Adversarial examples and stability of neural networks

SITE Conference: Long Time Behavior and Singularity Formation in PDEs-Part III (June 13-17, 2021)
New York University Abu Dhabi

Hatem Hajri

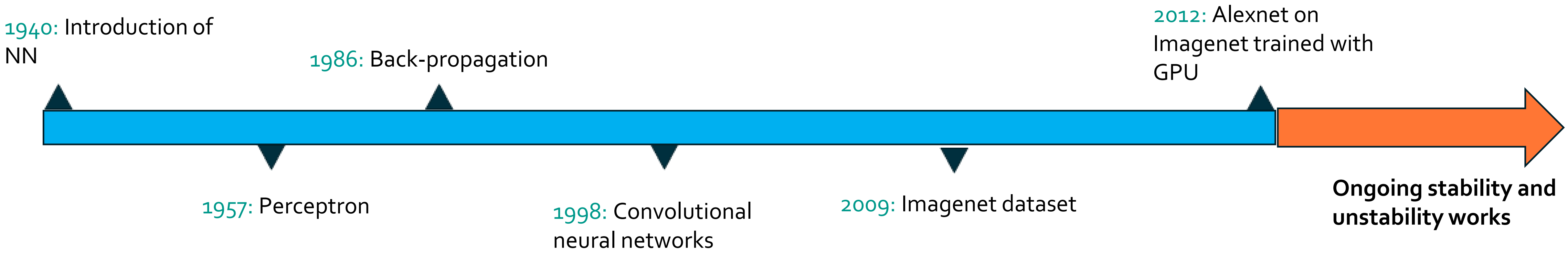
IRT SystemX

Plan

- Neural networks
- Unstability of NN
- Towards stabilising NN

Neural networks

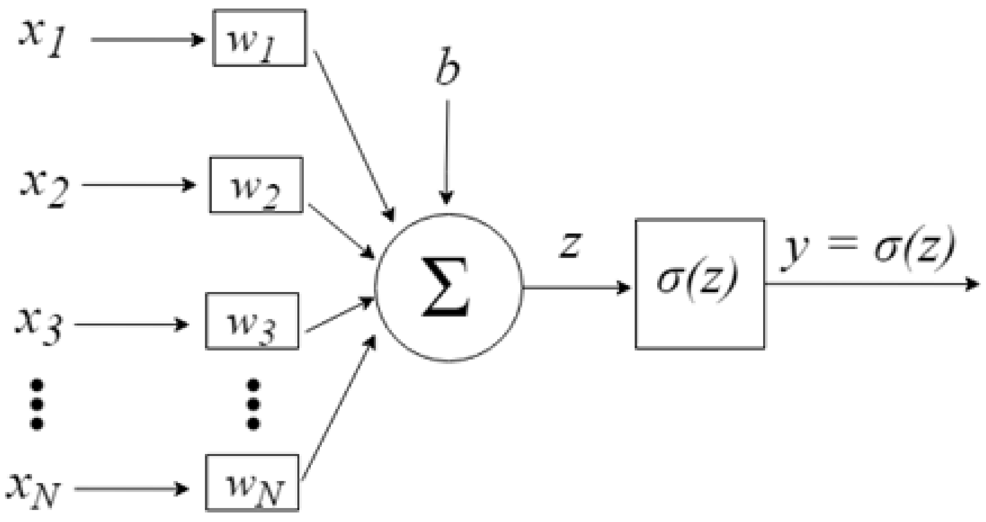
Some key events:



Perceptrons:

- An artificial neuron is a function f of the input x = ($\boldsymbol{x_1}$,…, $\boldsymbol{x_N}$) weighted by a vector of connection weights $\boldsymbol{w}$ = ($\boldsymbol{w_1}$,…, $\boldsymbol{w_N}$), completed by a neuron bias $\boldsymbol{b}$, and associated to an activation function ϕ , namely

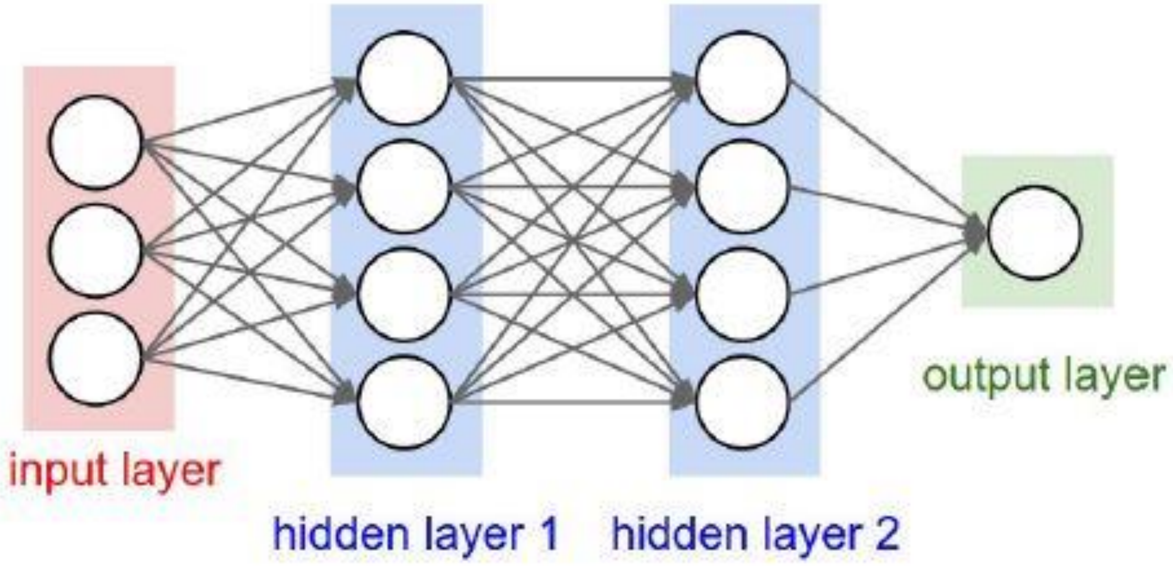
$$\boldsymbol{y} = \boldsymbol{\sigma}(<x, \boldsymbol{w}> + \boldsymbol{b})$$



- Several activation functions can be considered:
  - Id: $\boldsymbol{\sigma}$(x)=x, Sigmoid: $\boldsymbol{\sigma}$(x)=1/(1+e^{-x}), Tan: $\boldsymbol{\sigma}$(x)=tanh(x), ReLu: $\boldsymbol{\sigma}$(x)=max(x,0)

Neural networks:

- A multilayer perceptron is a structure composed by several hidden layers of neurons where the output of a neuron of a layer becomes the input of a neuron of the next layer.



A NN with two hidden layers and an output layer of dimension 1

Neural network classifiers:

- NN classifiers are used to predict the class of an input x among a family of given possible classes $c_1, \ldots, c_K$.

Neural network classifiers:

- NN classifiers are used to predict the class of an input x among a family of given possible classes $c_1 ,\dots, c_K$.

- They are given by the succession of hidden layers and the last layer is normalized using the transformation

$$p_i = \mathrm{softmax}_i(z) = \exp(z_i)/\sum_j \exp(z_j) \in [0,1]$$

Neural network classifiers:

- NN classifiers are used to predict the class of an input x among a family of given possible classes $c_1, \ldots, c_K$.

