Machine Learning for Cyber-security

A Deep Dive on Adversarial Attacks

Alexandre Araujo

October 4, 2023

Who I am ? – Alexandre Araujo

PhD at University Paris-Dauphine – PSL (2017 - 2021)

Subject: Compact and Robust Deep Neural Networks with Toeplitz Matrices

Advisors: Jamal Atif, Yann Chevaleyre and Benjamin Negrevergne

Postdoc at ENS / INRIA Paris (Sept. 2021 – Dec. 2022)

Subject: Computer Vision

Advisors: Jean Ponce, Julien Mairal

Postdoc at NYU (since Jan. 2023)

Subject: Trustworthy AI

Advisors: Siddharth Garg, Farshad Khorrami

Contact: `alexandre.araujo@nyu.edu`

Teaching Assistants

Mingzhi Zhu: mz3379@nyu.edu

Mahmoud Shabana: ms9761@nyu.edu

Chandana S. Yatisha: cs7074@nyu.edu

Summary of the course

1. **Introduction and Background on Convolutional Networks**

Summary of the course

Summary of the course

Summary of the course

Introduction
Background on Convolutional Networks

Context

In the last 10 years, Machine Learning models have been deployed in many **day-to-day applications**.

Context

In the last 10 years, Machine Learning models have been deployed in many **day-to-day applications**.

- **Large scale applications**: web search, translation, vocal assistants, etc.

Context

In the last 10 years, Machine Learning models have been deployed in many **day-to-day applications**.

- **Large scale applications**: web search, translation, vocal assistants, etc.
- **IoT devices**: cameras, headphones, drones, etc.

Context

In the last 10 years, Machine Learning models have been deployed in many **day-to-day applications**.

- **Large scale applications**: web search, translation, vocal assistants, etc.
- **IoT devices**: cameras, headphones, drones, etc.
- **Critical Systems**: self-driving cars, justice, healthcare, etc.

Context

In the last 10 years, Machine Learning models have been deployed in many **day-to-day applications**.

- **Large scale applications**: web search, translation, vocal assistants, etc.
- **IoT devices**: cameras, headphones, drones, etc.
- **Critical Systems**: self-driving cars, justice, healthcare, etc.

Machine Learning models need to be more **trustworthy** to society.

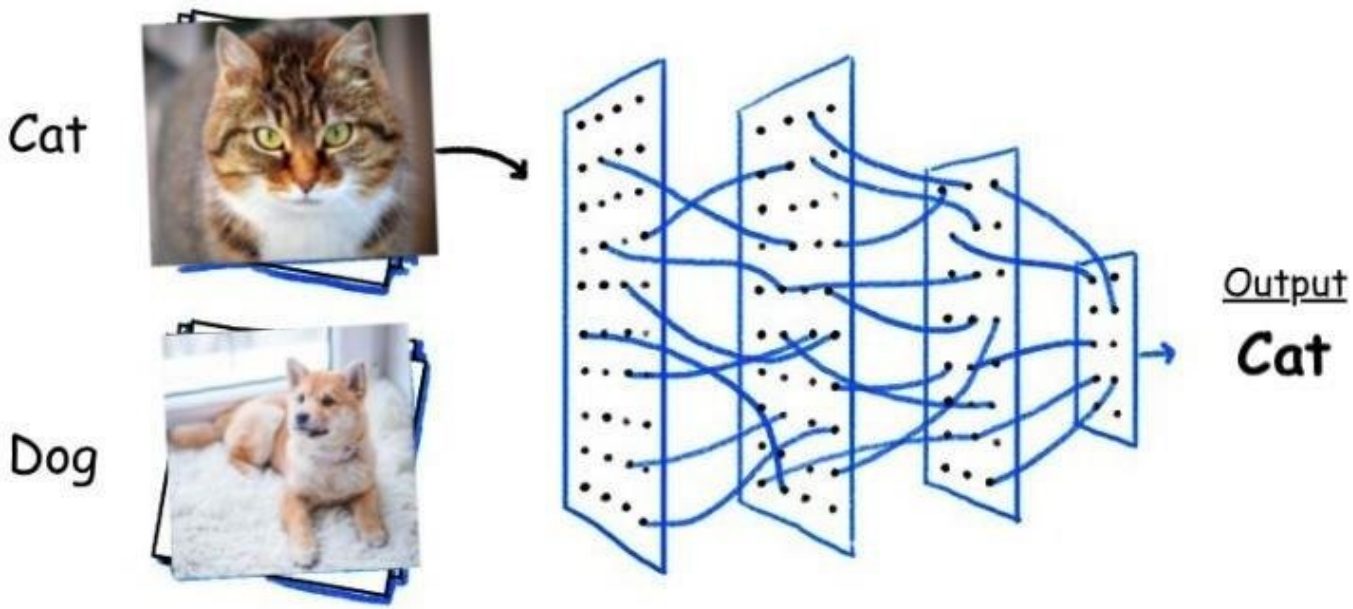
Classification in Machine Learning

Input space $\mathcal{X} \subset \mathbb{R}^d$ to a label space $\mathcal{Y} := \{1, \ldots, K\}$.

Let $f : \mathcal{X} \to \mathbb{R}^K$ be a classifier such that the predicted label for $\mathbf{x}$ is

$$\mathbf{f}(\mathbf{x}) := \arg\max_k [f(\mathbf{x})]_k.$$

Input-label $(\mathbf{x}, y)$ is correctly classified if $\mathbf{f}(\mathbf{x}) = y$.



Supervised Learning Algorithms

Features			Label
$x_1^{(1)}$	$\cdots$	$x_n^{(1)}$	$y^{(1)}$
$x_1^{(2)}$	$\cdots$	$x_n^{(2)}$	$y^{(2)}$
$\vdots$	$\cdots$	$\vdots$	$\vdots$
$x_1^{(m)}$	$\cdots$	$x_n^{(m)}$	$y^{(m)}$

Given a set of m **training examples** let us denote $\mathcal{S} = \left\{\mathbf{x}^{(1)}, y^{(1)}, \ldots, \mathbf{x}^{(m)}, y^{(m)}\right\}$ the training set where $\mathbf{x}^{(i)} \in \mathcal{X}$ is the feature vector and $y^{(i)} \in \mathcal{Y}$ is the corresponding label.

Assumption: there is a function f matching any feature vector to its label.

The goal of a **learning algorithm** is to approximate f by a parameterized function f_Ω. In order to measure how well the function fits, **a loss function** $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ is defined. The standard method to learn the set of parameters Ω is the **empirical risk minimization (ERM)**:

$$R_{ERM} \triangleq \arg\min_{\Omega} \frac{1}{m} \sum_{\mathbf{x}, y \in \mathcal{S}} L(f_\Omega(\mathbf{x}), y)$$

Deep neural networks

Neural Neural can be analytically described as a composition of linear functions interlaced with non-linear functions:

Neural Network

A neural network of p layers is defined as follows:

$$f(\mathbf{x}) = \phi_{\mathbf{W}^{(p)},\mathbf{b}^{(p)}} \circ \rho \circ \phi_{\mathbf{W}^{(p-1)},\mathbf{b}^{(p-1)}} \circ \rho \circ \cdots \circ \rho \circ \phi_{\mathbf{W}^{(1)},\mathbf{b}^{(1)}}(\mathbf{x})$$

where $\phi_{\mathbf{W},\mathbf{b}} \triangleq \mathbf{x} \mapsto \mathbf{W}\mathbf{x} + \mathbf{b}$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^n$, $\mathbf{W} \in \mathbb{R}^{n \times n}$, ρ some non linear functions.

Deep neural networks

Neural Neural can be analytically described as a composition of linear functions interlaced with non-linear functions:

Neural Network

A neural network of p layers is defined as follows:

$$f(\mathbf{x}) = \phi_{\mathbf{W}^{(p)},\mathbf{b}^{(p)}} \circ \rho \circ \phi_{\mathbf{W}^{(p-1)},\mathbf{b}^{(p-1)}} \circ \rho \circ \cdots \circ \rho \circ \phi_{\mathbf{W}^{(1)},\mathbf{b}^{(1)}}(\mathbf{x})$$

where $\phi_{\mathbf{W},\mathbf{b}} \triangleq \mathbf{x} \mapsto \mathbf{W}\mathbf{x} + \mathbf{b}$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^n$, $\mathbf{W} \in \mathbb{R}^{n \times n}$, ρ some non linear functions.

Evaluation of Neural Networks

- Classical evaluation with accuracy
- Robust evaluation against adversarial attacks

Convolutional Neural Networks

Convolutional Neural Networks are state-of-the-art for image classification.

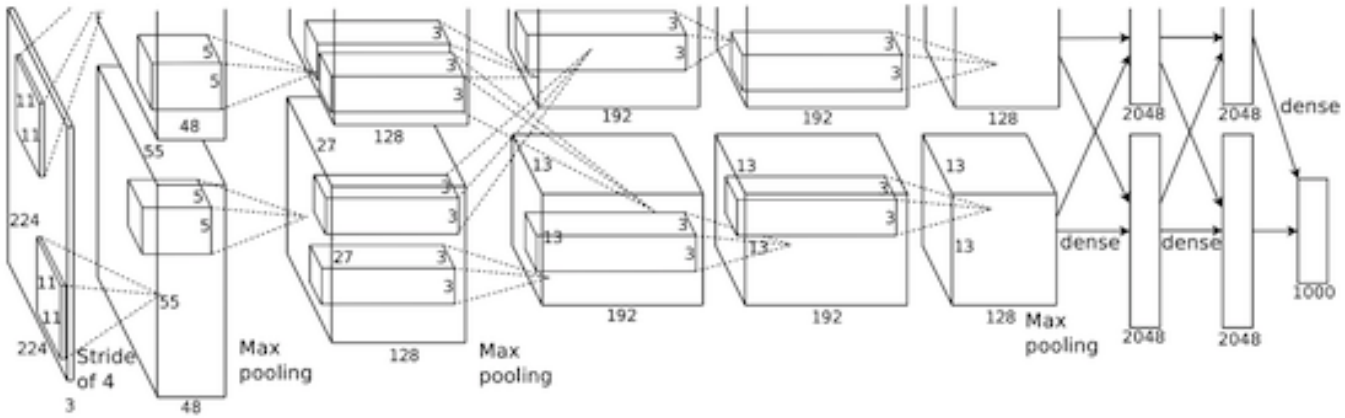


Figure 1: Architecture of AlexNet

Convolutional Neural Networks

Convolutional Neural Networks are state-of-the-art for image classification.

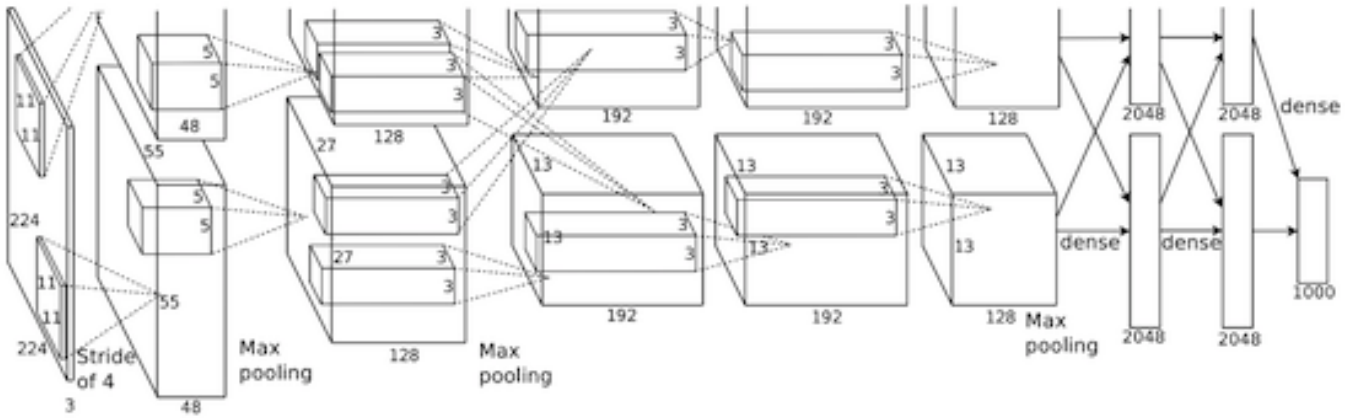
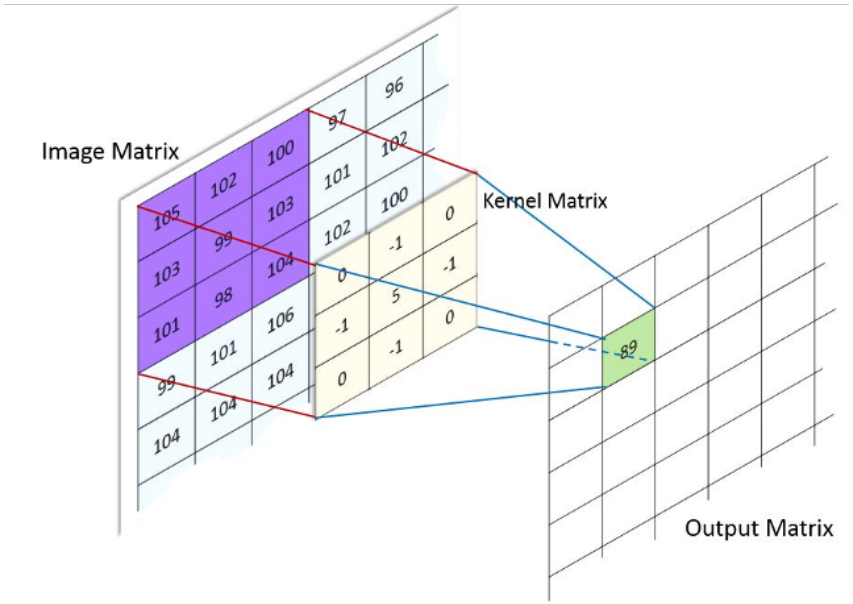


Figure 1: Architecture of AlexNet

Convolutional Neural Networks use a specific **structure as linear operations**.

Convolution as matrix-multiplication



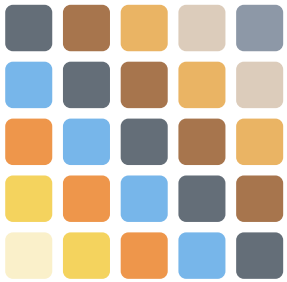
Convolution between a 2-dimensional image and a 2 dimensional kernel

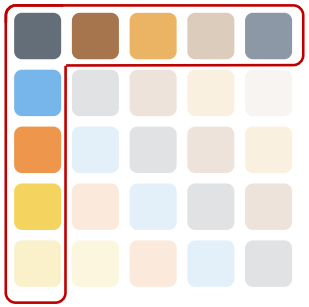
A discrete convolution between a signal $\mathbf{x}$ and a kernel $\mathbf{k}$ can be expressed as a product between the vectorization of $\mathbf{x}$ and a structured matrix.

Structure of Convolutions: matrices from the Toeplitz family

A Toeplitz matrix can be used to compute 1-dimensional discrete convolution. A Toeplitz matrix is a matrix with constant diagonal:

$$\begin{pmatrix} a & b & c & d \\ e & a & b & c \\ f & e & a & b \\ d & f & e & a \end{pmatrix}$$



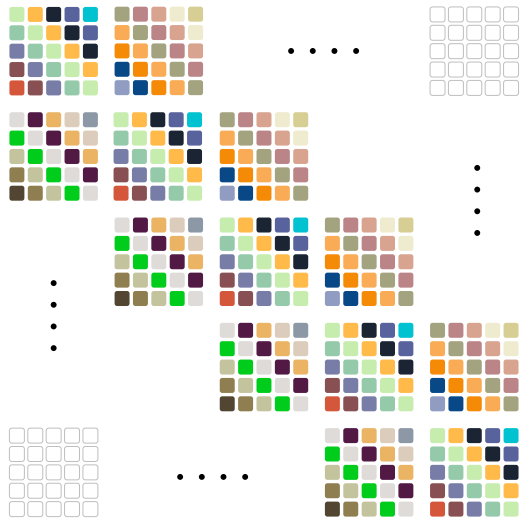


$\Rightarrow$ A $n \times n$ Toeplitz matrix has $2n - 1$ unique values.

Structure of Convolutions: matrices from the Toeplitz family

A block Toeplitz matrix is a matrix which contains **blocks that are repeated down the diagonals** of the matrix.

