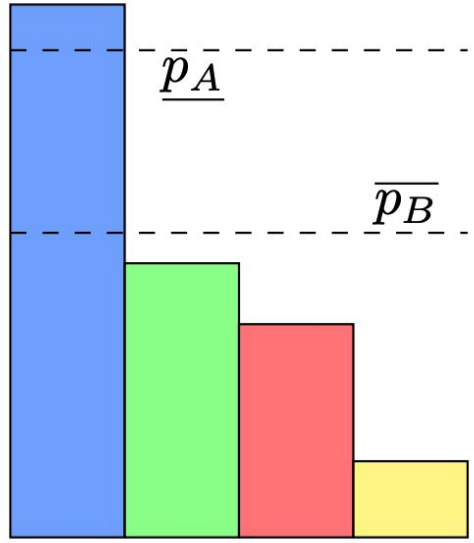
Radius is then determined by:

$$\Phi\left(\Phi^{-1}(\underline{p_A}) - \frac{\|\delta\|_2}{\sigma}\right) > \frac{1}{2}$$

Robustness Guarantee Intuition

If inference with gaussian smoothing predicts “panda” 98% accuracy params: N(0, σ)





Robustness Guarantee

- Assume: $p_a \in (½, 1]$
- Paper demonstrates a radius R
- within which even if perturbed gaussian smoothed classifier predicts class C_A with probability over ½
- Two dimensional example
- Proof holds for N dimensions

$$\mathbb{P}(f(x+\varepsilon) = c_A) \geq \underline{p_A} \geq \overline{p_B} \geq \max_{c \neq c_A} \mathbb{P}(f(x+\varepsilon) = c)$$

Then $g(x+\delta) = c_A$ *for all* $\|\delta\|_2 < R$, *where*

$$R = \frac{\sigma}{2}(\Phi^{-1}(\underline{p_A}) - \Phi^{-1}(\overline{p_B}))$$