- They are given by the succession of hidden layers and the last layer is normalized using the transformation

$$p_i = \mathrm{softmax}_i(z) = \exp(z_i) / \sum_j \exp(z_j) \in [0,1]$$

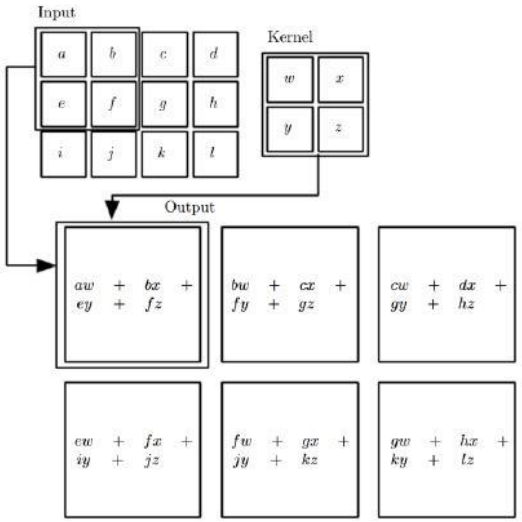
- The final outputs are K probabilities $p_1, \ldots, p_K$ and the predicted class is

$$\mathrm{Class}(x) = \mathrm{argmax}_i \, p_i(x)$$

Convolutional neural networks:

- **Matrix multiplications are replaced with convolutions:**

Convolution by a small kernels

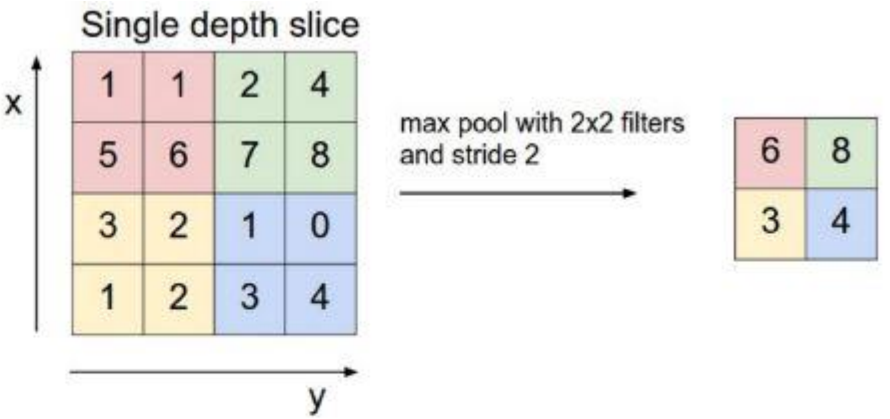


Convolution by a kernel

➡

- Extract specific features from each image by compressing them to reduce their initial size

Pooling operations

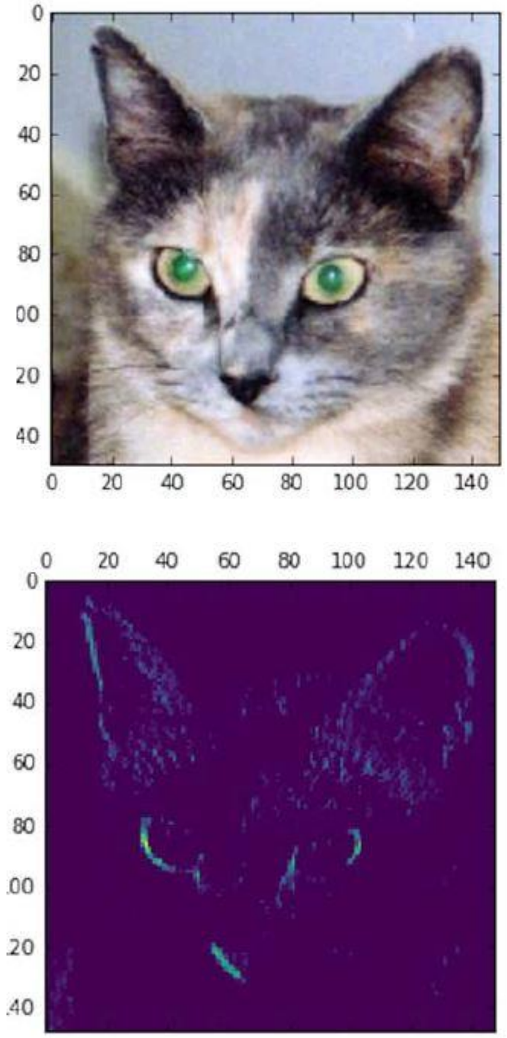


Max pooling operations

➡

- Summarize data and reduce complexity
- Less sensitivity to small translations

Input/output of a convolutional layer



Training Neural networks (Back-propagation):

- ***Problem:*** ***Given a family of training data (x_i, c_i), find the optimal weights (matrices for multiplayer perceptron NN) or kernels for convolutional NN that give the highest prediction (accuracy) : c_i is the class of x_i.***

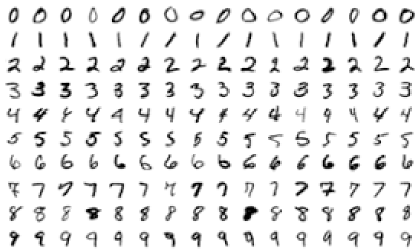
Training Neural networks (Back-propagation):

- ***Problem:*** ***Given a family of training data (x_i, c_i), find the optimal weights (matrices for multiplayer perceptron NN) or kernels for convolutional NN that give the highest prediction (accuracy) : c_i is the class of x_i.***
- **Solution (BP algorithm):**
  - Choose an architecture
  - Initialize weights/kernels W
  - For every (x_i c_i), *make a small update on W (in the direction to maximize p_c):*

$$\boldsymbol{W} \longleftarrow \mathrm{W}+\epsilon \cdot \operatorname{sign}(\nabla_W \, \boldsymbol{p_c}(x, W)\,)$$