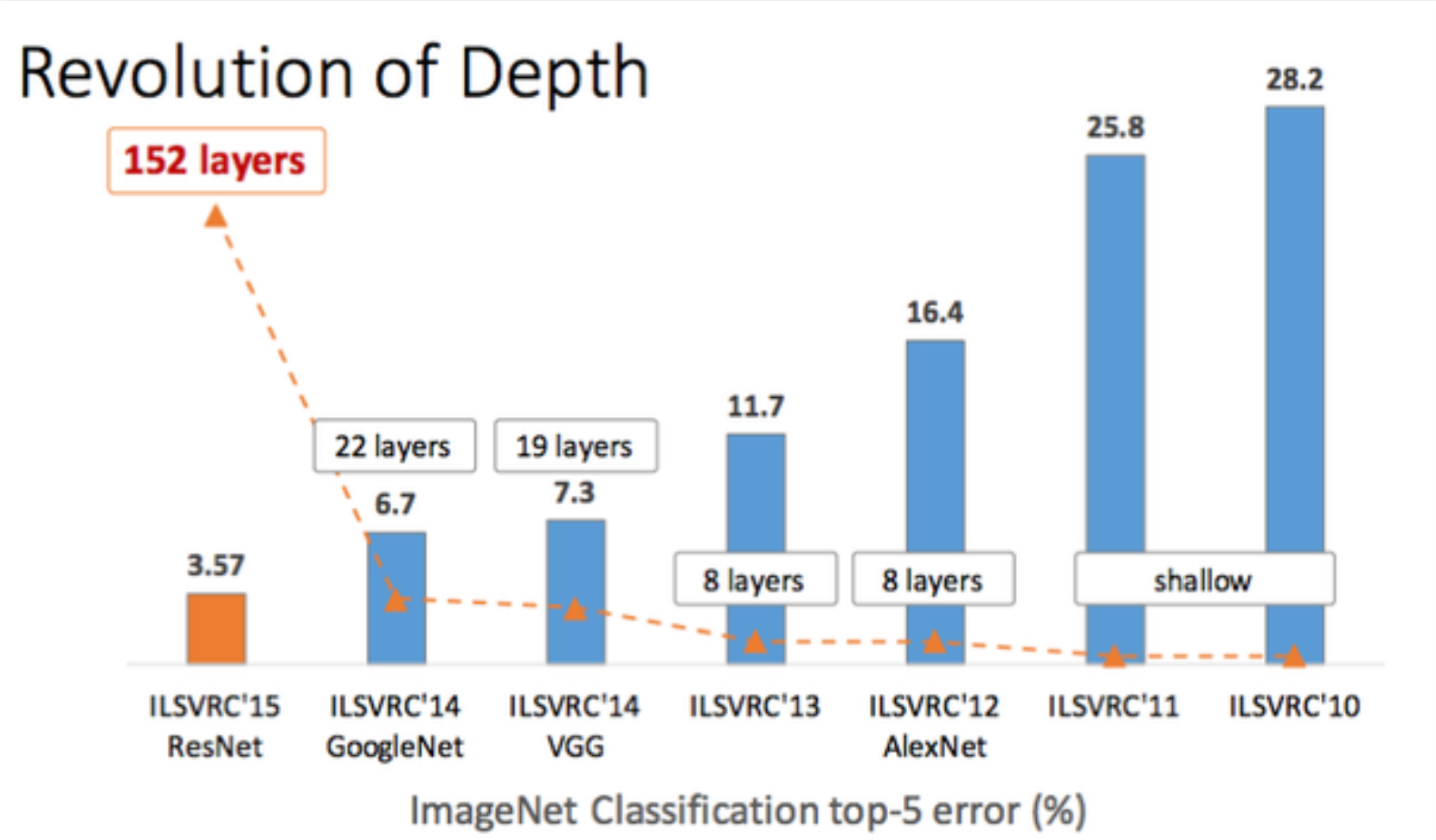
A **doubly-block Toeplitz matrix** is a block Toeplitz matrix where all blocks are also Toeplitz.



Doubly-block Toeplitz matrices

$\Rightarrow$ Doubly-block matrices are equivalent to the 2d convolution.

Performance of Convolutional Neural Networks



Introduction to Adversarial Attacks

Nightmare at test time: robust learning by feature deletion

A Globerson (2006)

Robustness is important in context with:

- Domains with nonstationary feature distributions
- Input sensor failures

What they proposed:

- An algorithm for avoiding single feature over-weighting by analyzing robustness using a game theoretic formalization.
- Classifiers which are optimally resilient to deletion of features in a minimax sense.

Evasion Attacks against Machine Learning at Test Time

B. Biggio (2013)

Quote: "In one pertinent, well-motivated attack scenario, an adversary may attempt to evade a deployed system at test time by carefully manipulating attack samples"

Adversarial Examples is prevalent in context such that:

- Spam filtering
- Malware detection
- Network intrusion detection

Evasion Attacks against Machine Learning at Test Time

B. Biggio (2013)

Quote: "In one pertinent, well-motivated attack scenario, an adversary may attempt to evade a deployed system at test time by carefully manipulating attack samples"

Adversarial Examples is prevalent in context such that:

- Spam filtering
- Malware detection
- Network intrusion detection

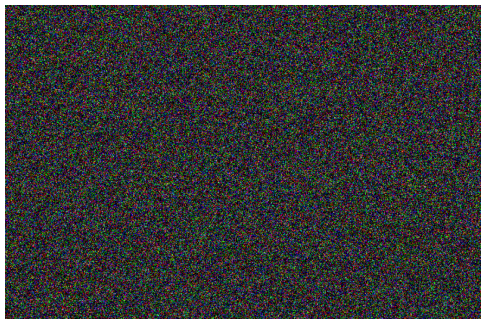
What they proposed:

- Simulate attack scenarios that exhibit different risk levels for the classifier by increasing the attacker's knowledge of the system
- Gives the classifier designer a better picture of the classifier performance under evasion attacks
- Allows him to perform a more informed model selection (or parameter setting)

How to properly defined Adversarial attacks?

How to properly defined Adversarial attacks?

Gaussian Noise sigma = 0.5



Adversarial Image

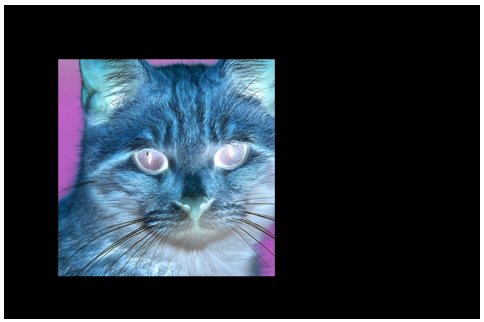
How to properly defined Adversarial attacks?

Uniform Noise sigma = 0.5

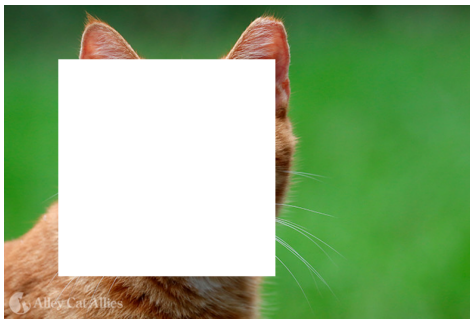
Adversarial Image

How to properly defined Adversarial attacks?

Inverse Perturbation

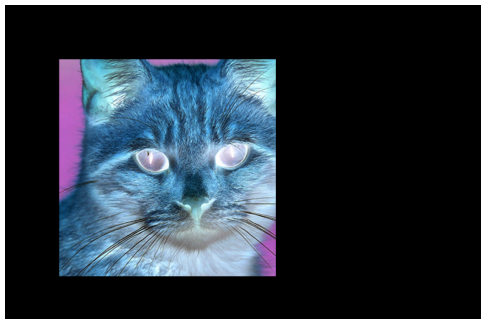


Adversarial Image

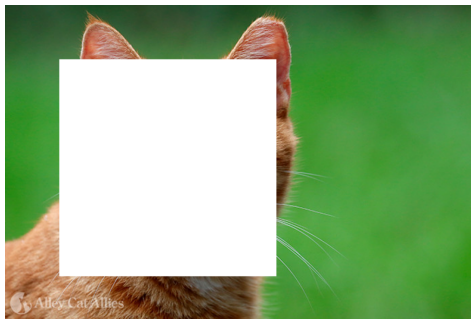


How to properly defined Adversarial attacks?

Inverse Perturbation



Adversarial Image



Adversarial attacks needs to be defined based on “amount” of pertubation

Distance in High Dimensions

ℓ_p Norm

A norm is a mathematical concept that measures the size or length of a vector in a vector space. It quantifies the distance of a vector from the origin.

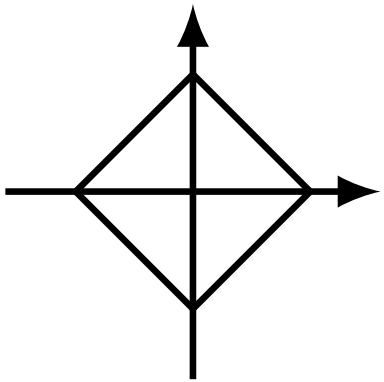
For a vector $x \in \mathbb{R}^n$, the ℓ_p norm is defined by

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}$$

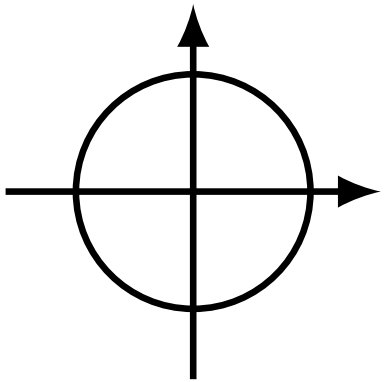
ℓ_1 norm	ℓ_2 norm	ℓ_∞ norm
Manhattan norm	Euclidean norm	Infinity norm
$\|x\|_1 = \|x_1\| + \|x_2\|$	$\|x\|_2 = \sqrt{x_1^2 + x_2^2}$	$\|x\|_\infty = \max(\|x_1\|, \|x_2\|)$

- Each norm measures the length of the vector x differently.
- The choice of norm depends on the specific problem or application.

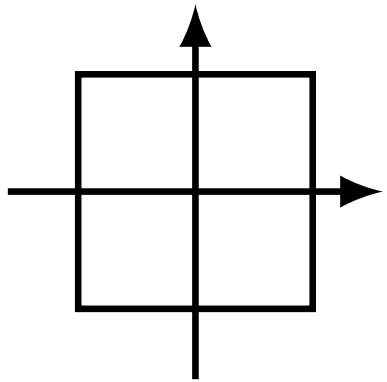
Distance in High Dimensions



ℓ_1 norm

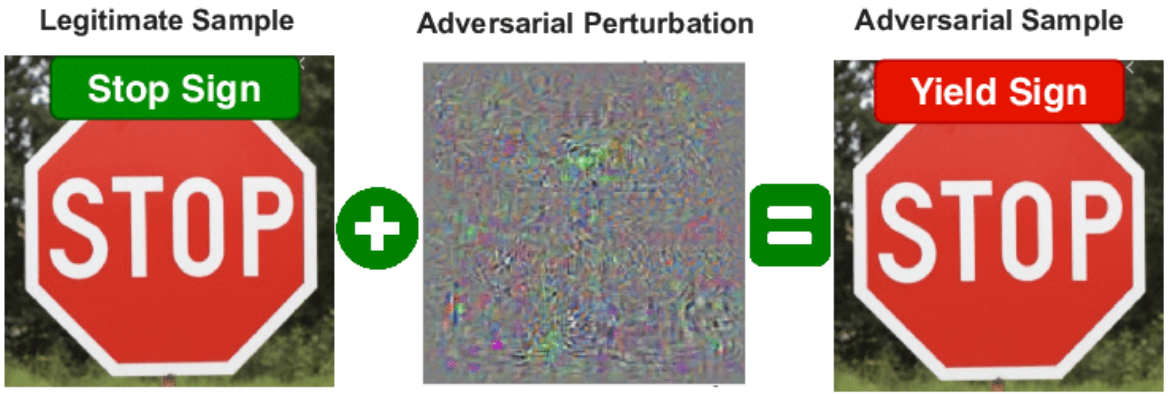


ℓ_2 norm



ℓ_∞ norm

How to defined Adversarial Attacks – ℓ_p norm definition

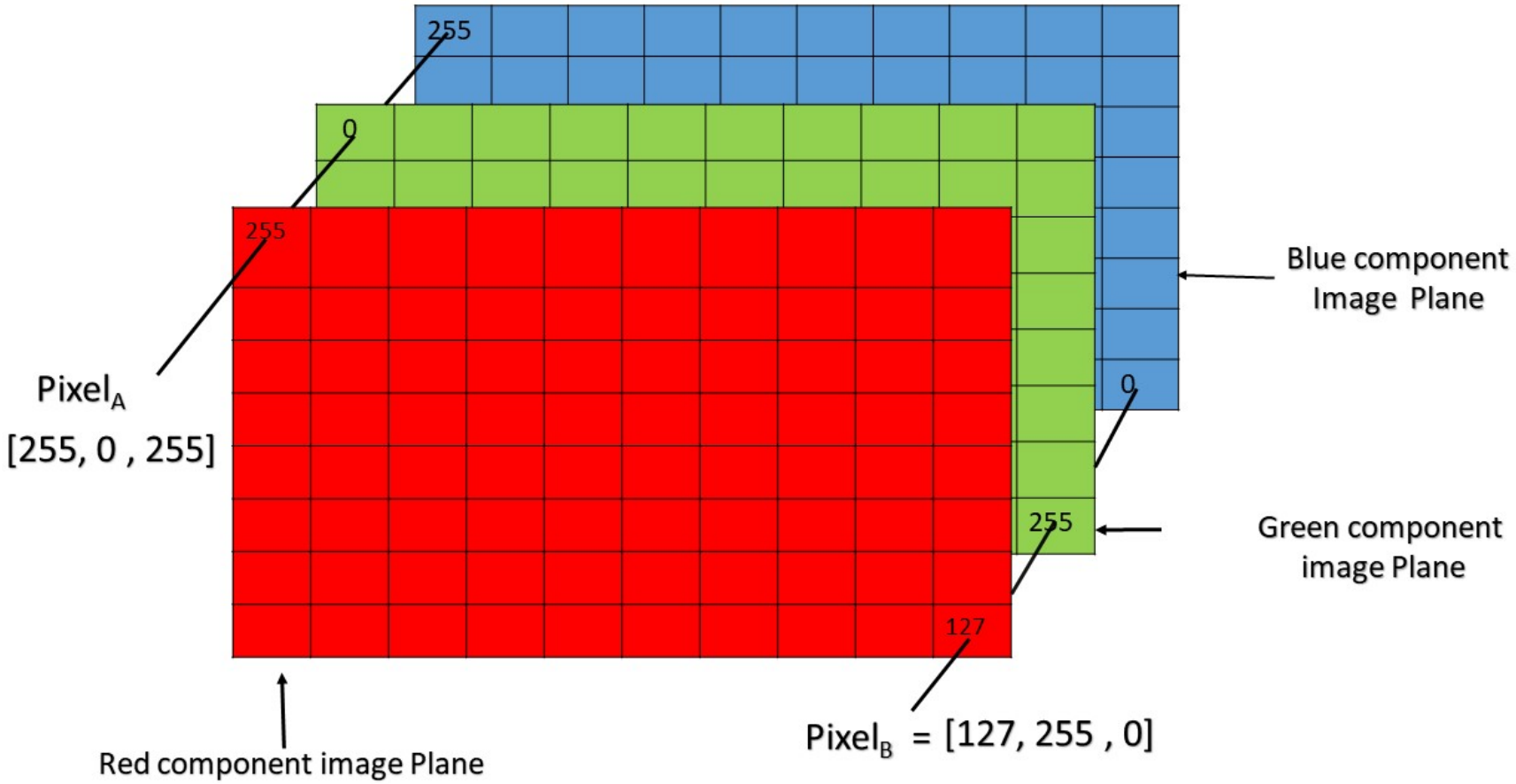


Definition (Adversarial Attacks)

Let $\mathbf{x} \in \mathcal{X}$, $y \in \mathcal{Y}$ the label of $\mathbf{x}$ and let $\mathbf{f}$ be a classifier. An adversarial attack of budget ε is a perturbation τ such that $\|\tau\|_2 \leq \varepsilon$ such that:

$$\mathbf{f}(\mathbf{x} + \tau) \neq y$$

Representation of RGB Images



Pixel of an RGB image are formed from the corresponding pixel of the three component images

Adversarial Attacks as a Combinatorial Problem

The RGB space of images is a bounded and discrete space.

Computing the optimal Adversarial Attack

We could find the optimal adversarial attack with respect to an image by iterating over all the values of all the pixel and each time checking if $f(x + \tau) = y$.

$\longrightarrow$ This algorithm is exponential with respect to the dimension of the image.

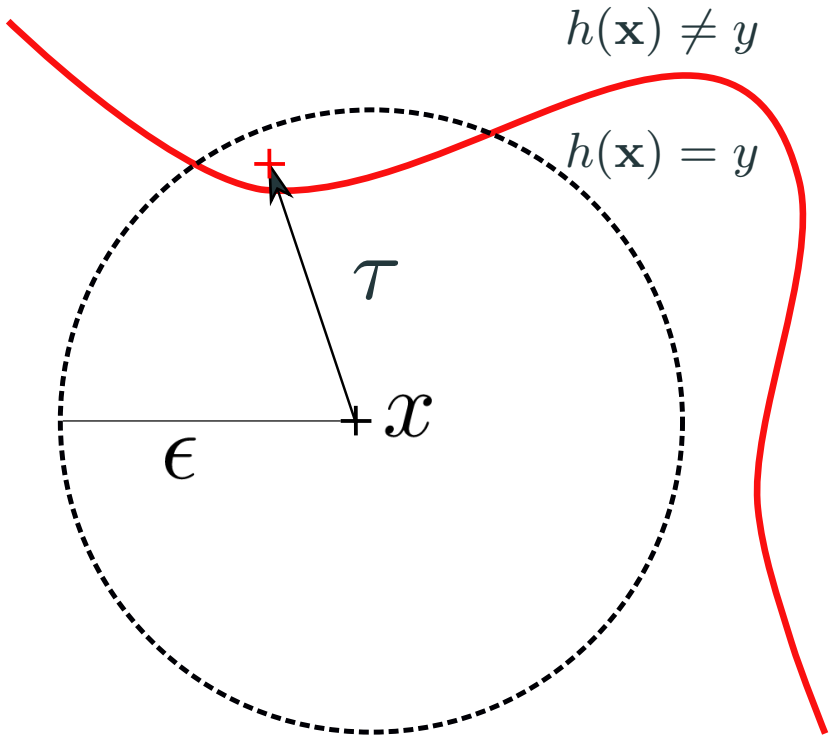
Adversarial Attacks as a Gradient-Based Optmization Problem

An **Adversarial Attack** aims at finding a **perturbation** τ such that:

- the network misclassifies
- $\|\tau\|_p \leq \epsilon$ where ϵ is small

Finding the perturbation τ can be formalized by:

$$\tau^{\text{adv}}(\mathbf{x}, y; f) \triangleq \underset{\|\tau\|_p \leq \epsilon}{\arg\max}\, L(f(\mathbf{x}+\tau), y)$$



Fast Gradient Signed Method

Explaining and Harnessing Adversarial Examples

I Goodfellow (2014)

Recall the objective:

$$\max_{\|\tau\|_\infty \leq \epsilon} L(f(x+\tau), y)$$

Fast Gradient Signed Method

Explaining and Harnessing Adversarial Examples

I Goodfellow (2014)

Recall the objective:

$$\max_{\|\tau\|_\infty \leq \epsilon} L(f(x+\tau), y)$$

Let us do linear expansion at x:

$$L(f(x+\tau), y) \approx \underbrace{L(f(x), y)}_{\text{constant}} + \langle \tau, \nabla_x L(f(x), y) \rangle$$

Fast Gradient Signed Method

Explaining and Harnessing Adversarial Examples

I Goodfellow (2014)

Recall the objective:

$$\max_{\|\tau\|_\infty \leq \epsilon} L(f(x+\tau), y)$$

Let us do linear expansion at x:

$$L(f(x+\tau), y) \approx \underbrace{L(f(x), y)}_{\text{constant}} + \langle \tau, \nabla_x L(f(x), y) \rangle$$

So the problem then reduces to:

$$\max_{\|\tau\|_\infty \leq \epsilon} \langle \tau, \nabla_x L(f(x), y) \rangle$$

Closed-Form Solution:

$$\tau^* = \epsilon \cdot \text{sign}\left(\nabla_x L(f(x), y)\right)$$

Fast Gradient Signed Method

Explaining and Harnessing Adversarial Examples

I Goodfellow (2014)

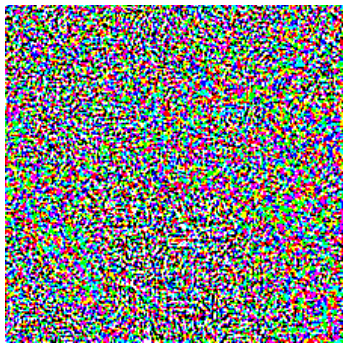
x

"panda"

57.7%

confidence

$+ .007 \times$



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

"nematode"

8.2% confidence

$=$

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

"gibbon"

99.3 %

confidence

Fast Gradient Signed Method

To compute an attack efficently, the attacker requires the weights of the model in order to compute the gradient.

Fast Gradient Signed Method

To compute an attack efficently, the attacker requires the weights of the model in order to compute the gradient.

Model Proection ?

Can one protect against avdersarial attacks by hidding the weights (*e.g.*, in private servers) ?

Fast Gradient Signed Method

To compute an attack efficently, the attacker requires the weights of the model in order to compute the gradient.

Model Proection ?

Can one protect against avdersarial attacks by hidding the weights (*e.g.*, in private servers) ?

NO!

Fast Gradient Signed Method

To compute an attack efficently, the attacker requires the weights of the model in order to compute the gradient.

Model Proection ?

Can one protect against avdersarial attacks by hidding the weights (*e.g.*, in private servers) ?

NO!

1. Attacks are transferable between models
2. Black-box attacks, while less efficient, also exist

Attack Taxonomy

Adversarial attack can be further classified into:

1. **White-box attack**: Attackers have full knowledge about the ML model

   i.e., they have access to parameters, hyperparameters, gradients, architecture, etc.

2. **Black-box attack**: Attackers don't have access to the ML model parameters, gradients, architecture

   They may have some knowledge about the used ML algorithm
   e.g., attackers may know that a ResNet50 model is used for classification, but they don't have access to the model parameters
   Attackers may query the model to obtain knowledge (can get examples)

Attack Taxonomy

Evasion attack can be further classified into:

1. **Non-targeted attack**

   The goal is to mislead the classifier to predict any labels other than the ground truth label

   More difficult to defend against

2. **Targeted attack**

   The goal is to mislead the classifier to predict a target label for an image

   More difficult for the attacker

   e.g., perturb an image of a turtle, so that the model predicts it is a riffle

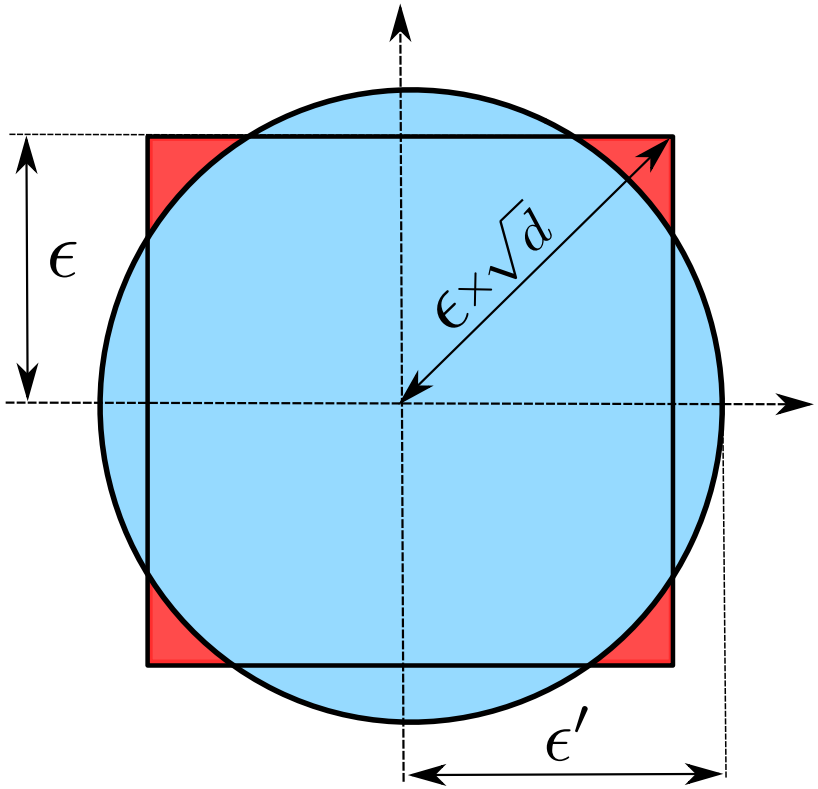
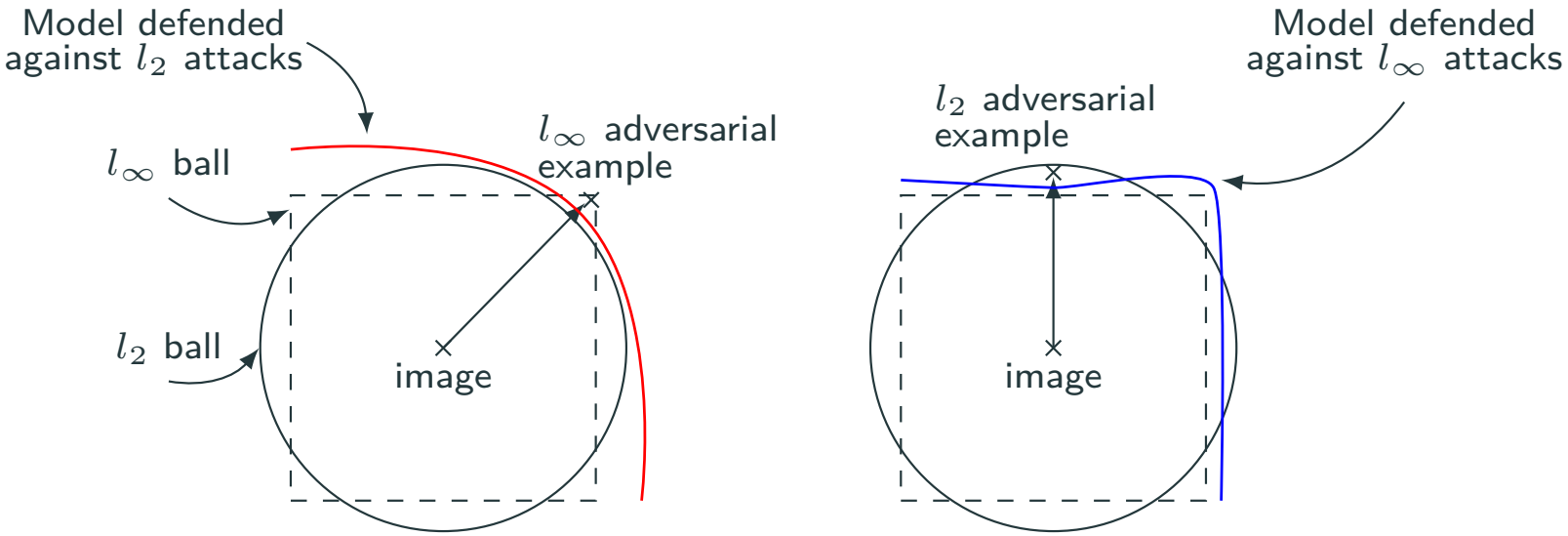


Figure 3: 2D representation of the ℓ_∞ and ℓ_2 balls of respective radius ϵ and ϵ'

Protecting Neural Networks against Several Types of Attacks



- Protecting against one type of attack does not protect against other types
- The volume of the intersection of the balls is negligible when d is large
- It is necessary to mix defense strategies
  - Advocating for multiple defense strategies against adversarial examples, A Araujo (2020)
  - Adversarial robustness against the union of multiple perturbation models, P Maini (2020)

Let us defined the volume of the ℓ_p ball of radis r in dimension d as $V(r, \ell_p, d)$.

We want to compare the proportion of an inscribed hypersphere with radius r and dimension d, to that of a hypercube with edges of length $2r$

Let us defined the volume of the ℓ_p ball of radis r in dimension d as $V(r, \ell_p, d)$.

We want to compare the proportion of an inscribed hypersphere with radius r and dimension d, to that of a hypercube with edges of length $2r$ In n dimension, the volume of the ℓ_2 and ℓ_∞ respectivly becomes:

$$V(r, \ell_2, d) = \frac{2r^d \pi^{\frac{d}{2}}}{d\,\Gamma\left(\frac{d}{2}\right)} \qquad V(2r, \ell_\infty, d) = (2r)^d$$

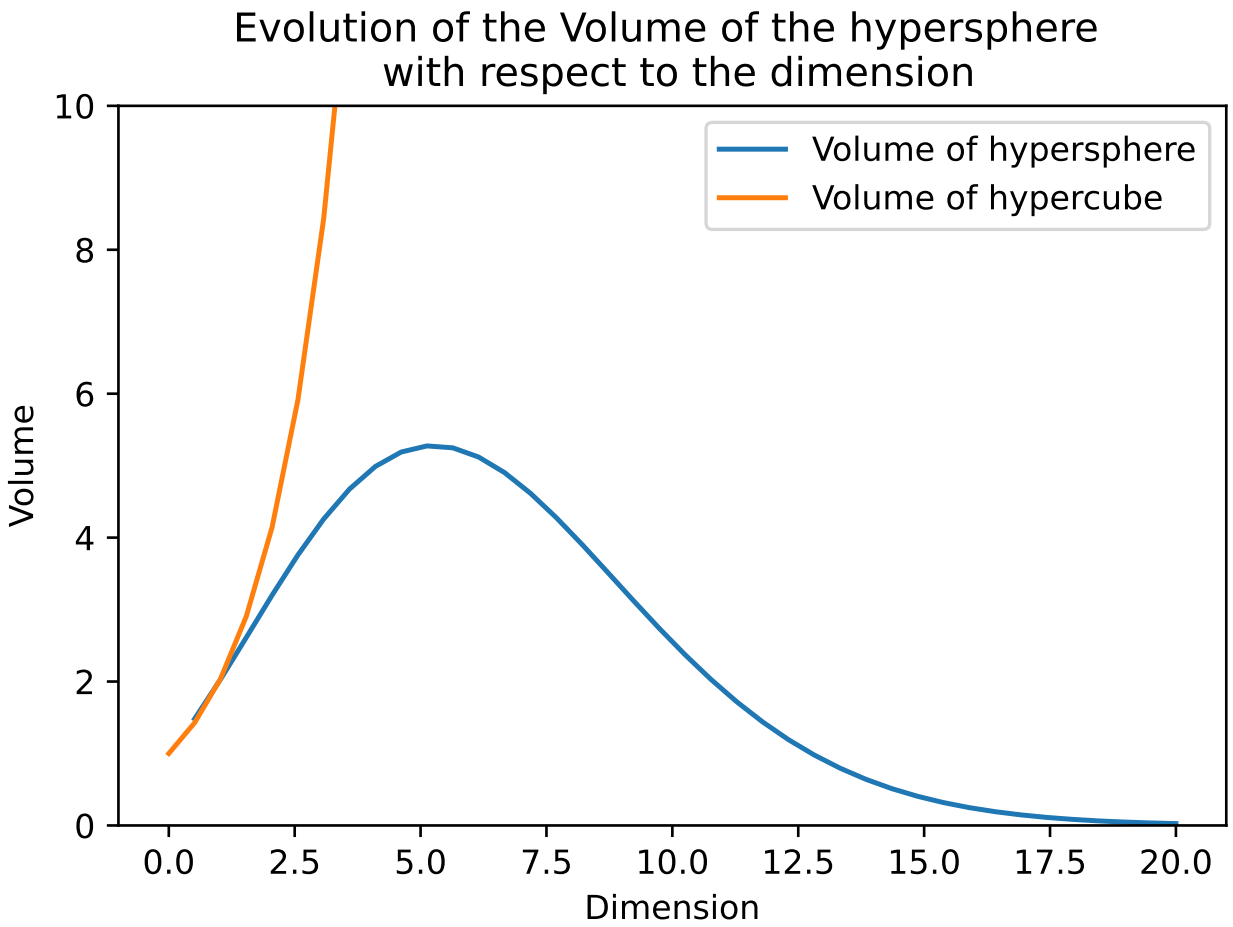
Let us defined the volume of the ℓ_p ball of radis r in dimension d as $V(r, \ell_p, d)$.

We want to compare the proportion of an inscribed hypersphere with radius r and dimension d, to that of a hypercube with edges of length $2r$ In n dimension, the volume of the ℓ_2 and ℓ_∞ respectivly becomes:

$$V(r, \ell_2, d) = \frac{2r^d \pi^{\frac{d}{2}}}{d\,\Gamma\left(\frac{d}{2}\right)} \qquad V(2r, \ell_\infty, d) = (2r)^d$$

We have:

$$\frac{V(r, \ell_2, d)}{V(2r, \ell_\infty, d)} = \frac{\pi^{\frac{d}{2}}}{d\ 2^{d-1}\ \Gamma\left(\frac{d}{2}\right)} \underset{d\to\infty}{\longrightarrow} 0$$


Evolution of the Volume of the hypersphere
with respect to the dimension
Volume of hypersphere
Volume of hypercube
Volume
Dimension
0
2
4
6
8
10
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

Intuition Behind Adversarial Attacks – Spectral Analysis

Intriguing properties of neural networks

C Szegedy (2023)

Adversarial attacks are related to the expressivity and sensitivity of the network.