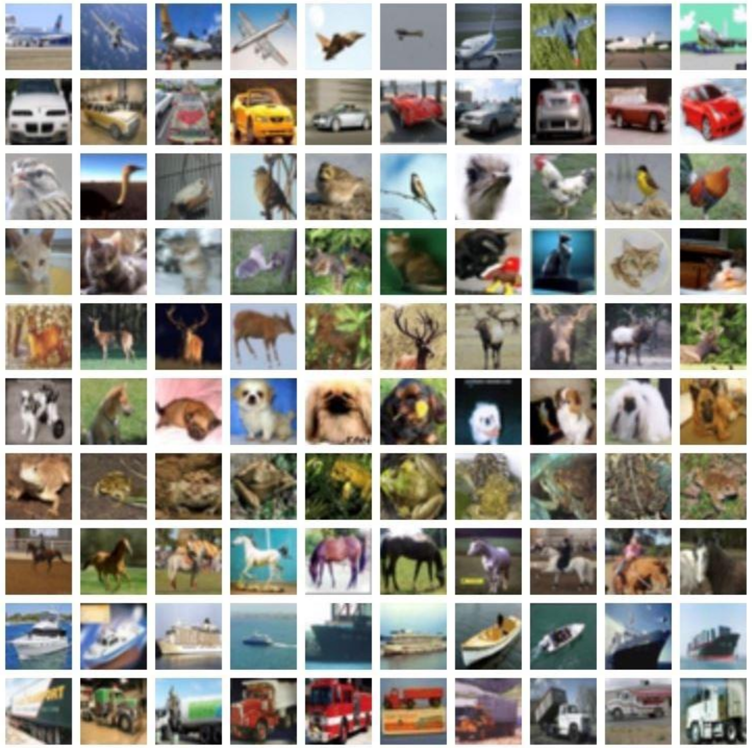
Datasets

MNIST



70 000 images of 28x28 pixel handwritten digits

CIFAR-10

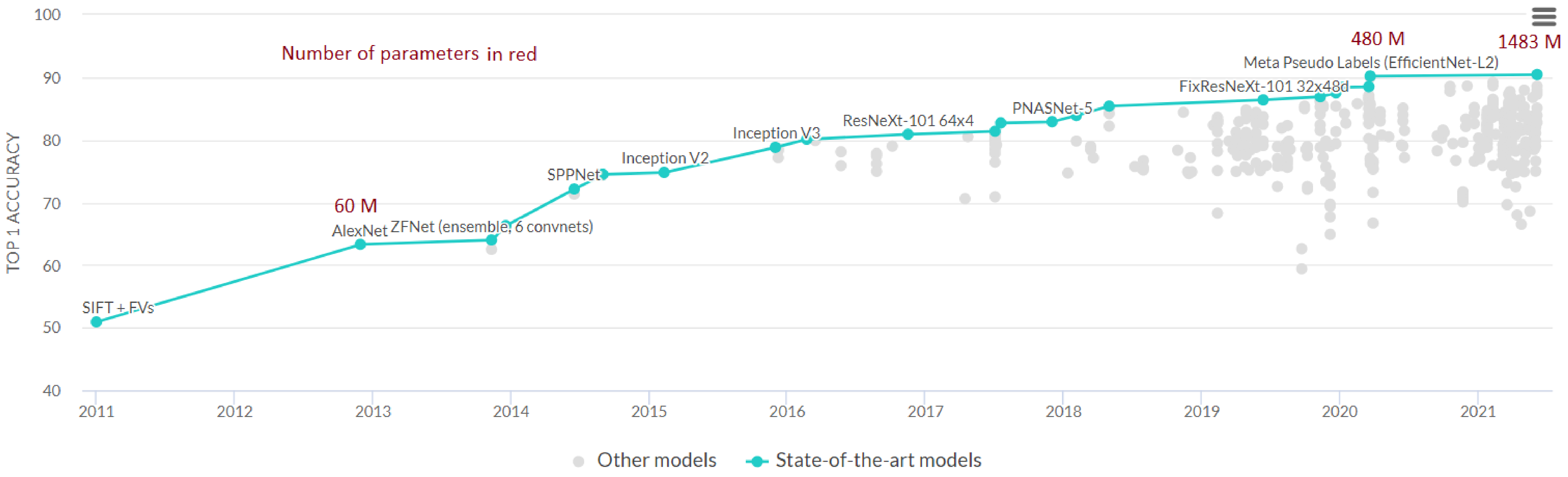


60000 RGB images 32x32x3 in 10 classes

IMAGENET

More than 14 million high-resolution and hand-annotated images into 1000 classes

The challenge of training NN:



State-of-the-art performances on IMAGENET

Unstability of NN

Intriguing properties of neural networks (Szegedy et al. December 2013):

- **Are neural networks stable?**

Intriguing properties of neural networks (Szegedy et al. December 2013):

- **Are neural networks stable?**
- ***Problem formulation: Given (x, c) solve***

$$\mathbf{Min}_r\ \boldsymbol{\varepsilon}\,||r||_2 + p_c(x + r)$$ ***(simultaneously decrease p_c and $||r||_2$)***

Intriguing properties of neural networks (Szegedy et al. December 2013):

- **Are neural networks stable?**
- ***Problem formulation: Given (x, c) solve***

  $\mathbf{Min}_r\ \boldsymbol{\varepsilon}\, //r//_2 + p_c(x + r)$ ***(simultaneously decrease p_c and $//r//_2$)***
- **This is the notion of an adversarial attack: The attack is successful if and only if**

$$x_{adv} := x + r_{optimal}$$

changes the class of x.

Intriguing properties of neural networks (Szegedy et al. December 2013):

- **Are neural networks stable?**
- ***Problem formulation: Given (x, c) solve***

  $$\mathbf{Min}_r\ \boldsymbol{\varepsilon}\,||r||_2 + p_c(x + r)$$ ***(simultaneously decrease p_c and $||r||_2$)***
- **This is the notion of an adversarial attack: The attack is successful if and only if**

  $$x_{adv} := x + r_{optimal}$$

  changes the class of x.
- **The problem is solved by stochastic gradient descent**

Intriguing properties of neural networks (Szegedy et al. December 2013):

- **Adversarial attacks as previously formulated are 100% successful. Adversarial examples constructed on one Neural network tend to be successful on other architectures → Adversarial examples transfer well.**

— Succes rate — transferability

	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

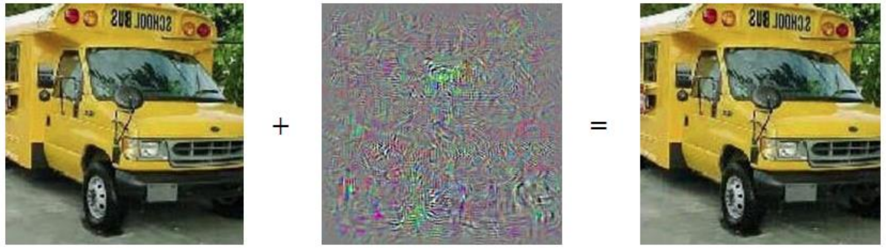
Intriguing properties of neural networks (Szegedy et al. December 2013): Results

- **Adversarial attacks as previously formulated are 100% successful. Adversarial examples constructed on one Neural network tend to be successful on other architectures → Adversarial examples transfer well.**

— Succes rate — transferability

	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

- **Adversarial examples are imperceptible to humans.**



Schoolbus + Perturbation (rescaled for visualization) = Ostrich

Explaining and Harnessing adversarial examples (Goodfellow et al. December 2014):

- **Adversarial attacks are much easier to construct: After training the network, for each (x,c), do one gradient step to decrease p_c :**

$$x_{adv} := x \;-\; \epsilon \cdot \mathrm{sign}(\nabla_x\, \boldsymbol{p_c}(x, W)\,)$$ ***(minimize p_c)***

Explaining and Harnessing adversarial examples (Goodfellow et al. December 2014):

- **Adversarial attacks are much easier to construct: After training the network, for each (x,c), do one gradient step to decrease p_c :**

$$x_{adv} := x - \epsilon \cdot \mathrm{sign}(\nabla_x \, \boldsymbol{p}_c(x, W))$$