Intuition Behind Adversarial Attacks – Spectral Analysis

Intriguing properties of neural networks
C Szegedy (2023)

Adversarial attacks are related to the expressivity and sensitivity of the network.

Most neural networks are Lipschitz continuous functions (however, not Transformers)

Intriguing properties of neural networks

C Szegedy (2023)

Adversarial attacks are related to the expressivity and sensitivity of the network.

Most neural networks are Lipschitz continuous functions (however, not Transformers)

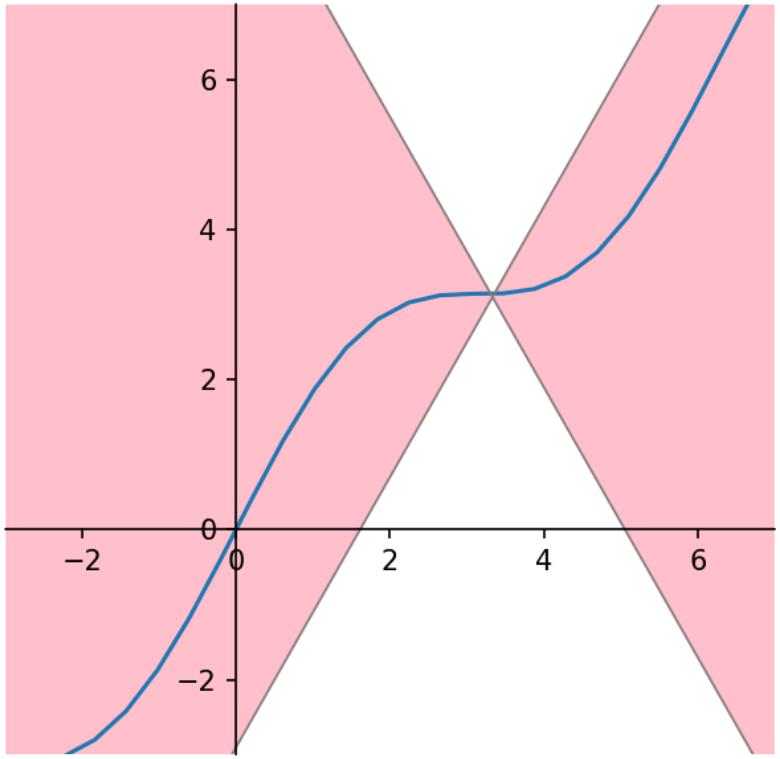
Let f be a neural network, we call f Lipschitz if for $x_1, x_2 \in \mathcal{X}$ and $K > 0$, the following holds:

$$\|f(\mathbf{x}_1) - f(\mathbf{x}_2)\|_p \leq K\|\mathbf{x}_1 - \mathbf{x}_2\|_p$$

The smallest constant K for which this inequality holds is called the Lipschitz constant of f.

Lipschitz Constant of Neural Networks

For a Lipschitz continuous function, there exists a double cone (white) whose origin can be moved along the graph so that the whole graph always stays outside the double cone.



Lipschitz Constant of Neural Networks

Gradient-based definition of the Lipschitz Constant. For a function f that is Lipschitz continuous concerning the ℓ_p-norm, its Lipschitz constant is represented as:

$$\mathrm{Lip}(f) = \sup_{x \in \mathcal{X}} \|\nabla f(x)\|_p \ .$$

Properties of Lipschitz Constant for several functions

The relationships between the Lipschitz constants and some functionals which frequently appears in deep neural networks: composition, addition, and concatenation.
Let f and g be functions with Lipschitz constants bounded by L_1 and L_2, respectively. The Lipschitz constant of output for each functional is bounded as follows:

$$f \circ g : L_1, \cdot L_2, \qquad f + g : L_1 + L_2, \qquad (f, g) : \sqrt{L_1^2 + L_2^2}.$$

Recall the neural network:

Neural Network

A neural network of p layers is defined as follows:

$$f(\mathbf{x}) = \phi^{(p)} \circ \rho \circ \phi^{(p-1)} \circ \rho \circ \cdots \circ \rho \circ \phi^{(1)}(\mathbf{x})$$

where $\phi_{\mathbf{W},\mathbf{b}} \triangleq \mathbf{x} \mapsto \mathbf{W}\mathbf{x} + \mathbf{b}$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^n$, $\mathbf{W} \in \mathbb{R}^{n \times n}$, ρ some non linear functions and Ω corresponds to the set of all parameters.

Recall the neural network:

Neural Network

A neural network of p layers is defined as follows:

$$f(\mathbf{x}) = \phi^{(p)} \circ \rho \circ \phi^{(p-1)} \circ \rho \circ \cdots \circ \rho \circ \phi^{(1)}(\mathbf{x})$$

where $\phi_{\mathbf{W},\mathbf{b}} \triangleq \mathbf{x} \mapsto \mathbf{W}\mathbf{x} + \mathbf{b}$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^n$, $\mathbf{W} \in \mathbb{R}^{n \times n}$, ρ some non linear functions and Ω corresponds to the set of all parameters.

The network is a composition of linear function $\phi_{\mathbf{W},\mathbf{b}} \triangleq \mathbf{x} \mapsto \mathbf{W}\mathbf{x} + \mathbf{b}$ and non linear functions ρ. Most non linear functions used in Deep Learning are 1-Lipschitz.

Recall the neural network:

Neural Network

A neural network of p layers is defined as follows:

$$f(\mathbf{x}) = \phi^{(p)} \circ \rho \circ \phi^{(p-1)} \circ \rho \circ \cdots \circ \rho \circ \phi^{(1)}(\mathbf{x})$$

where $\phi_{\mathbf{W},\mathbf{b}} \triangleq \mathbf{x} \mapsto \mathbf{W}\mathbf{x} + \mathbf{b}$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^n$, $\mathbf{W} \in \mathbb{R}^{n \times n}$, ρ some non linear functions and Ω corresponds to the set of all parameters.

The network is a composition of linear function $\phi_{\mathbf{W},\mathbf{b}} \triangleq \mathbf{x} \mapsto \mathbf{W}\mathbf{x} + \mathbf{b}$ and non linear functions ρ. Most non linear functions used in Deep Learning are 1-Lipschitz.

Can we evaluate (or bound) the Lipschitz of a neural network ?

Intuition Behind Adversarial Attacks – Spectral Analysis

Let f be a neural network, the Lipschitz can be bounded as follows:

$$\mathrm{Lip}(f) \leq \prod_{i=1}^{p} \mathrm{Lip}\left(\phi^{(i)}\right) = \prod_{i=1}^{p} \left\|\mathbf{W}^{(i)}\right\|_p$$

Intuition Behind Adversarial Attacks – Spectral Analysis

Let f be a neural network, the Lipschitz can be bounded as follows:

$$\mathrm{Lip}(f) \leq \prod_{i=1}^{p} \mathrm{Lip}\left(\phi^{(i)}\right) = \prod_{i=1}^{p} \left\|\mathbf{W}^{(i)}\right\|_p$$

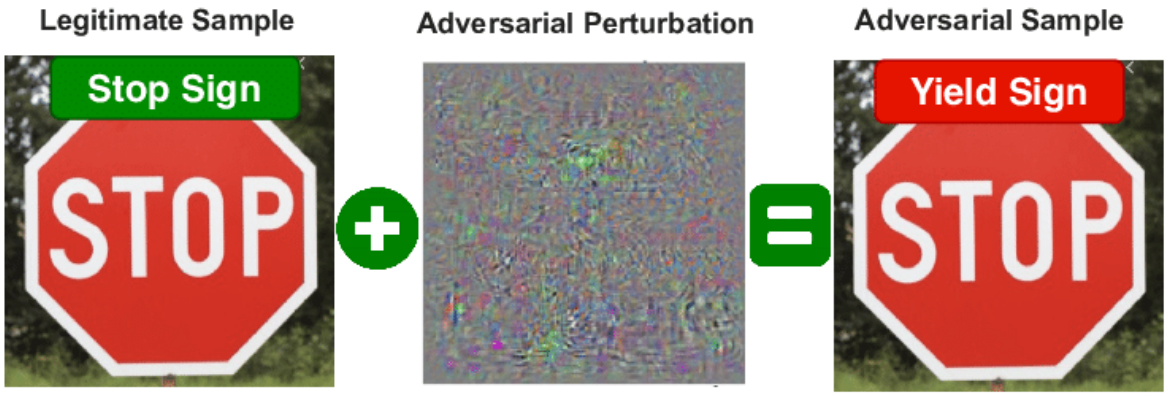
Finally, we can bound the difference in output by:

$$\|f(x) - f(x+\tau)\| \leq \mathrm{Lip}(f)\epsilon$$

where $\|\tau\| \leq \epsilon$.

How to create Adversarial Attacks

How to defined Adversarial Attacks – ℓ_p norm definition



Definition (Adversarial Attacks)

Let $\mathbf{x} \in \mathcal{X}$, $y \in \mathcal{Y}$ the label of $\mathbf{x}$ and let $\mathbf{f}$ be a classifier. An adversarial attack of budget ε is a perturbation τ such that $\|\tau\|_p \leq \varepsilon$ such that:

$$\mathbf{f}(\mathbf{x} + \tau) \neq y$$

Distance in High Dimensions

ℓ_p Norm

A norm is a mathematical concept that measures the size or length of a vector in a vector space. It quantifies the distance of a vector from the origin.

For a vector $x \in \mathbb{R}^n$, the ℓ_p norm is defined by

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}$$

ℓ_1 norm	ℓ_2 norm	ℓ_∞ norm
Manhattan norm	Euclidean norm	Infinity norm
$\|x\|_1 = \|x_1\| + \|x_2\|$	$\|x\|_2 = \sqrt{x_1^2 + x_2^2}$	$\|x\|_\infty = \max(\|x_1\|, \|x_2\|)$

- Each norm measures the length of the vector x differently.
- The choice of norm depends on the specific problem or application.

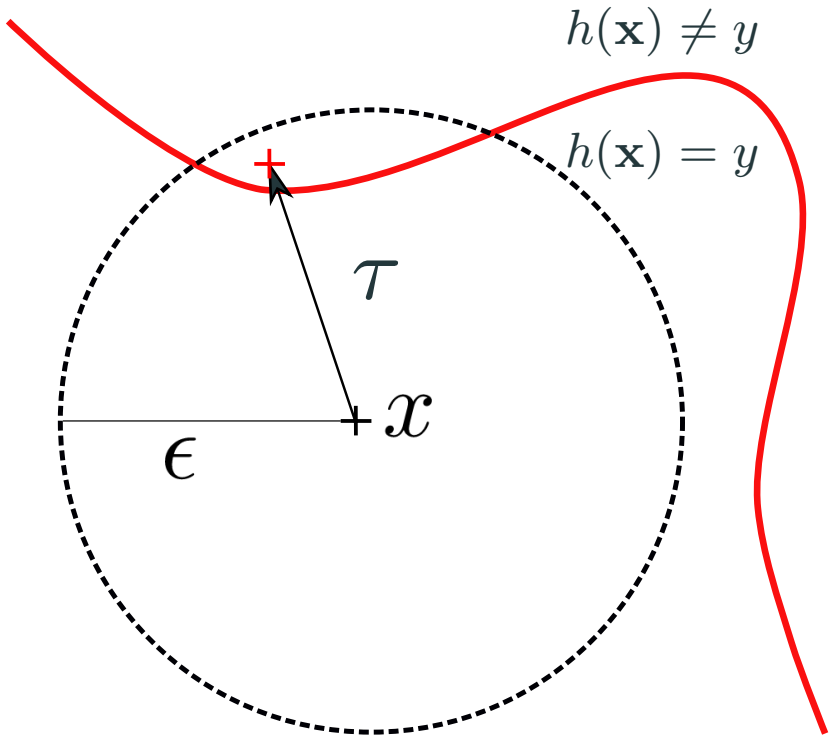
Adversarial Attacks as a Gradient-Based Optmization Problem

An **Adversarial Attack** aims at finding a **perturbation** τ such that:

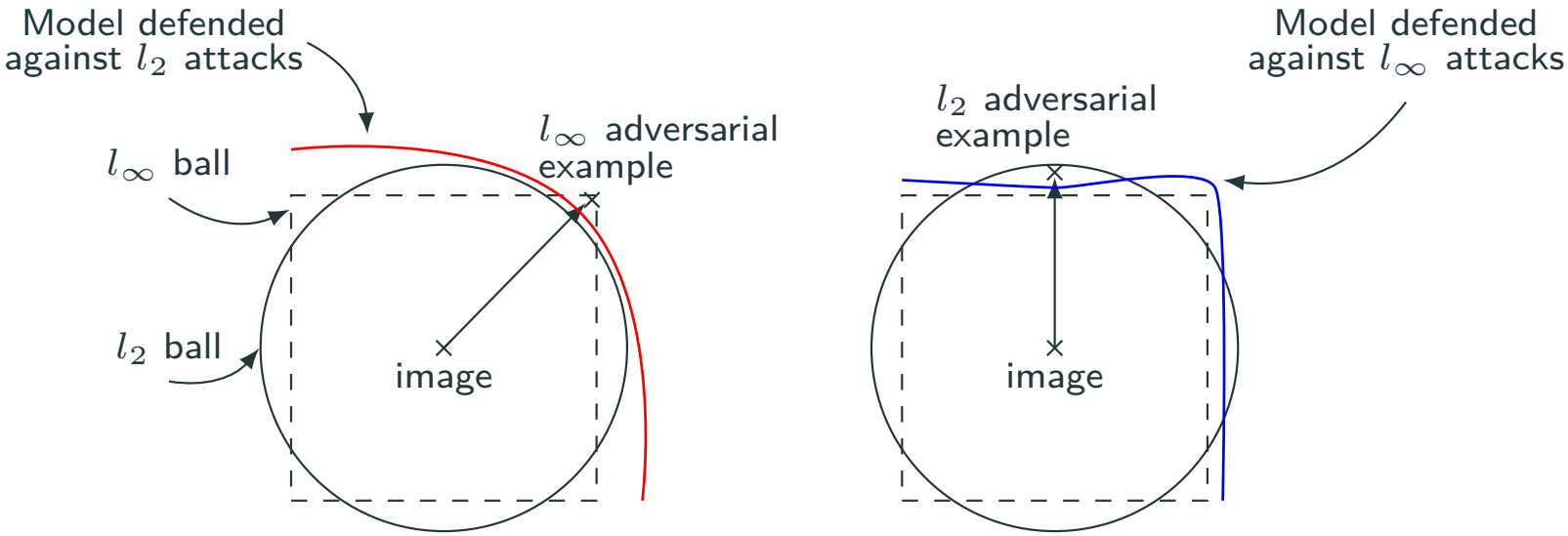
- the network misclassifies
- $\|\tau\|_p \leq \epsilon$ where ϵ is small

Finding the perturbation τ can be formalized by:

$$\tau^{\text{adv}}(\mathbf{x}, y; f) \triangleq \underset{\|\tau\|_p \leq \epsilon}{\arg\max}\, L(f(\mathbf{x}+\tau), y)$$



Protecting Neural Networks against Several Types of Attacks



- Protecting against one type of attack does not protect against other types
- The volume of the intersection of the balls is negligible when d is large
- It is necessary to mix defense strategies
  - Advocating for multiple defense strategies against adversarial examples, A Araujo (2020)
  - Adversarial robustness against the union of multiple perturbation models, P Maini (2020)

Recall FGSM Attack

Explaining and Harnessing Adversarial Examples

I Goodfellow (2014)

Recall the objective:

$$\max_{\|\tau\|_\infty \leq \epsilon} L(f(x+\tau), y)$$

Recall FGSM Attack

Explaining and Harnessing Adversarial Examples

I Goodfellow (2014)

Recall the objective:

$$\max_{\|\tau\|_\infty \leq \epsilon} L(f(x+\tau), y)$$

Let us do linear expansion at x:

$$L(f(x+\tau), y) \approx \underbrace{L(f(x), y)}_{\text{constant}} + \langle \tau, \nabla_x L(f(x), y) \rangle$$

Recall FGSM Attack

Explaining and Harnessing Adversarial Examples

I Goodfellow (2014)

Recall the objective:

$$\max_{\|\tau\|_\infty \leq \epsilon} L(f(x+\tau), y)$$

Let us do linear expansion at x:

$$L(f(x+\tau), y) \approx \underbrace{L(f(x), y)}_{\text{constant}} + \langle \tau, \nabla_x L(f(x), y) \rangle$$

So the problem then reduces to:

$$\max_{\|\tau\|_\infty \leq \epsilon} \langle \tau, \nabla_x L(f(x), y) \rangle$$

Closed-Form Solution:

$$\tau^* = \epsilon \cdot \text{sign}\left(\nabla_x L(f(x), y)\right)$$

BIM Attack

Basic iterative method (BIM) attack

Adversarial examples in the physical world
A Kurakin (2017)

- BIM is a variant of FGSM: it repeatedly adds noise to the image x in multiple iterations, in order to cause misclassification
- The number of iterations steps is t, and α is the amount of noise that is added at each step

$$x_{\text{adv}}^{(t)} = x^{(t-1)} + \alpha \text{sign}\left(\nabla_x L(f(x^{(t-1)}), y)\right)$$

- The perturbed image after the t iterations is x_{adv}^{t}
- Multiple steps of adding noise increase the chances of misclassifying the image

PGD Attack

Projected gradient descent (PGD) attack

Towards Deep Learning Models Resistant to Adversarial Attacks
A Madry (2018)

- PGD is an extension of BIM (and FGSM), where after each step of perturbation, the adversarial example is projected back onto the ℓ_p-ball of size ε using a projection function Π

$$x_{\text{adv}}^{(t)} = \Pi_{B_p(x,\varepsilon)} \left(x^{(t-1)} + \alpha \nabla_x L(f(x^{(t-1)}), y) \right)$$

  where $B_p(x, \varepsilon) = \{x + \tau \text{ s.t } \|\tau\| \leq \varepsilon\}$

- Different from BIM, PGD uses random initialization for x, by adding random noise from a uniform distribution with values in the range $[-\varepsilon, \varepsilon]$
- PGD is regarded as the strongest first-order attack

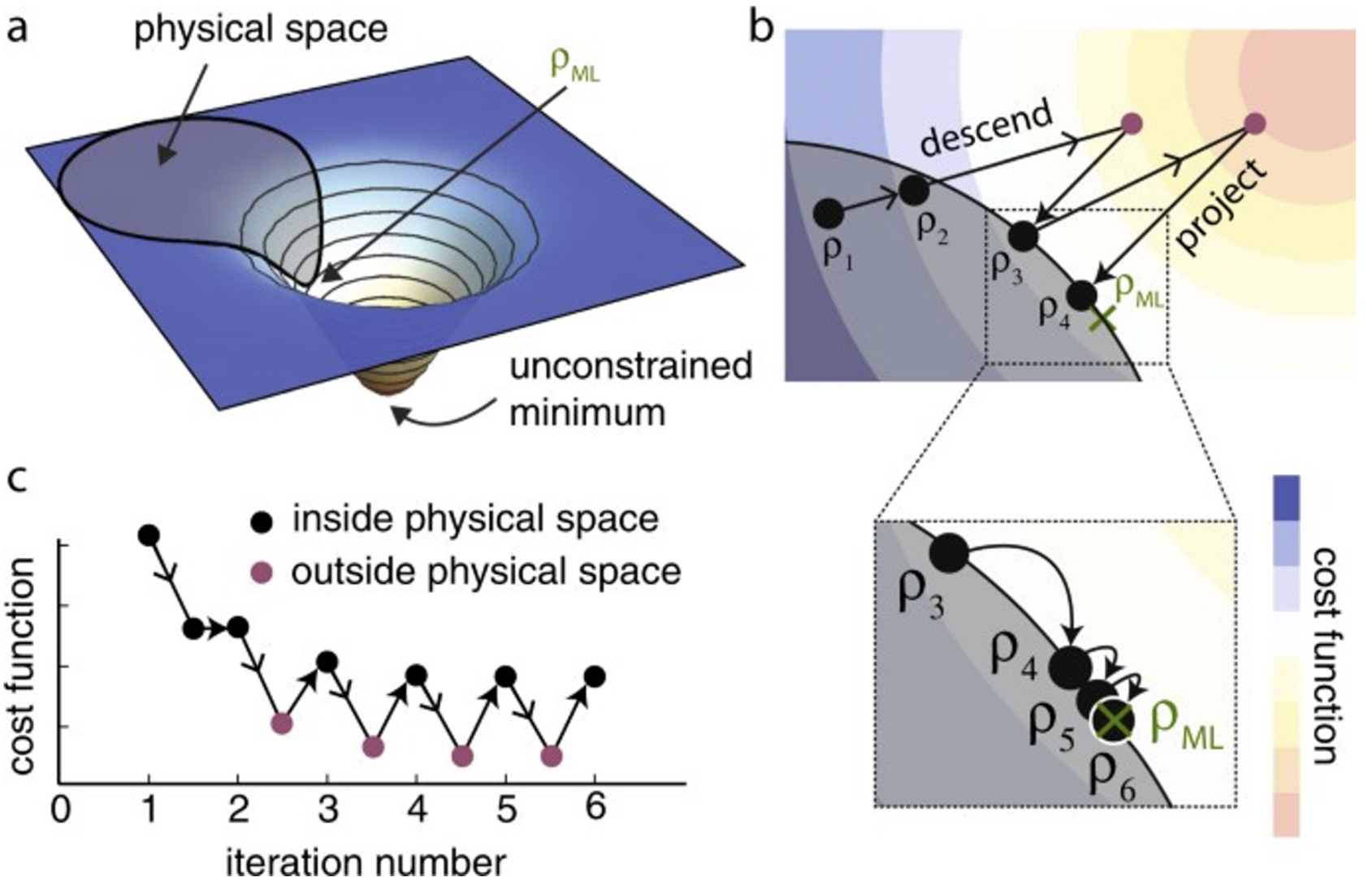
PGD Attack

- Gradient approaches can also be designed as targeted white-box attacks
- The added perturbation noise aims to minimize the loss function of the image for a specific class label
- For a targeted attack, if the target class label is denoted y', adversarial examples are created by using

$$x_{\text{adv}}^{(t)} = \Pi_{B_p(x,\varepsilon)} \left(x^{(t-1)} - \alpha \nabla_x L(f(x^{(t-1)}), y') \right)$$

- It is based on minimizing the loss function with respect to the target class y'
- This is opposite to non-targeted attacks, which maximize the loss function with respect to the true class label

PGD Attack


a
physical space
ρ_{ML}
unconstrained
minimum
b
descend
project
ρ_1
ρ_2
ρ_3
ρ_4
ρ_{ML}
ρ_3
ρ_4
ρ_5
ρ_6
ρ_{ML}
cost function
c
inside physical space
outside physical space
cost function
0
1
2
3
4
5
6
iteration number

CW Attack

Carlini-Wagner (CW) attack Towards Evaluating the Robustness of Neural Networks

N Carlini (2017)

Advantages of CW attack:

- Achieve low amount of perturbation to fool a classifier
- Resist to defense algorithms
- The generated adversarial images are transferrable across DL models
- The adversarial example is hard to be detected

CW Attack

Carlini-Wagner (CW) attack Towards Evaluating the Robustness of Neural Networks

N Carlini (2017)

Three elements in the design of CW attack:

- Create an adversarial image, so that the distance is minimal
- Targeted: The classifier should output a class label t, where t is different from the true label
- Each entry of $x + \tau$ should be valid in $[0, 1]$

$$\begin{aligned} &\text{minimize } \|\tau\|_p \\ &\text{such that } f(x+\tau) = t \\ &\qquad \text{and } x + \tau \in [0, 1] \end{aligned}$$

CW Attack

Carlini-Wagner (CW) attack Towards Evaluating the Robustness of Neural Networks

N Carlini (2017)

This initial formulation of the optimization problem is difficult to solve

- Because the constraint $f(x+\tau) = t$ is non-differentiable
- Instead, we can replace the constraint with another one:

$$\begin{aligned} &\text{minimize } \|\tau\|_p \\ &\text{such that } F(x+\tau) \leq 0 \\ &\text{and } x+\tau \in [0,1] \end{aligned}$$

where the function F is choosen such that when $f(x+\tau) = t$ we have $F(x+\tau) \leq 0$ and F should be differentiable.

CW Attack

Carlini-Wagner (CW) attack Towards Evaluating the Robustness of Neural Networks

N Carlini (2017)

They considered 7 variants for the choice of function F

Let us define $\text{ce}(x, y)$ as the crossentropy loss, z the logits, S the softmax, $\text{softplus}(x) = \log(1 + e^x)$

- $F_1(x') = -\text{loss}_{F,t}(x') + 1$
- $F_2(x') = \text{ReLU}\left(\max_{i \neq t} S(x')_i - S(x')_t\right)$
- $F_3(x') = \text{softplus}\left(\max_{i \neq t} S(x')_i - S(x')_t\right) - \log(2)$
- $F_4(x') = \text{ReLU}\left(0.5 - S(x')_t\right)$
- $F_5(x') = -\log(2S(x')_t - 2)$
- $F_6(x') = \text{ReLU}\left(\max_{i \neq t} z_i - z_t\right)$
- $F_7(x') = \text{softplus}\left(\max_{i \neq t} z_i - z_t\right) - \log(2)$

F_6 have been shown to be the best.

Square Attack **Square Attack: a query-efficient black-box adversarial attack via random search**

M Andriushchenko (2020)

Black-Box Attacks

Square Attack **Square Attack: a query-efficient black-box adversarial attack via random search**
M Andriushchenko (2020)

Black box attacks

- don't assume access to the model but assume access to the prediciton via an API
- query the model seevral times
- compute an approximation of the gradient or make random moves

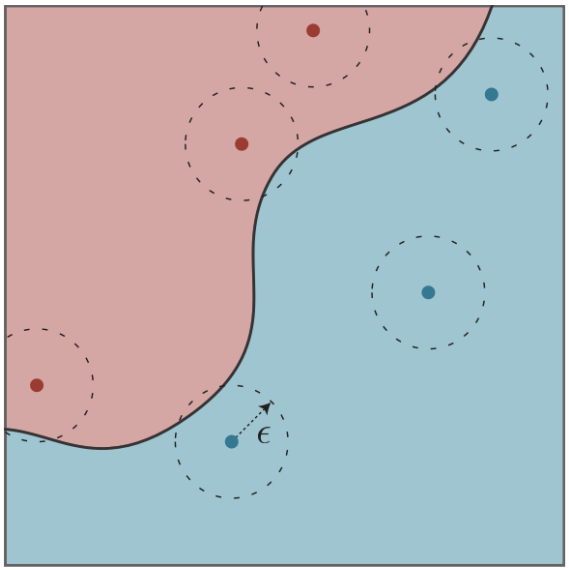
AutoAttack

AutoAttack: Ensemble of Diverse Methods **Reliable Evaluation of Adversarial Robustness with an Ensemble of Diverse Parameter-free Attacks**
F Croce (2020)

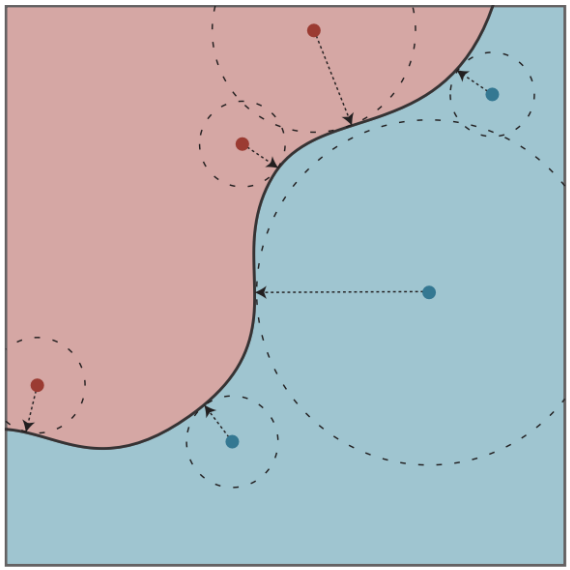
Current State-of-the art attack: Ensemble different attacks

Measuring robustness to adversarial perturbations

First approach: given $\varepsilon \geq 0$, one can define robustness as the proportion of input samples for which there exist no adversarial perturbation with $\|\tau\| \leq \varepsilon$.



Second approach alternatively, robustness can be measured as the average norm of "minimal adversarial perturbations".



Heuristic Defenses

Improving robustness

Deep networks are vulnerable to adversarial perturbations; even an imperceptible perturbation can cause misclassification.

So far we have seen how to evaluate robustness of deep networks to adversarial perturbations.

A naïve method to improve robustness?

Not only we want to classify a given input x correctly, but the classifier should also classify any point in the vicinity of x the same as x.

By vicinity we mean an ℓ_p ball around x.

A naïve method to improve robustness?

Not only we want to classify a given input x correctly, but the classifier should also classify any point in the vicinity of x the same as x.

By vicinity we mean an ℓ_p ball around x.

- Standard training loss

$$\frac{1}{m} \min_{\theta} \sum_{i=0}^{m} L(f(x_i), y_i)$$

A naïve method to improve robustness?

Not only we want to classify a given input x correctly, but the classifier should also classify any point in the vicinity of x the same as x.

By vicinity we mean an ℓ_p ball around x.

- Standard training loss

$$\frac{1}{m} \min_{\theta} \sum_{i=0}^{m} L(f(x_i), y_i)$$

- Robust training loss

$$\frac{1}{m} \min_{\theta} \sum_{i=0}^{m} \max_{\tau} L(f(x_i + \tau), y_i)$$

A naïve method to improve robustness?

Not only we want to classify a given input x correctly, but the classifier should also classify any point in the vicinity of x the same as x.

By vicinity we mean an ℓ_p ball around x.

- Standard training loss

$$\frac{1}{m} \min_{\theta} \sum_{i=0}^{m} L(f(x_i), y_i)$$

- Robust training loss

$$\frac{1}{m} \min_{\theta} \sum_{i=0}^{m} \max_{\tau} L(f(x_i + \tau), y_i)$$

We alternatively minimise, w.r.t. parameters, and maximize, w.r.t. perturbation.

Adversarial training Instead of training on the original data, train on the most difficult samples, a.k.a. adversarial examples.

Adversarial Training – Results

Standard training

	ResNet	W-ResNet
Clean	**92.7%**	**95.2%**
FGSM	**27.5%**	**32.7%**
PGD-10	**0.8%**	**3.5%**

FGSM training

	ResNet	W-ResNet
Clean	**87.4%**	**90.3%**
FGSM	**90.9%**	**95.1%**
PGD-10	**0.0%**	**0.0%**

PGD-10 training

	ResNet	W-ResNet
Clean	**79.4%**	**87.3%**
FGSM	**51.7%**	**56.1%**
PGD-10	**43.7%**	**45.8%**

Adversarial Training comes with important drawbacks:

- **Cost of training**
- **Protect against one norm**

Adversarial Training – Results

K-PGD adversarial training [Madry et al., 2017] is generally slow. For example, the 7-PGD training of a WideResNet [Zagoruyko and Komodakis, 2016] on CIFAR-10 in Madry et al. [2017] takes about four days on a Titan X GPU.

Adversarial Training – Drawbacks

Possible solutions:

Adversarial Training for Free!

A Shafahi (2017)

Adversarial Training – Drawbacks

Possible solutions:

Adversarial robustness against the union of multiple perturbation models
P Maini (2020)

Other Types of Defense

Distillation as a Defense to Adversarial Perturbations against Deep NN

Towards Deep Neural Network Architectures Robust to Adversarial Examples

Mitigating Adversarial Effects through Randomization

Adversarial Logit Pairing

Improving the Robustness of Deep Neural Networks via Stability Training.

Hardening against adversarial examples with the smooth gradient method

Distillation as a Defense to Adversarial Perturbations against Deep NN

N Papernot (2016)

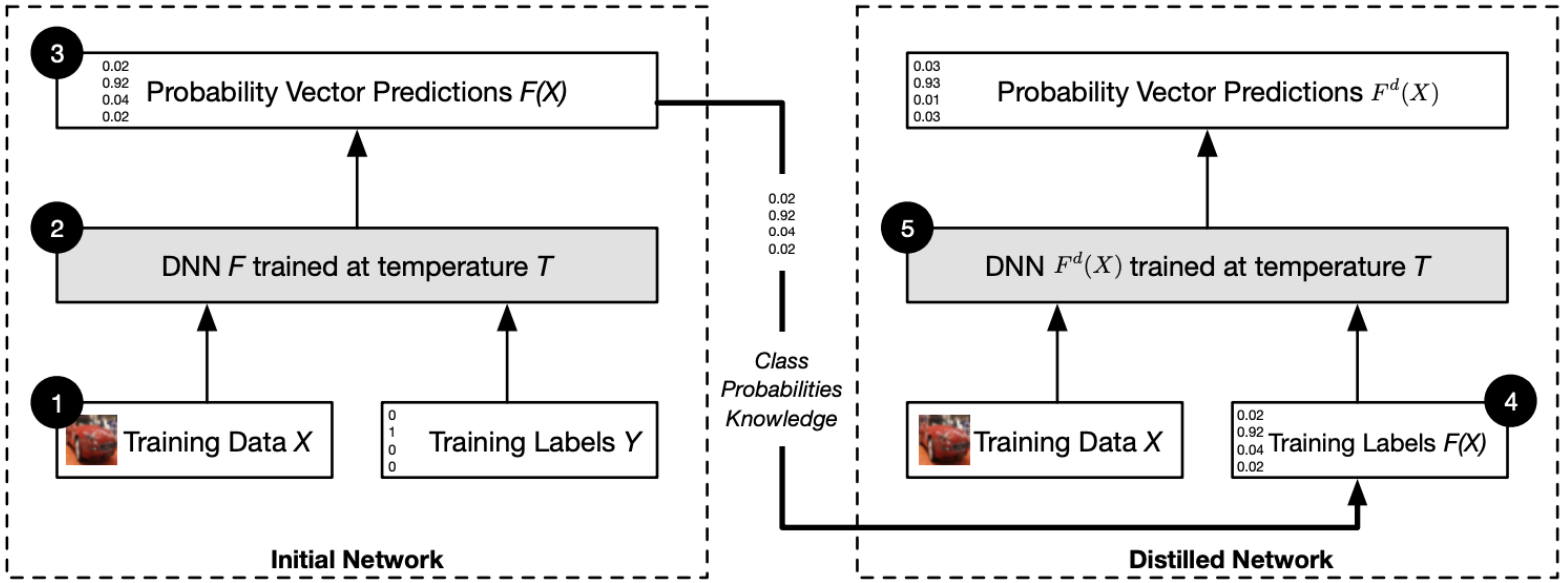
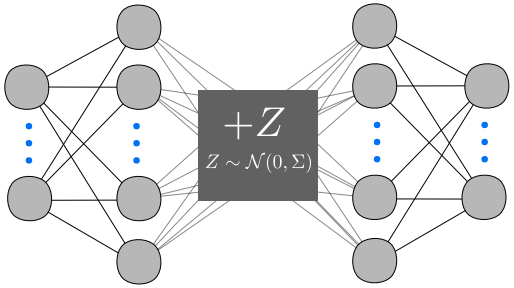


Fig. 5: **An overview of our defense mechanism based on a transfer of knowledge contained in probability vectors through distillation:** We first train an initial network F on data X with a softmax temperature of T. We then use the probability vector $F(X)$, which includes additional knowledge about classes compared to a class label, predicted by network F to train a distilled network F^d at temperature T on the same data X.