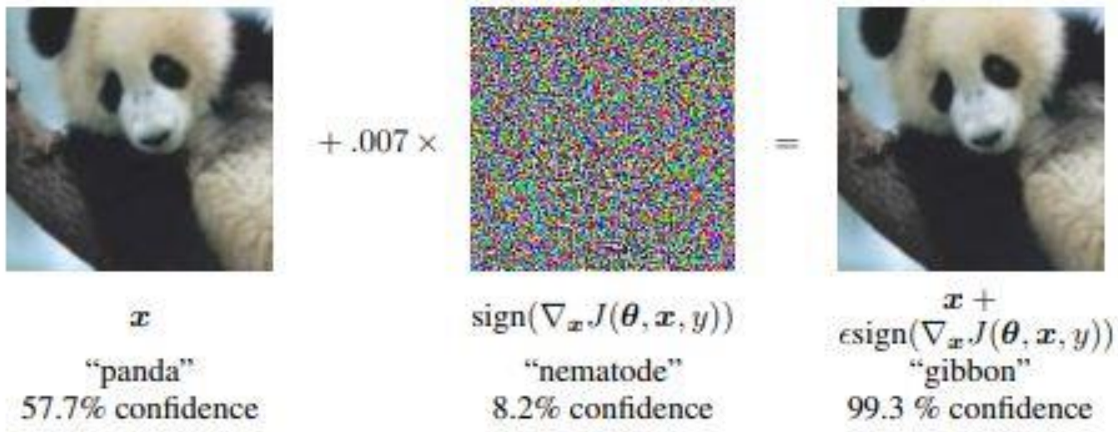
- **This method is called Fast gradient sign method: FGSM. It is 100% successful on IMAGENET even for small ϵ.**

Explaining and Harnessing adversarial examples (Goodfellow et al. December 2014):

- **Adversarial attacks are much easier to construct: After training the network, for each (x,c), do one gradient step to decrease $\boldsymbol{p_c}$:**

$$x_{adv} := x \; - \; \epsilon \cdot \mathrm{sign}(\nabla_x \, \boldsymbol{p_c}(x, W)\,)$$

- **This method is called Fast gradient sign method: FGSM. It is 100% successful on IMAGENET even for small ϵ.**
- ***The famous Panda example on IMAGENET:***



A terminology related to attacks :

- **Distance metrics between x and x_{adv}: $D(x, x_{adv})$**
  - L_0 norm: the number of elements in x_{adv} such that $x^i \neq x^i_{adv}$
  - L_2 , L_∞ norms

A terminology related to attacks :

- **Distance metrics between x and x_{adv}: $D(x, x_{adv})$**
  - L_0 norm: the number of elements in x_{adv} such that $x^i \neq x_{adv}^i$
  - L_2 , L_∞ norms
- **Attack 1 is stronger than Attack 2 in the L_p distance if it has more ability to generate successful $x_{adv} \in B_p(x, \epsilon)$**

A terminology related to attacks :

- **Distance metrics between x and x_{adv}: $D(x, x_{adv})$**
  - L_0 norm: the number of elements in x_{adv} such that $x^i \neq x^i_{adv}$
  - L_2 , L_∞ norms
- **Attack 1 is stronger than Attack 2 in the L_p distance if it has more ability to generate successful $x_{adv} \in B_p(x, \epsilon)$**
- **Attacks can be targeted or untargeted (the class of x_{adv} is given or not)**

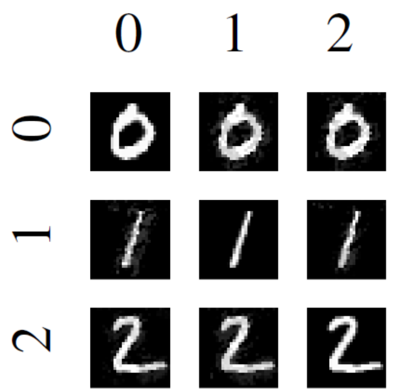
A terminology related to attacks :

- **Distance metrics between x and x_{adv}: $D(x, x_{adv})$**
  - L_0 norm: the number of elements in x_{adv} such that $x^i \neq x_{adv}^i$
  - L_2 , L_∞ norms
- **Attack 1 is stronger than Attack 2 in the L_p distance if it has more ability to generate successful $x_{adv} \in B_p(x, \epsilon)$**
- **Attacks can be targeted or untargeted (the class of x_{adv} is given or not)**
- **What are the best attacks? Hope: The attack is unsuccessful is equivalent to the model is robust.**

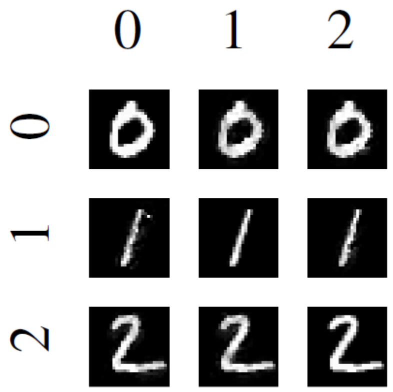
Towards Evaluating the Robustness of Neural Networks (Carlini et al. August 2016)

- ***Carlini-Wagner (CW) attacks are the best L_0, L_2, L_∞ attacks (in 2016).***
  - By considering the outputs of the last-to-one layer one can decrease/increase more efficiently p_c .
  - L_2 attacks are generated following Szegedy et al.
  - L_∞ and L_0 attacks are generated using approximations by differentiable functions of the L_∞ and L_0 *norms.*
  - *Examples of CW targeted attacks on MNIST:*

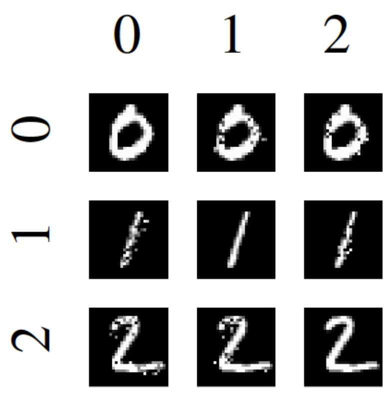
L_∞ attack



L_2 attack



L_0 attack



Towards Deep Learning Models Resistant to Adversarial Attacks (Madry et al. June 2017)

- ***Projected gradient descent (PGD) attack*** **is an extension of FGSM, where after each step of perturbation, the adversarial example is projected back onto the ϵ-ball of x using a projection function Π**

$$x_{adv}^{t} = \Pi_{\epsilon}\left(x^{t-1} - \alpha \cdot \mathrm{sign}(\nabla_x \boldsymbol{p_c}(x^{t-1}, W))\right)$$

Towards Deep Learning Models Resistant to Adversarial Attacks (Madry et al. June 2017)

- ***Projected gradient descent (PGD) attack*** **is an extension of FGSM, where after each step of perturbation, the adversarial example is projected back onto the ϵ-ball of x using a projection function Π**