How does noise injection work?



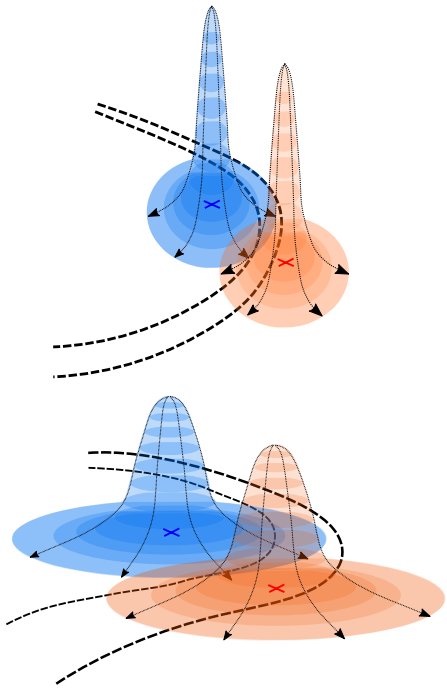
Formally: for a Feedforward network, we have

$$\tilde{F}(x) = \phi^{(N)}_{W_N, b_N} \circ \cdots \circ \tilde{\phi}^{(i)}_{W_i, b_i} \circ \cdots \circ \phi^{(1)}_{W_1, b_1}(x)$$

Where $\tilde{\phi}^{(i)}_{W_i, b_i}(z) = \sigma(W_i z + b_i) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \Sigma)$.

Several possible interpretations:

1) Robust optimization: Noise helps locally smoothing the network.
2) Data augmentation: Noise helps the network minimize the generalisation error.
3) Geometrical: Noise at pushes the decision boundary/makes it "probabilistic".
4) Game theory: there is no pure Nash equilibrium $\implies$ one needs a mixed strategy.



Theoretcial notion of risk:

$$R(f) \triangleq \mathbb{E}_{(x,y)\sim\mathcal{D}}\ \mathcal{L}\left(f(\mathbf{x}), y\right)$$

Defending against Adversarial Attacks

Theoretcial notion of risk:

$$R(f) \triangleq \mathbb{E}_{(x,y)\sim\mathcal{D}}\ \mathcal{L}\left(f(\mathbf{x}), y\right)$$

Adversarial Risk:

$$R^{\mathrm{adv}}(f) \triangleq \mathbb{E}_{(x,y)\sim\mathcal{D}}\ \mathcal{L}\left(f(\mathbf{x} + \tau^{\mathrm{adv}}(\mathbf{x}, y; f)), y\right)$$

Defending against Adversarial Attacks

One of the best empirical defense against adversarial attacks is to minimize the **Empirical Adversarial Risk** (Goodfellow et al. [3]):

$$\hat{R}_m^{\text{adv}}(f) \triangleq \frac{1}{m} \sum_{m=1}^{m} \mathcal{L}\left(f(\mathbf{x}_i + \tau^{\text{adv}}(\mathbf{x}_i, y_i; f)), y_i\right)$$

Defending against Adversarial Attacks

One of the best empirical defense against adversarial attacks is to minimize the **Empirical Adversarial Risk** (Goodfellow et al. [3]):

$$\hat{R}_m^{\text{adv}}(f) \triangleq \frac{1}{m} \sum_{m=1}^{m} \mathcal{L}\left(f(\mathbf{x}_i + \tau^{\text{adv}}(\mathbf{x}_i, y_i; f)), y_i\right)$$

We can bound the adversarial risk by the empirical adversarial risk and a complexity penalty:

$$R^{\text{adv}}(h) \quad \leq \quad \hat{R}_m^{\text{adv}}(f) \quad + \quad \underbrace{P^{\text{adv}}(f)}_{\substack{\text{Complexity} \\ \text{Penalty}}}$$

The **Adversarial Complexity Penalty** depends on the Lipschitz constant of the network (Farnia et al. [1]).

Defending against Adversarial Attacks

One of the best empirical defense against adversarial attacks is to minimize the **Empirical Adversarial Risk** (Goodfellow et al. [3]):

$$\hat{R}_m^{\text{adv}}(f) \triangleq \frac{1}{m} \sum_{m=1}^{m} \mathcal{L}\left(f(\mathbf{x}_i + \tau^{\text{adv}}(\mathbf{x}_i, y_i; f)), y_i\right)$$

We can bound the adversarial risk by the empirical adversarial risk and a complexity penalty:

$$R^{\text{adv}}(h) \quad \leq \quad \hat{R}_m^{\text{adv}}(f) \quad + \quad \underbrace{P^{\text{adv}}(f)}_{\substack{\text{Complexity} \\ \text{Penalty}}}$$

The **Adversarial Complexity Penalty** depends on the Lipschitz constant of the network (Farnia et al. [1]).

Reducing the Lipschitz constant of the Neural Network improves the robustness against adversarial attacks.

Combining Adversarial Training and Lipschitz regularization:

$$\arg\min_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} \underbrace{\ell\left(h(\mathbf{x}_i + \tau^{\mathrm{adv}}(\mathbf{x}_i, y_i; h)), y_i\right)}_{\text{Adversarial Training}} + \underbrace{\lambda \operatorname{Lip}(h)}_{\text{Lipschitz Regularization}}$$

Combining Adversarial Training and Lipschitz regularization:

NP-hard problem
Virmaux et al. [7]

$$\arg\min_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} \underbrace{\ell\left(h(\mathbf{x}_i + \tau^{\text{adv}}(\mathbf{x}_i, y_i; h)), y_i\right)}_{\text{Adversarial Training}} + \boxed{\underbrace{\lambda \operatorname{Lip}(h)}_{\text{Lipschitz Regularization}}}$$

Combining Adversarial Training and Lipschitz regularization:

NP-hard problem
Virmaux et al. [7]

$$\arg\min_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} \underbrace{\ell\left(h(\mathbf{x}_i + \tau^{\text{adv}}(\mathbf{x}_i, y_i; h)), y_i\right)}_{\text{Adversarial Training}} + \boxed{\underbrace{\lambda \operatorname{Lip}(h)}_{\text{Lipschitz Regularization}}}$$

Bounding $\operatorname{Lip}(h)$

Combining Adversarial Training and Lipschitz regularization:

NP-hard problem
Virmaux et al. [7]

$$\arg\min_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} \underbrace{\ell\left(h(\mathbf{x}_i + \tau^{\text{adv}}(\mathbf{x}_i, y_i; h)), y_i\right)}_{\text{Adversarial Training}} + \boxed{\underbrace{\lambda \operatorname{Lip}(h)}_{\text{Lipschitz Regularization}}}$$

Bounding $\operatorname{Lip}(h)$

Tight bound on the Lipschitz

- Autograd
  Virmaux et al. [7]
- Semi-definite Programming
  Fazlyab et al. [2]

Combining Adversarial Training and Lipschitz regularization:

NP-hard problem
Virmaux et al. [7]

$$\arg\min_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} \underbrace{\ell\left(h(\mathbf{x}_i + \tau^{\text{adv}}(\mathbf{x}_i, y_i; h)), y_i\right)}_{\text{Adversarial Training}} + \underbrace{\lambda \operatorname{Lip}(h)}_{\text{Lipschitz Regularization}}$$

Bounding $\operatorname{Lip}(h)$

Tight bound on the Lipschitz

- Autograd
  Virmaux et al. [7]
- Semi-definite Programming
  Fazlyab et al. [2]

$$\operatorname{Lip}(h) \leq \prod_{i=1}^{p} \sigma_1\left(\mathbf{W}^{(i)}\right)$$

Efficient Computation of Lipschitz Bound

Algorithm 3.1 Power method for producing the largest singular value, σ_1, of a non-square matrix, $\mathbf{W}$ (Golub & Van der Vorst, 2000; Gouk et al. 2018)

Require: affine function $f(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}$, number of iteration N
Ensure: approximation of the Lipschitz constant $\mathrm{Lip}(f)$

1: Randomly initialise $\mathbf{x}$
2: **for** $i = 1$ **to** N **do**
3: $\quad \mathbf{x} \leftarrow \mathbf{W}^\top \mathbf{W}\mathbf{x} / \|\mathbf{x}\|_2$
4: **end for**
5: **return** $\|\mathbf{W}\mathbf{x}\|_2 / \|\mathbf{x}\|_2$

Efficient Computation of Lipschitz Bound

Algorithm 3.2 Convolutional power method (Farnia et al. 2019)

Require: 2d-convolution function $f : \mathbb{R}^{n\times n} \to \mathbb{R}^{m\times m}$ with kernel k, 2d-convolution-transpose function $g : \mathbb{R}^{n\times n} \to \mathbb{R}^{m\times m}$ with kernel k number of iteration N

Ensure: approximation of the Lipschitz constant $\mathrm{Lip}(f)$

1: Initialize $\mathbf{x}$ with a random vector matching the shape of the convolution input
2: **for** $i = 1$ **to** N **do**
3: $\quad \mathbf{x} \leftarrow f(\mathbf{x})/\|f(\mathbf{x})\|_2$
4: $\quad \mathbf{x} \leftarrow g(\mathbf{x})/\|g(\mathbf{x})\|_2$
5: **end for**
6: **return** $\|f(\mathbf{x})\|_2/\|\mathbf{x}\|_2$

Bounding the largest singular value

Exploit the properties of convolutions to find a fast and accurate approximation of their largest singular values.

What is a Structured Matrix?

A $n \times n$ structured matrix can be represented with less than n^2 parameters.

$$\begin{pmatrix} a & b & c & d \\ e & a & b & c \\ f & e & a & b \\ g & f & e & a \end{pmatrix}$$

Toeplitz Matrix

$$\begin{pmatrix} a & b & c & d \\ d & a & b & c \\ c & d & a & b \\ b & c & d & a \end{pmatrix}$$

Circulant Matrix

What is a Structured Matrix?

A $n \times n$ structured matrix can be represented with less than n^2 parameters.

Toeplitz Matrix

Circulant Matrix

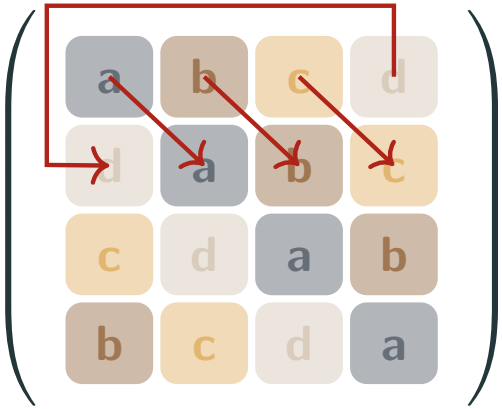
- A Toeplitz matrix is a matrix with constant diagonals

  $\rightarrow$ only $2n - 1$ unique values

What is a Structured Matrix?

A $n \times n$ structured matrix can be represented with less than n^2 parameters.

Toeplitz Matrix



Circulant Matrix

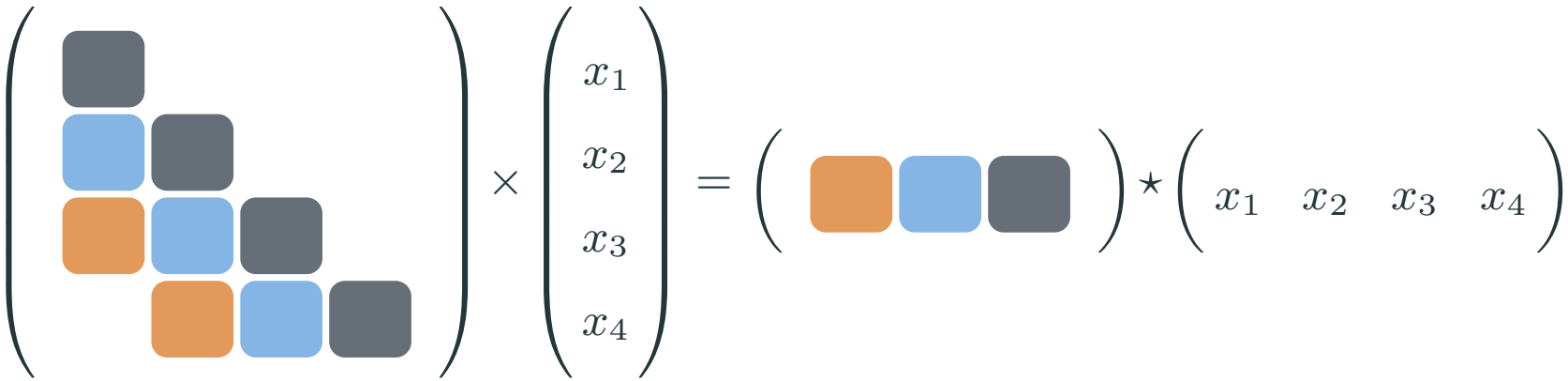
- A Toeplitz matrix is a matrix with constant diagonals
  - $\rightarrow$ only $2n - 1$ unique values
- A circulant matrix is a matrix where each row is a cyclic right shift of the previous one
  - $\rightarrow$ only n unique values

Toeplitz matrices are Related to Convolutions

A discrete convolution between a signal and a kernel can be expressed as a product between the vectorization of the signal and a matrix.

Toeplitz matrices are Related to Convolutions

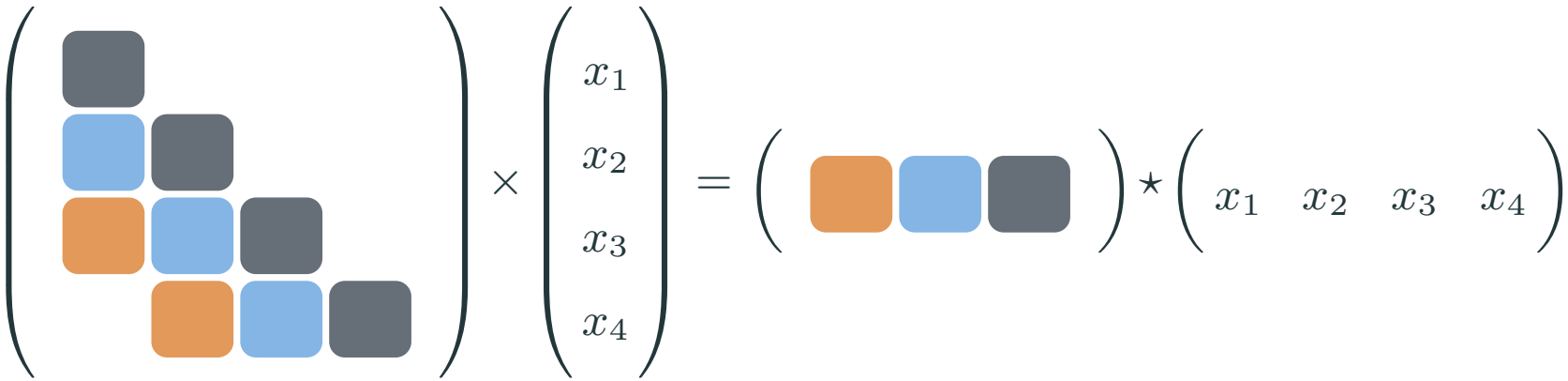
A discrete convolution between a signal and a kernel can be expressed as a product between the vectorization of the signal and a matrix.



- 1d convolution $\rightarrow$ Toeplitz matrices

Toeplitz matrices are Related to Convolutions

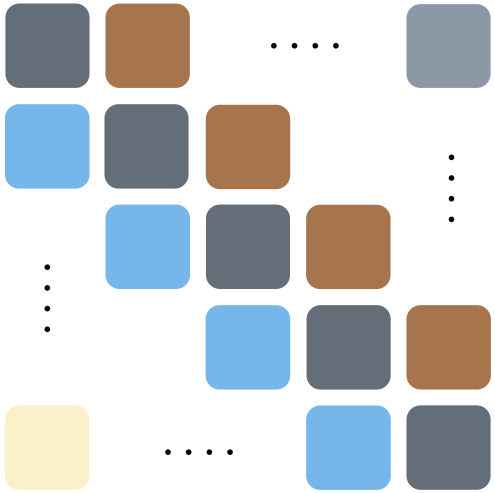
A discrete convolution between a signal and a kernel can be expressed as a product between the vectorization of the signal and a matrix.



- 1d convolution $\rightarrow$ Toeplitz matrices
- 2d convolution $\rightarrow$ **Doubly-block Toeplitz matrices**

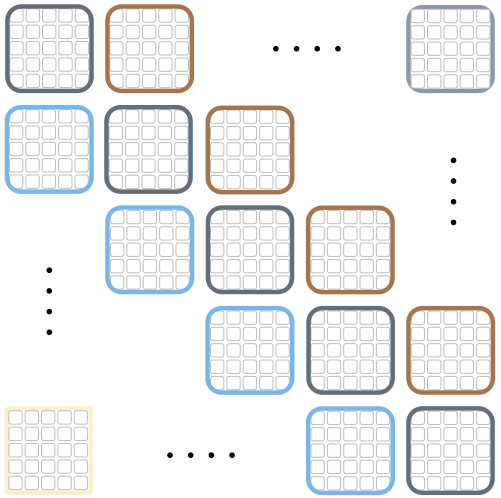
Doubly-Block Toeplitz Matrices

Doubly-block Toeplitz Matrices are structured matrices from the Toeplitz Family.



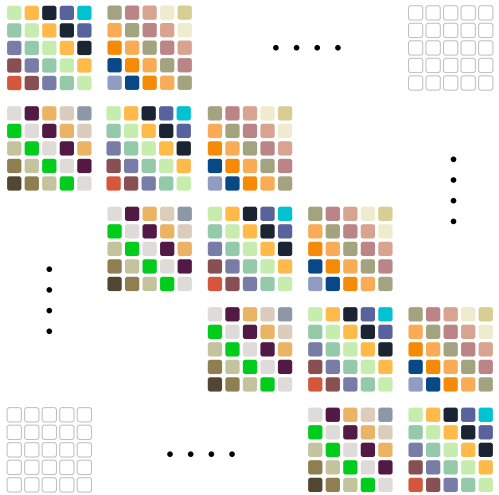
Doubly-Block Toeplitz Matrices

Doubly-block Toeplitz Matrices are structured matrices from the Toeplitz Family.



Doubly-Block Toeplitz Matrices

Doubly-block Toeplitz Matrices are structured matrices from the Toeplitz Family.



A **doubly-block Toeplitz matrix** is a block Toeplitz matrix where all blocks are also Toeplitz.

We can easily bound the largest singular value of Toeplitz matrices

Theorem ([4])

Let $\mathbf{T}$ be a Toeplitz matrix and f be the inverse Fourier Transform of the Toeplitz sequence, then:

$$\sigma_1(\mathbf{T}) \leq \sup_{\omega \in [0,2\pi]} |f(\omega)|$$

In this result, $f : [0, 2\pi] \to \mathbb{C}$ is a complex-valued function of the following form:

$$\mathbf{T} = \begin{pmatrix} a_0 & a_1 & \cdots & a_{n-1} \\ a_{-1} & a_0 & \ddots & \vdots \\ \vdots & \ddots & a_0 & a_1 \\ a_{-n+1} & \cdots & a_{-1} & a_0 \end{pmatrix} \qquad f(\omega) = \sum_{h=-n+1}^{n-1} a_h e^{\mathbf{i}h\omega}$$

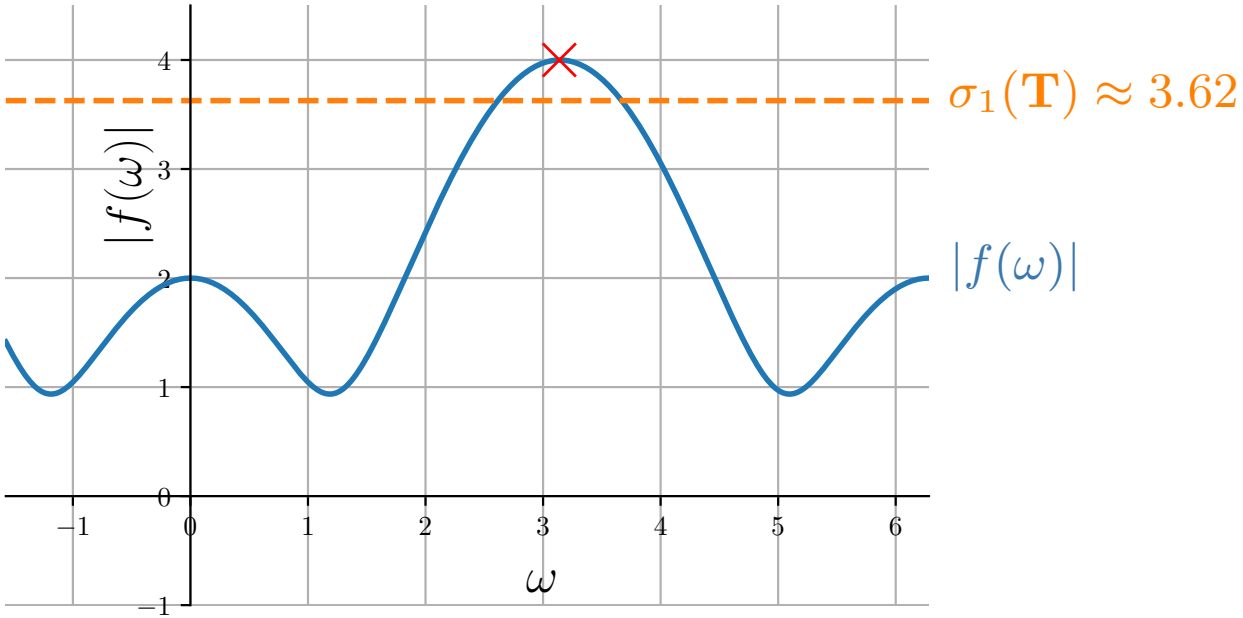
Remark: The function f is a trigonometric polynomial of degree $n-1$

Example with a Banded Toeplitz Matrix

$$\mathbf{T} = \overbrace{\begin{pmatrix} -1 & 1 & 0 & 0 & 0 \\ 2 & -1 & 1 & 0 & 0 \\ 0 & 2 & -1 & 1 & 0 \\ 0 & 0 & 2 & -1 & 1 \\ 0 & 0 & 0 & 2 & -1 \end{pmatrix}}^{n=5}$$

$$f(\omega) = 2e^{-\mathbf{i}\omega} - 1 + e^{\mathbf{i}\omega}$$

$$\sup_{\omega \in [0,2\pi]} |f(\omega)| = 4$$



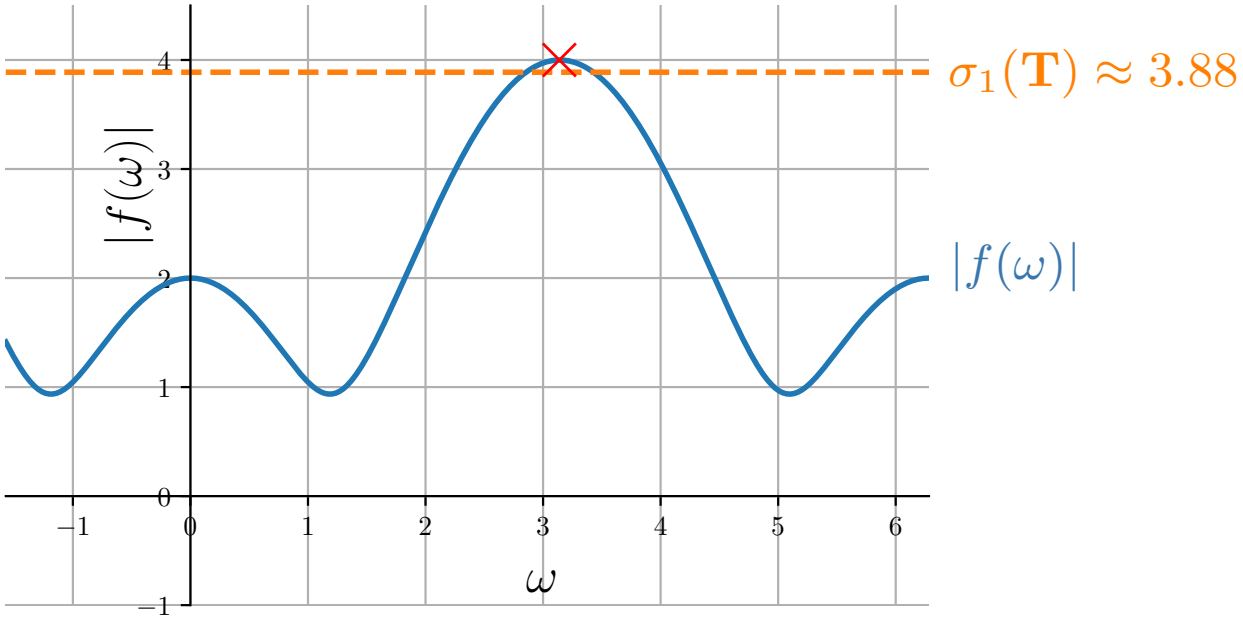
Example with a Banded Toeplitz Matrix

$$\mathbf{T} = \overbrace{\begin{pmatrix} -1 & 1 & \cdots & 0 & 0 \\ 2 & -1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & -1 & 1 \\ 0 & 0 & \cdots & 2 & -1 \end{pmatrix}}^{n=10}$$

The function f stays the same regardless of the size of the matrix.

$$f(\omega) = 2e^{-\mathbf{i}\omega} - 1 + e^{\mathbf{i}\omega}$$

$$\sup_{\omega \in [0,2\pi]} |f(\omega)| = 4$$



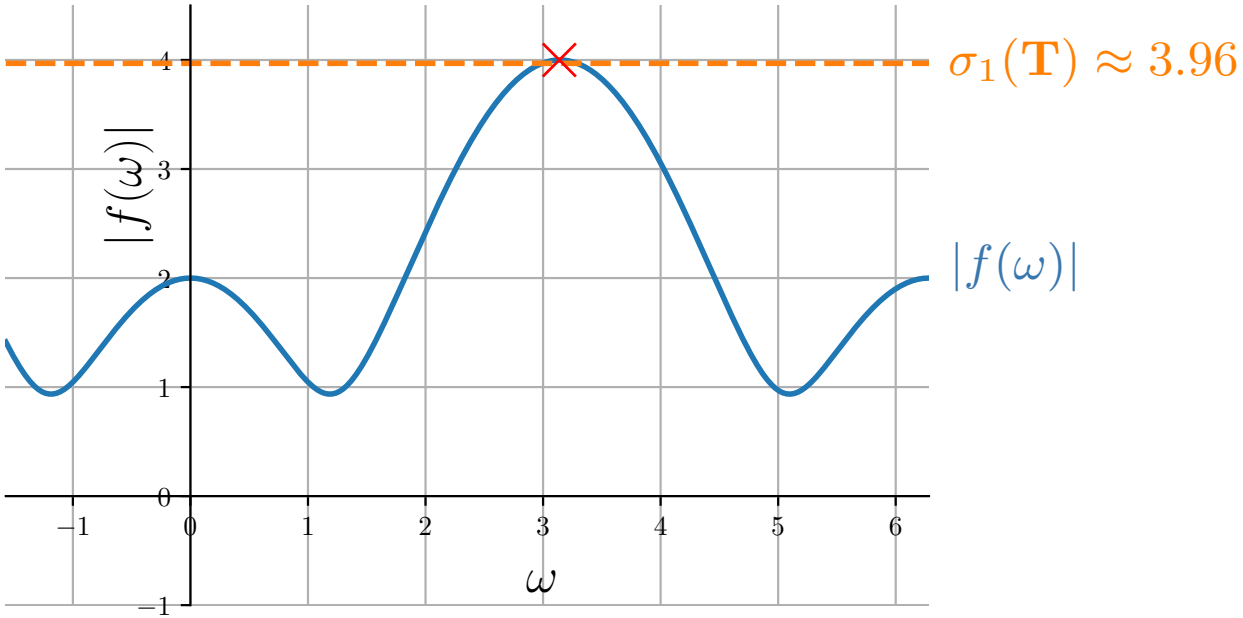
Example with a Banded Toeplitz Matrix

$$\mathbf{T} = \overbrace{\begin{pmatrix} -1 & 1 & \cdots & 0 & 0 \\ 2 & -1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & -1 & 1 \\ 0 & 0 & \cdots & 2 & -1 \end{pmatrix}}^{n=20}$$

The function f stays the same regardless of the size of the matrix.

$$f(\omega) = 2e^{-\mathbf{i}\omega} - 1 + e^{\mathbf{i}\omega}$$

$$\sup_{\omega \in [0,2\pi]} |f(\omega)| = 4$$



Equivalent Reasoning for Block Toeplitz Matrices

Let $g : [0, 2\pi] \to \mathbb{C}^{m\times m}$ be a matrix-valued function and the inverse Fourier Transform of the sequence $\{\mathsf{B}_h\}_{\{-n+1,\dots,n-1\}}$:

$$\mathbf{B} = \begin{pmatrix} \mathsf{B}_0 & \mathsf{B}_1 & \cdots & \mathsf{B}_{n-1} \\ \mathsf{B}_{-1} & \mathsf{B}_0 & \ddots & \vdots \\ \vdots & \ddots & \mathsf{B}_0 & \mathsf{B}_1 \\ \mathsf{B}_{-n+1} & \cdots & \mathsf{B}_{-1} & \mathsf{B}_0 \end{pmatrix} \qquad g(\omega) = \sum_{h=-n+1}^{n-1} \mathsf{B}_h e^{\mathbf{i}h\omega}$$

Theorem ([5])

Let $\mathbf{B}$ a block Toeplitz matrix and g the inverse Fourier transform of its associated sequence, then:

$$\sigma_1(\mathbf{B}) \leq \sup_{\omega\in[0,2\pi]} \sigma_1(g(\omega))$$

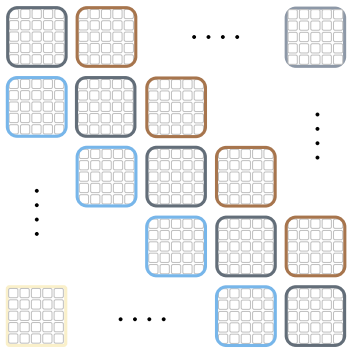
Extension to Doubly-Block Toeplitz Matrices

We extend the reasoning to doubly-block Toeplitz matrices with **2d trigonometric polynomials**.

Extension to Doubly-Block Toeplitz Matrices

We extend the reasoning to doubly-block Toeplitz matrices with **2d trigonometric polynomials**.

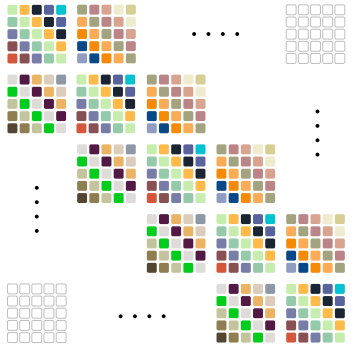
- The first dimension generates the block Toeplitz structure



Extension to Doubly-Block Toeplitz Matrices

We extend the reasoning to doubly-block Toeplitz matrices with **2d trigonometric polynomials**.

- The first dimension generates the block Toeplitz structure
- The second dimension generates the Toeplitz structure of each block



Bound on the Singular Values of Doubly-Block Toeplitz Matrices

In the same way as [4], we can bound the singular values of doubly-block Toeplitz matrices with their associated function.

Bound on the Singular Values of Doubly-Block Toeplitz Matrices

In the same way as [4], we can bound the singular values of doubly-block Toeplitz matrices with their associated function.

Theorem (Bound on the singular values of a Doubly-Block Toeplitz)

Let $\mathbf{D}$ *be a doubly-block Toeplitz matrix and* f *its associated function, then:*

$$\sigma_1(\mathbf{D}) \leq \sup_{\omega_1,\omega_2 \in [0,2\pi]^2} |f(\omega_1,\omega_2)|$$

Bound on the Singular Values of Doubly-Block Toeplitz Matrices

In the same way as [4], we can bound the singular values of doubly-block Toeplitz matrices with their associated function.