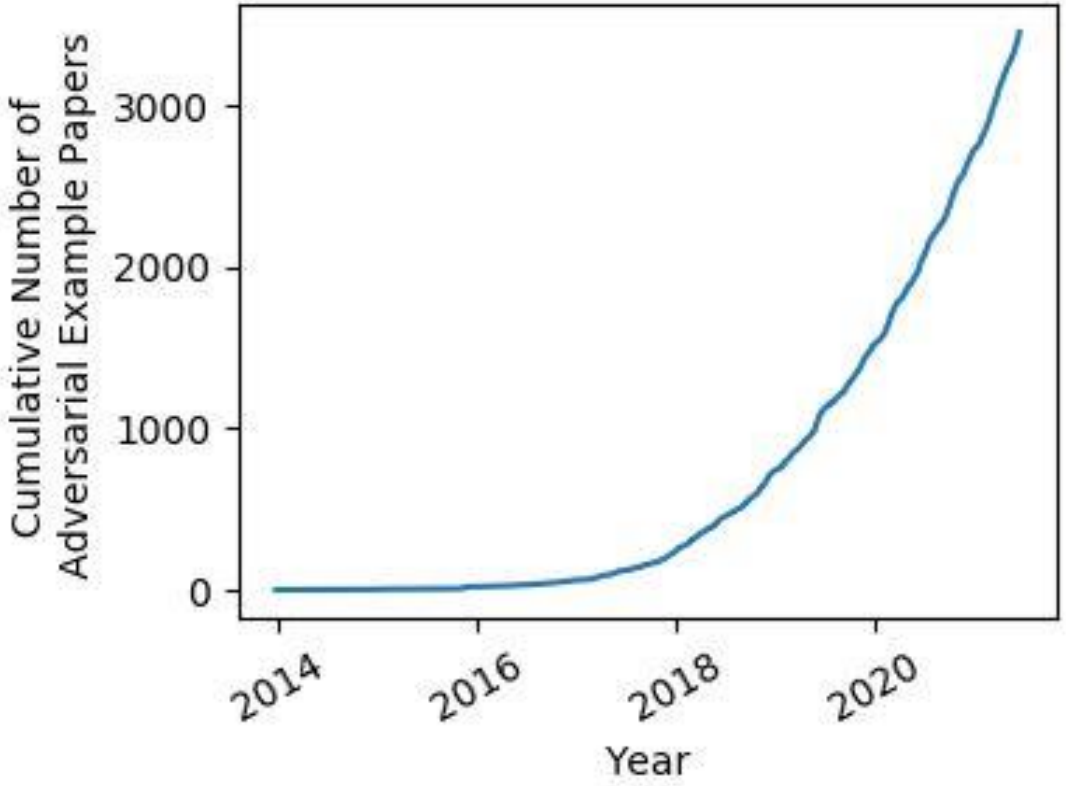
$$x_{adv}^{t} = \Pi_{\epsilon}\left(x^{t-1} - \alpha \cdot \mathrm{sign}(\nabla_{x}\, \boldsymbol{p}_{\boldsymbol{c}}(x^{t-1}, W))\right)$$

- **PGD is regarded as the strongest L_{∞} attack**

One Pixel Attack for Fooling Deep Neural Networks (Su et al. October 2017)

- **One pixel attacks are more spectacular: only one pixel is allowed to be changed.**
- **Inspired from genetic algorithms:**
  - Randomly fix candidate pixels $\{X_i\}$
  - Mutate each X_i as follows: $\text{mutation}(X_i) = X_i + \lambda(X_k - X_l)$ (k and l are random candidate indices)
  - Choose between X_i and $\text{mutation}(X_i)$ according to which pixel decreases the most the current probability.

Adversarial attacks papers



Key takeaways:

- CW, PGD are the most powerful attacks. There has been very slight improvements since then.
- L_0, L_2, L_∞ are generally imperceptible.
- More perceptible attacks have also been studied: e.g. attacks by adding foreign objects (patches, stickers), by changing the background of the image (semantic) etc.

An attack by adding stickers: picture from Robust physical world attacks on deep learning models

Towards stabilising NN

First:

- The accuracy of a model is the fraction of inputs which are correctly classified

First:

- The accuracy of a model is the fraction of inputs which are correctly classified
- The ϵ-robustness score (also depending on the L_p norm) is the faction of inputs x such that class(x)=class(y) for all $\mathbf{y} \in \mathrm{B}_p(x,\epsilon)$

First:

- **The accuracy** of a model is the fraction of inputs which are correctly classified
- The ϵ-**robustness score** (also depending on the L_p norm) is the faction of inputs x such that **class(x)=class(y) for all y** ∈ **B**$_p(x, \epsilon)$
- Adversarial examples have shown that highly accurate models may have **zero robustness scores.**

Towards Deep Learning Models Resistant to Adversarial Attacks (Madry et al. June 2017)

- **Classical training tries to solve for each (x,c) :**

Max$_W$ $p_c(x, W)$ (weights maximising p_c)

Towards Deep Learning Models Resistant to Adversarial Attacks (Madry et al. June 2017)

- **Classical training tries to solve for each (x,c) :**

$$\mathbf{Max}_W \ \boldsymbol{p_c}(x, W) \text{ (weights maximising } \boldsymbol{p_c}\text{)}$$

- **To target ϵ-robust networks, Madry et al. proposes to solve:**

$$\mathbf{Max}_W \ \mathbf{Min}\{ \boldsymbol{p_c}(y, W), y \in \mathrm{B}_p(x,\epsilon)\} \text{ (weights maximising } \boldsymbol{p_c} \text{ uniformly)}$$

Towards Deep Learning Models Resistant to Adversarial Attacks (Madry et al. June 2017)

- **Classical training tries to solve for each (x,c) :**

$\mathbf{Max}_W\, \boldsymbol{p_c}(x, W)$ **(weights maximising $\boldsymbol{p_c}$)**

- **To target ϵ-robust networks, Madry et al. proposes to solve:**

$\mathbf{Max}_W\ \mathbf{Min}\{\ \boldsymbol{p_c}(y, W),\ y \in \mathbf{B}_p(x, \epsilon)\}$ **(weights maximising $\boldsymbol{p_c}$ uniformly)**

- **Problem:** How to compute $\mathbf{Min}\{\ \boldsymbol{p_c}(y, W),\ y \in \mathbf{B}_p(x, \epsilon)\}$ **?**

Towards Deep Learning Models Resistant to Adversarial Attacks (Madry et al. June 2017)

- **Solution: Adversarial training: for each (x,c)**
  - Find a good $x_{adv} := \mathbf{M}\text{in}\{\, p_{\boldsymbol{c}}(y, W),\ y \in \mathrm{B}_{\boldsymbol{p}}(x,\epsilon)\}$ by gradient ascent using PGD.
  - Once x_{adv} is found, update W by gradient ascent solving $\mathbf{Max}_{W}\, p_{\boldsymbol{c}}(\, x_{adv}, W)$

Towards Deep Learning Models Resistant to Adversarial Attacks (Madry et al. June 2017)

- **Solution: Adversarial training: for each (x,c)**
  - Find a good $x_{adv} := \mathbf{Min}\{\, p_c(y, W),\ y \in B_p(x,\epsilon)\}$ by gradient ascent using PGD.
  - Once x_{adv} is found, update W by gradient ascent solving $\mathbf{Max}_W\, p_c(x_{adv}, W)$
- **Validation (empirically):**

By showing that PGD and CW are significantly less successful on adversarially trained networks (for the first time)

Since 2017 no attack has been able to find adversarial examples for the 45.8 robust samples inside the L_∞ ball of radius ϵ =0.031.

Experiments on CIFAR-10

Accuracy of the adv. train. model

Method	Steps	Source	Accuracy
Natural	-	-	87.3%
FGSM	-	A	56.1%
PGD	7	A	50.0%
PGD	20	A	**45.8%**
CW	30	A	46.8%

Accuracy = 100 - performance of the attack

L_∞ $\varepsilon = 0.031$.

Empirical defense techniques

- Adversarial training is an empirical defense technique.
- Many empirical defense techniques have been presented but either they were completely broken or shown to be less efficient than adversarial training.

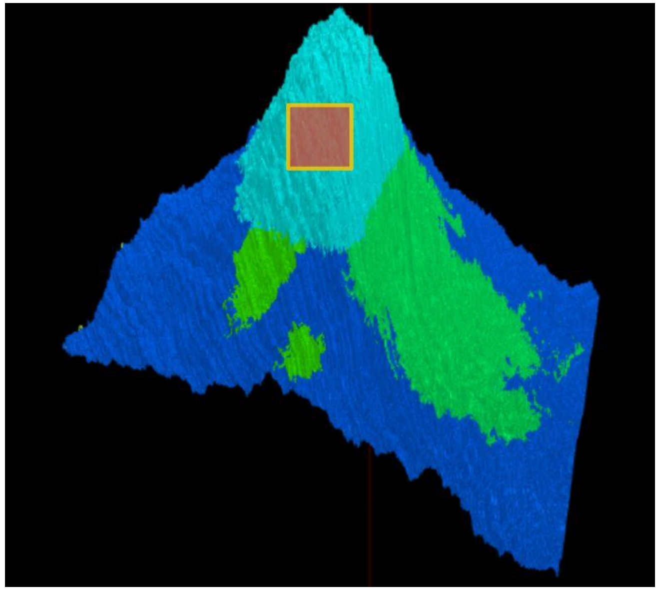
Break defenses: Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples (Athalye et al. Feb 2018)

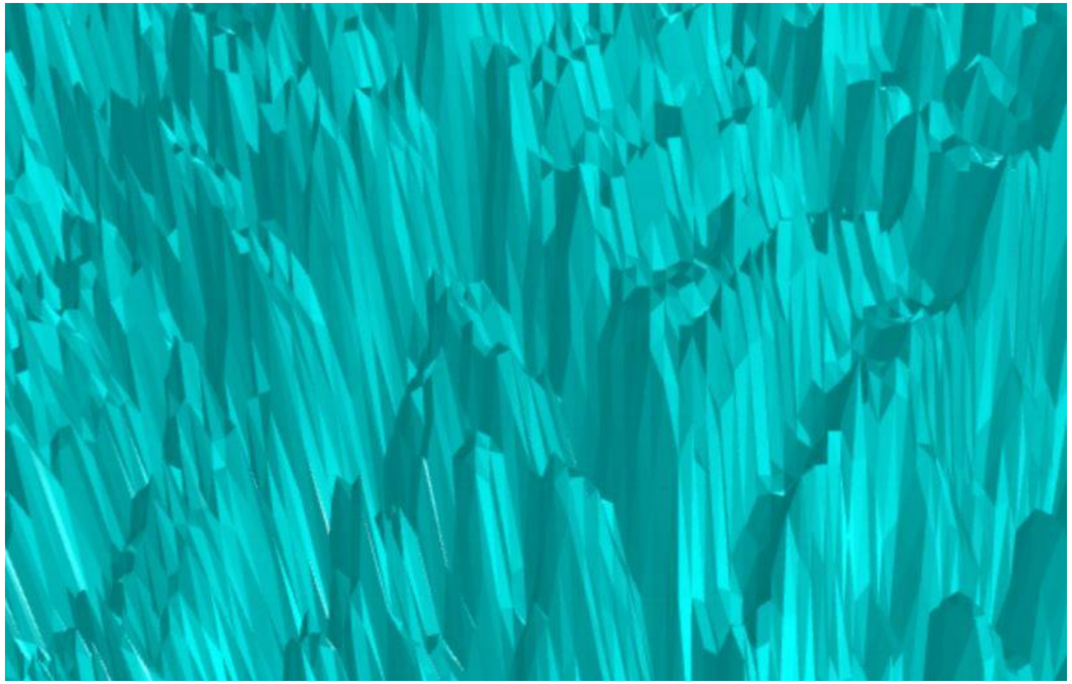
- **Many defense techniques rely on obfuscated gradients: gradients are incorrect as a consequence of non differentiable operations or unstable.**

Break defenses: Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples (Athalye et al. Feb 2018)

- **Many defense techniques rely on obfuscated gradients: gradients are incorrect as a consequence of non differentiable operations or unstable.**
- **Due to obfuscated gradients, many defense techniques provide apparent robustness against powerful attacks such as PGD, CW etc.**

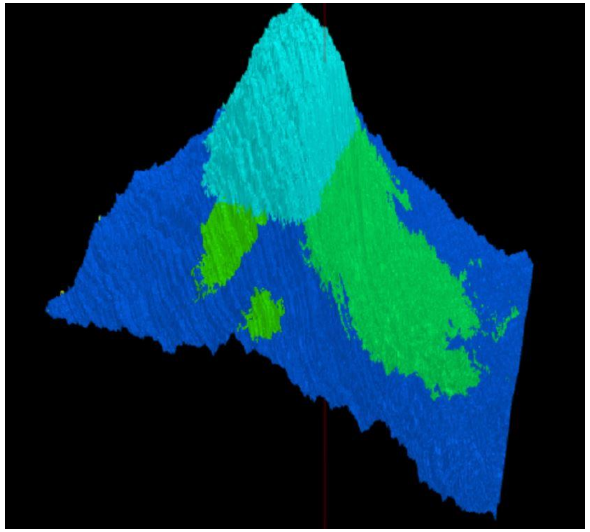
An illustration
of obfuscated gradients

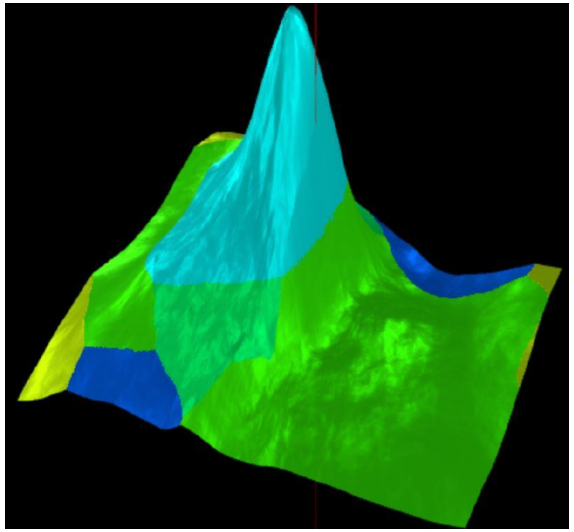




Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples (Athalye et al. Feb 2018)

- **Solution:** use smoothed gradients in attacking:





- **Results:** Seven defense techniques (already published) are broken:

broken

Defense	Dataset	Distance	Accuracy
Buckman et al. (2018)	CIFAR	0.031 (ℓ_∞)	0%*
Ma et al. (2018)	CIFAR	0.031 (ℓ_∞)	5%
Guo et al. (2018)	ImageNet	0.005 (ℓ_2)	0%*
Dhillon et al. (2018)	CIFAR	0.031 (ℓ_∞)	0%
Xie et al. (2018)	ImageNet	0.031 (ℓ_∞)	0%*
Song et al. (2018)	CIFAR	0.031 (ℓ_∞)	9%*
Samangouei et al. (2018)	MNIST	0.005 (ℓ_2)	55%**
Madry et al. (2018)	CIFAR	0.031 (ℓ_∞)	47%
Na et al. (2018)	CIFAR	0.015 (ℓ_∞)	15%

less

in another paper

NATTACK: Learning the Distributions of Adversarial Examples for an Improved Black-Box Attack on Deep Neural Networks(Li et al. May 2019): A simple way to break obfuscated gradient defenses

- **Apply attacks that do not rely on the gradient of the NN.**
  - Fix ϵ and minimize F(μ)= $\mathrm{E}[p_c\,(x+\mu+\epsilon\,\mathcal{N}(0,\mathrm{I})]$ over μ by gradient descent.

NATTACK: Learning the Distributions of Adversarial Examples for an Improved Black-Box Attack on Deep Neural Networks(Li et al. May 2019): A simple way to break obfuscated gradient defenses

- **Apply attacks that do not rely on the gradient of the NN.**
  - Fix ϵ and minimize F(μ)= $\mathrm{E}[p_c\,(x+\mu+\epsilon\,\mathcal{N}(0,\mathrm{I})]$ over μ by gradient descent.
  - Once μ is found, sample many $x_{adv} := x + \mu + \epsilon\,\mathcal{N}(0,\mathrm{I})$ and choose the best x_{adv}.

NATTACK: Learning the Distributions of Adversarial Examples for an Improved Black-Box Attack on Deep Neural Networks(Li et al. May 2019): A simple way to break obfuscated gradient defenses

- **Apply attacks that do not rely on the gradient of the NN.**
  - Fix ϵ and minimize F(μ)= $\mathrm{E}[p_c\,(\mathrm{x}+\mu+\epsilon\,\mathcal{N}(0,\mathrm{I})]$ over μ by gradient descent.
  - Once μ is found, sample many $x_{adv} := x + \mu + \epsilon\,\mathcal{N}(0,\mathrm{I})$ and choose the best x_{adv}.
  - An important point: The gradient $\nabla_\mu \mathrm{F}(\mu)$ does not require to compute $\nabla_\mu p_c$ but only ∇_μ of the Gaussian kernel.

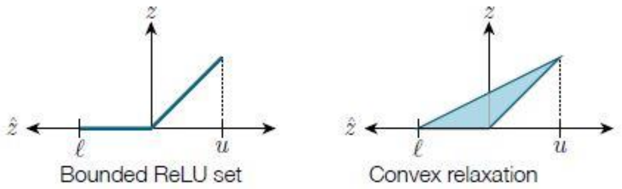
Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope (Wong et al.Nov 2017)

- **Can we develop defense techniques that have provable robustness properties (theoretical guarantees that any attack will not be successful)?**
- Define the polytope for a given (x,c) as $\mathcal{P} = N(B_\infty(x, \epsilon))$ **the image by the network.**

Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope (Wong et al.Nov 2017)

- **Can we develop defense techniques that have provable robustness properties (theoretical guarantees that any attack will not be successful)?**
- Define the polytope for a given (x,c) as $\mathcal{P} = N(B_\infty(x, \epsilon))$ **the image by the network.**
- $\mathcal{P}$ is a geometrically complicated space. The idea is to find a convex set $\mathcal{C}$ such that $\mathcal{P} \subseteq \mathcal{C}$ and then provide a condition under which $\mathcal{C}$ will not contain adversarial examples (in the image space).
- For ReLu(x)=max(x,0), we use

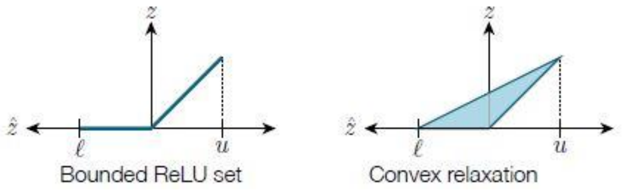
the convex relaxation:



Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope (Wong et al.Nov 2017)

- **Can we develop defense techniques that have provable robustness properties (theoretical guarantees that any attack will not be successful)?**
- Define the polytope for a given (x,c) as $\mathcal{P}$= N $(B_\infty(x,\epsilon))$ **the image by the network.**
- $\mathcal{P}$ is a geometrically complicated space. The idea is to find a convex set $\mathcal{C}$ such that $\mathcal{P} \subseteq \mathcal{C}$ and then provide a condition under which $\mathcal{C}$ will not contain adversarial examples (in the image space).
- For ReLu(x)=max(x,0), we use

the convex relaxation:



- This gives an outer convex bound:

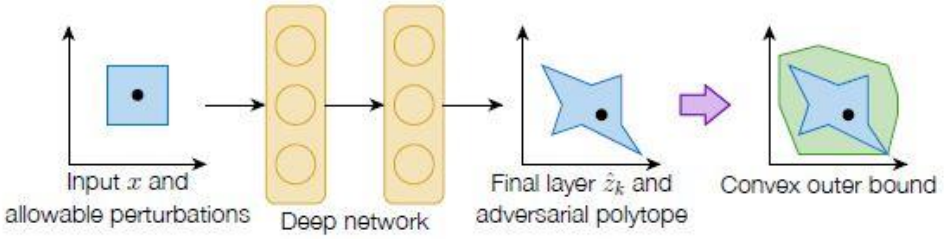


Figure 1. Conceptual illustration of the (non-convex) adversarial polytope, and an outer convex bound.

Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope (Wong et al.Nov 2017)

- We deduce the bound:

$p_c(\mathbf{y}_*, W) \leq \mathrm{Min}\{\, p_c(y, W) : y \in \mathcal{C} \} \leq \mathrm{Min}\{\, p_c(y, W) : y \in \mathrm{B}_\infty(\mathbf{x}, \epsilon) \}$

$\mathbf{y}_* \in \mathcal{C}$ *is a worst case point which can be found by convex optimisation.*

- Following adversarial training, a neural network can be trained by solving for each (x,c): $\mathrm{Max}_W\, p_c(\mathbf{y}_*, W)$
- In addition, Under some analytic condition involving $\mathbf{y}_*$, there does not exist any $x_{adv} \in \mathrm{B}_\infty(x, \epsilon)$.
- **Results:**

Table 1. Error rates for various problems and attacks, and our robust bound for baseline and robust models.

A model trained normaly / adversarially

PROBLEM	ROBUST	ϵ	TEST ERROR	FGSM ERROR	PGD ERROR	ROBUST ERROR BOUND
MNIST	×	0.1	1.07%	50.01%	81.68%	100%
MNIST	√	0.1	1.80%	3.93%	4.11%	5.82%
FASHION-MNIST	×	0.1	9.36%	77.98%	81.85%	100%
FASHION-MNIST	√	0.1	21.73%	31.25%	31.63%	34.53%
HAR	×	0.05	4.95%	60.57%	63.82%	81.56%
HAR	√	0.05	7.80%	21.49%	21.52%	21.90%
SVHN	×	0.01	16.01%	62.21%	83.43%	100%
SVHN	√	0.01	20.38%	33.28%	33.74%	40.67%

100% of data may contain adversaries

only 5.82% of data may contain adv. 4.11% are sure.

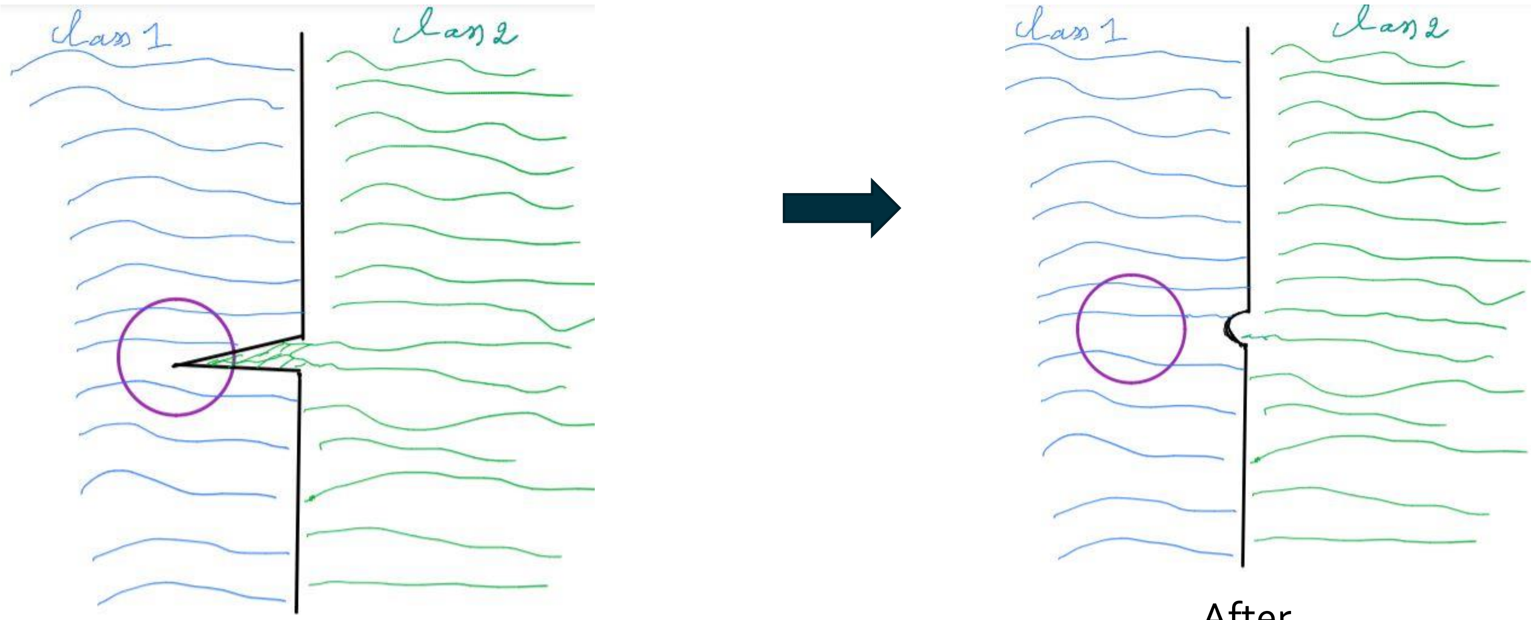
- **Disadavantage:**

scalabity to large datasets.

Certified Adversarial Robustness via Randomized Smoothing (Cohen et al. Feb 2019)

- **The smoothing of a classifier F is:**

$$g(x) = \mathbf{argmax}_i \; P(C(x+\varepsilon) = i\,), \quad \varepsilon \sim \mathcal{N}(0, \boldsymbol{\sigma}^2 \boldsymbol{I})$$



Before

After

Certified Adversarial Robustness via Randomized Smoothing (Cohen et al. Feb 2019)

Main result: Let p_i be the output probabilities of a neural network classifier and $C(x)=\mathbf{argmax}_i\ p_i(x)$. Define, as before:

$$g(x) = \mathbf{argmax}_i\ P(C(x+\varepsilon) = i\ ),\quad \varepsilon \sim \mathcal{N}(0, \sigma^2 I)$$

Let x be an input, $\mathbf{C}_A = g(x)$, $\mathbf{P}_A = P(C(x+\varepsilon) = \mathbf{C}_A)$ and $\mathbf{P}_B = \mathbf{argmax}_i\ P(C(x+\varepsilon) = i\ )$; $i \neq \mathbf{C}_A$.

We have $g(y) = C_A$ for all $y \in B_2(x, R)$ *with*:

$$R = (\sigma/2)\left(\Phi^{-1}(\mathbf{P}_A) - \Phi^{-1}(\mathbf{P}_B)\right)$$

Certified Adversarial Robustness via Randomized Smoothing (Cohen et al. Feb 2019)

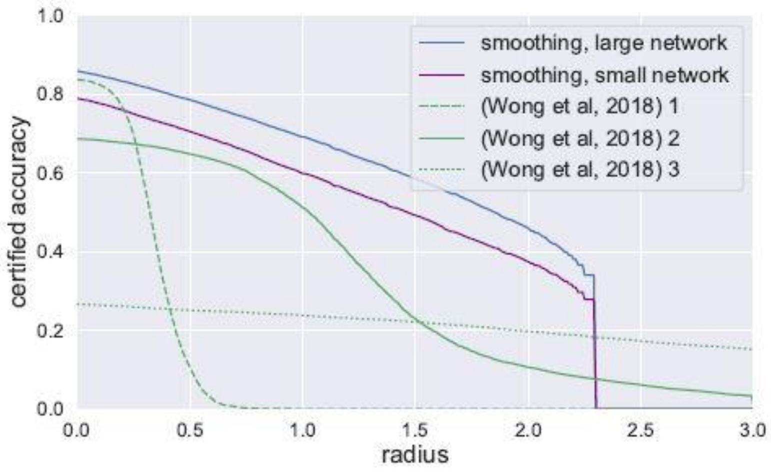
In Practice:

- The smoothed classifier g is estimated with Monte-Carlo.
- Since the estimations of $\mathbf{P}_A$ and $\mathbf{P}_B$ may not be accurate, we rather use an upper and lower bounds of these quantities in the previous theorem (which still holds).
- To improve the results, we also add the Gaussian noise in training.

Results:

ℓ_2 RADIUS	BEST σ	CERT. ACC (%)	STD. ACC(%)
0.5	0.25	49	67

State-of-the-art results on IMAGENET: 49% of samples are certified robust in the L_2 ball of radius 0.5. Accurcay is lower than standard training without smoothing.



Randomised smoothing certifies better than provable defense techniques on CIFAR.

Key takeaways:

- **Adversarial training, provable defenses and randomized smoothing are the only known and efficient defense methods.**
- **Adversarial training is not provably but only empirically robust .**
- **Provable defenses techniques work well for small architectures but scale very poorly to large architectures: The outer convex domain becomes much larger than the reachable domain.**
- **Randomized smoothing is the best defense method up to now. Moreover it is very simple to put in place.**
- **Although these methods are the best existing ones, they still certify on only very small/negligible domains.**

Thanks for your attention