Theorem (Bound on the singular values of a Doubly-Block Toeplitz)

Let $\mathbf{D}$ be a doubly-block Toeplitz matrix and f its associated function, then:

$$\sigma_1(\mathbf{D}) \leq \sup_{\omega_1,\omega_2 \in [0,2\pi]^2} |f(\omega_1,\omega_2)|$$

- Let ϕ be a convolution layer and $\mathbf{W}$ the doubly-block Toeplitz equivalent to the convolution

Bound on the Singular Values of Doubly-Block Toeplitz Matrices

In the same way as [4], we can bound the singular values of doubly-block Toeplitz matrices with their associated function.

Theorem (Bound on the singular values of a Doubly-Block Toeplitz)

Let $\mathbf{D}$ be a doubly-block Toeplitz matrix and f its associated function, then:

$$\sigma_1(\mathbf{D}) \leq \sup_{\omega_1,\omega_2 \in [0,2\pi]^2} |f(\omega_1,\omega_2)|$$

- Let ϕ be a convolution layer and $\mathbf{W}$ the doubly-block Toeplitz equivalent to the convolution
- Let f be the function associated with the matrix $\mathbf{W}$

Bound on the Singular Values of Doubly-Block Toeplitz Matrices

In the same way as [4], we can bound the singular values of doubly-block Toeplitz matrices with their associated function.

Theorem (Bound on the singular values of a Doubly-Block Toeplitz)

Let $\mathbf{D}$ *be a doubly-block Toeplitz matrix and* f *its associated function, then:*

$$\sigma_1(\mathbf{D}) \leq \sup_{\omega_1,\omega_2 \in [0,2\pi]^2} |f(\omega_1,\omega_2)|$$

- Let ϕ be a convolution layer and $\mathbf{W}$ the doubly-block Toeplitz equivalent to the convolution
- Let f be the function associated with the matrix $\mathbf{W}$

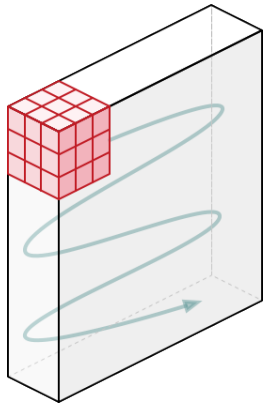
$\rightarrow$ Then, we can upper bound the Lipschitz constant of the convolution:

$$\text{Lip}(\phi) \leq \sigma_1(\mathbf{W}) \leq \sup_{\omega_1,\omega_2 \in [0,2\pi]^2} |f(\omega_1,\omega_2)|$$

Bound on the Singular Values of Convolution

In practice, images have multiple **channels**:

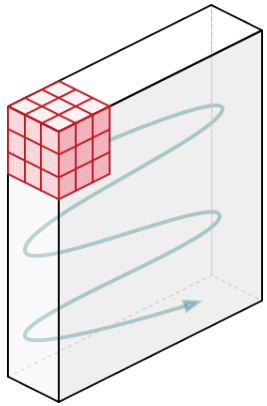
- Input channels (*e.g.*, RGB)
- Output channels (number of convolutions performed on the image)



Bound on the Singular Values of Convolution

In practice, images have multiple **channels**:

- Input channels (*e.g.*, RGB)
- Output channels (number of convolutions performed on the image)



A multi-channel convolution is equivalent to a matrix-vector product where the matrix is a **block matrix** where each block is a doubly-block Toeplitz matrix.

$$\mathbf{M} = \overset{\text{Channels out} \longrightarrow}{\begin{pmatrix} \mathbf{D}_1 & \cdots & \mathbf{D}_{1,cout} \\ \vdots & & \vdots \\ \mathbf{D}_{c}in & \cdots & \mathbf{D}_{cin,cout} \end{pmatrix}} \quad \text{Channels in} \downarrow$$

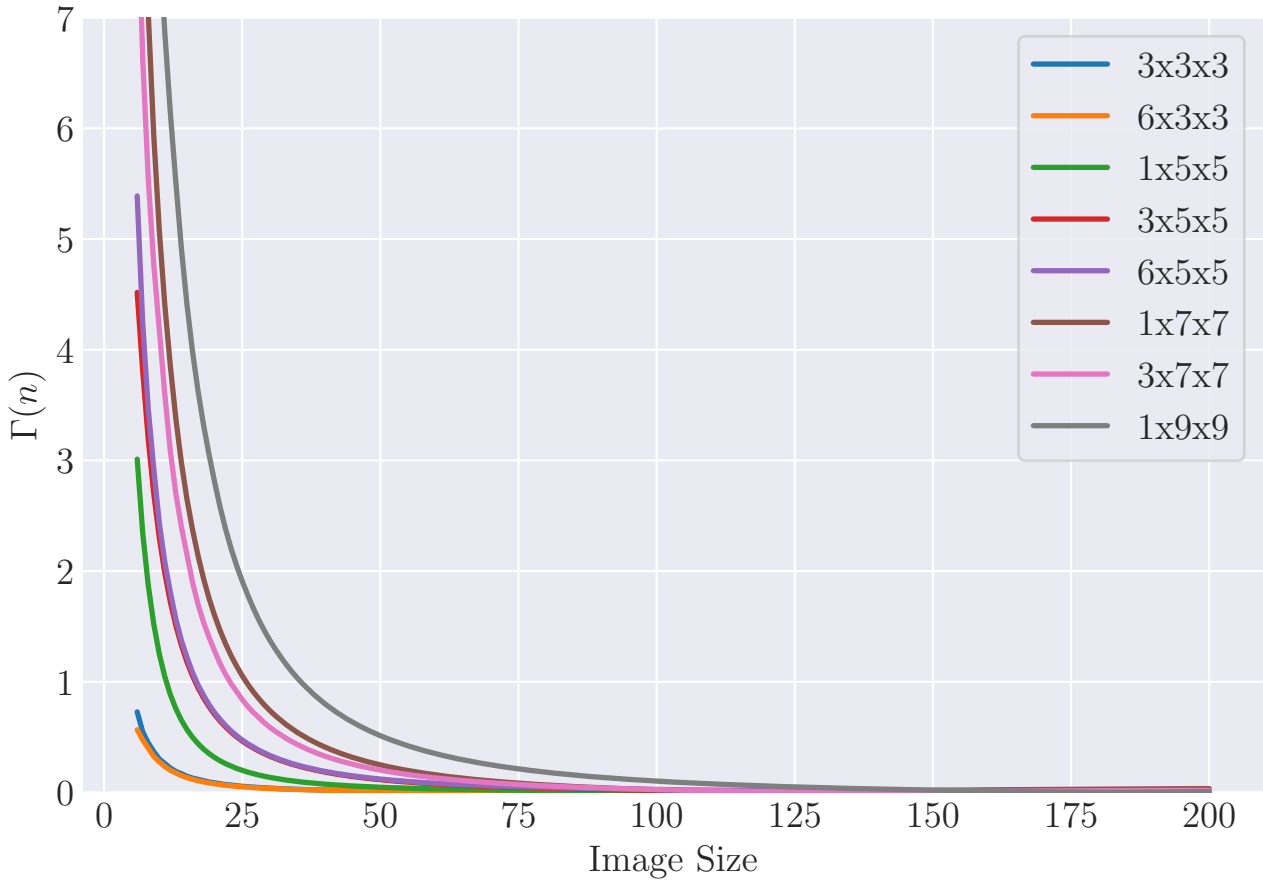
Theorem (Bound on the singular values of convolution)

$$\sigma_1(\mathbf{M}) \leq \sqrt{\sum_{i=1}^{cout} \sup_{\omega_1,\omega_2 \in [0,2\pi]^2} \sum_{j=1}^{cin} |f_{ij}(\omega_1,\omega_2)|^2} \triangleq \text{LipBound}(\mathbf{M})$$

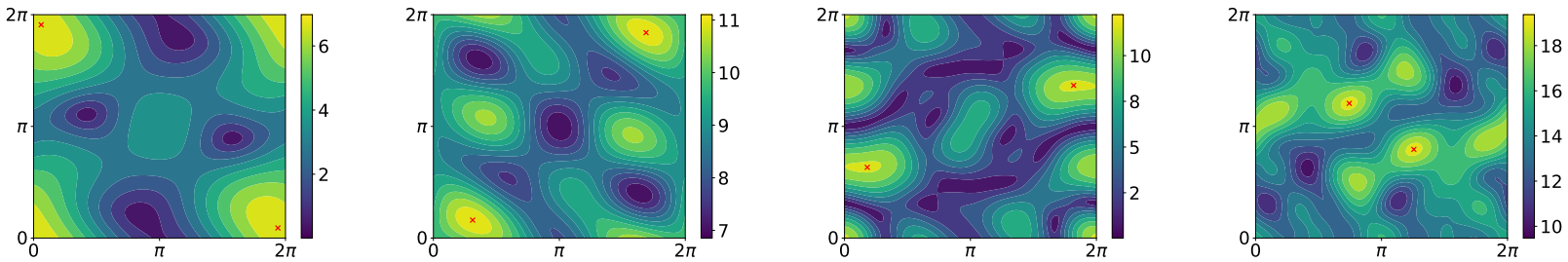
Tightness of LipBound

- The size of a matrix convolution is dependant on the size of the input
- When the size of the matrix increases, the associated function stays the same

$$\Gamma(n) = \text{LipBound}(\mathbf{M}(n)) - \sigma_1(\mathbf{M}(n))$$



Computing LipBound

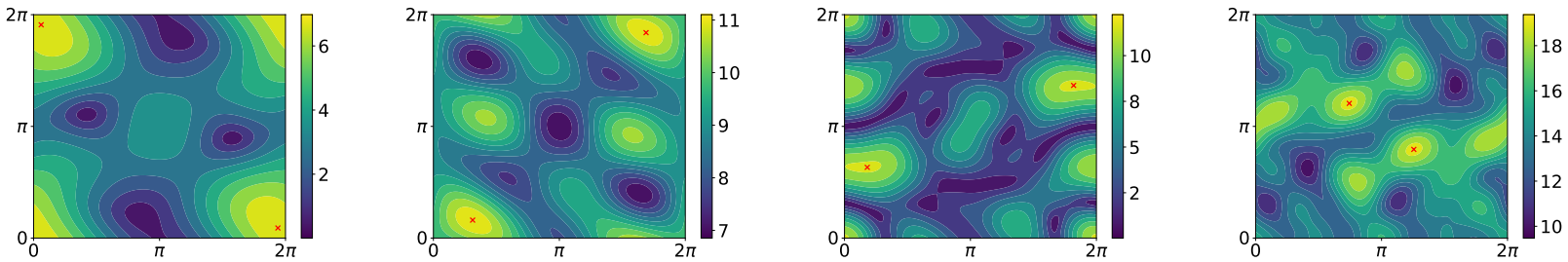


kernel $1 \times 3 \times 3$ kernel $9 \times 3 \times 3$ kernel $1 \times 5 \times 5$ kernel $9 \times 5 \times 5$

Contour plot of multivariate trigonometric polynomials.

The computation of $\mathrm{LipBound}$ involves computing the maximum modulus of a 2-dimensional trigonometric polynomial on $[0, 2\pi]^2$.

Computing LipBound



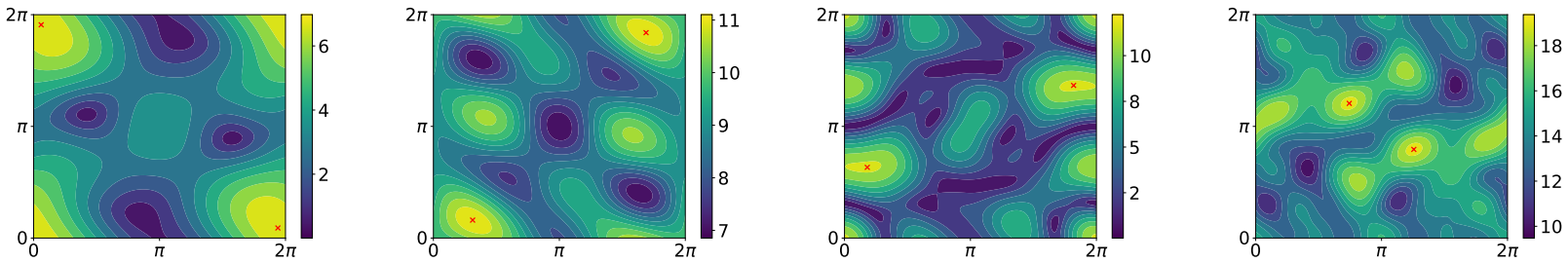
kernel $1 \times 3 \times 3$ kernel $9 \times 3 \times 3$ kernel $1 \times 5 \times 5$ kernel $9 \times 5 \times 5$

Contour plot of multivariate trigonometric polynomials.

The computation of $\mathrm{LipBound}$ involves computing the maximum modulus of a 2-dimensional trigonometric polynomial on $[0, 2\pi]^2$.

- This problem has been known to be NP-hard ([6]).

Computing LipBound



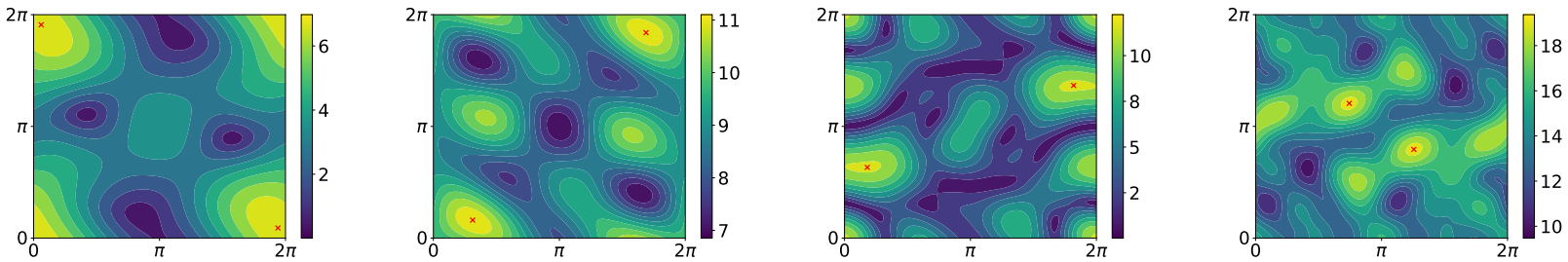
kernel $1 \times 3 \times 3$ kernel $9 \times 3 \times 3$ kernel $1 \times 5 \times 5$ kernel $9 \times 5 \times 5$

Contour plot of multivariate trigonometric polynomials.

The computation of $\mathrm{LipBound}$ involves computing the maximum modulus of a 2-dimensional trigonometric polynomial on $[0, 2\pi]^2$.

- This problem has been known to be NP-hard ([6]).
- However, trigonometric polynomials defined by usual convolutional kernels have a low degree (between 1 and 3)

Computing LipBound



kernel $1 \times 3 \times 3$ kernel $9 \times 3 \times 3$ kernel $1 \times 5 \times 5$ kernel $9 \times 5 \times 5$

Contour plot of multivariate trigonometric polynomials.

The computation of $\mathrm{LipBound}$ involves computing the maximum modulus of a 2-dimensional trigonometric polynomial on $[0, 2\pi]^2$.

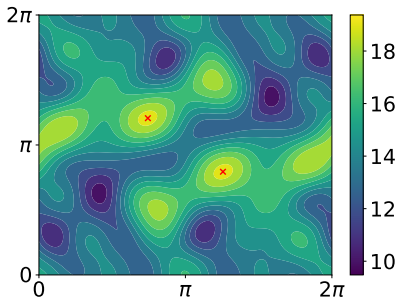
- This problem has been known to be NP-hard ([6]).
- However, trigonometric polynomials defined by usual convolutional kernels have a low degree (between 1 and 3)
- A simple grid search algorithm is efficient and can be implemented on GPU

Precision of the Grid Search Algorithm

- We can characterize the error of the grid search algorithm with respect to the number of samples S.

Precision of the Grid Search Algorithm

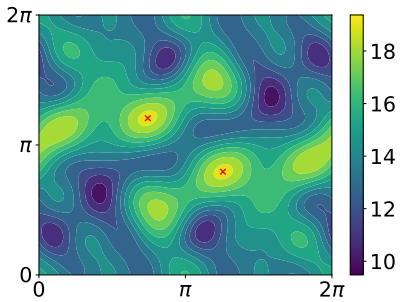
- We can characterize the error of the grid search algorithm with respect to the number of samples S.
- Let $f : [0, 2\pi]^2 \to \mathbb{C}$ be a two dimensional trigonometric polynomial.



$$\sup_{\omega_1, \omega_2 \in [0, 2\pi]^2} |f(\omega_1, \omega_2)|$$

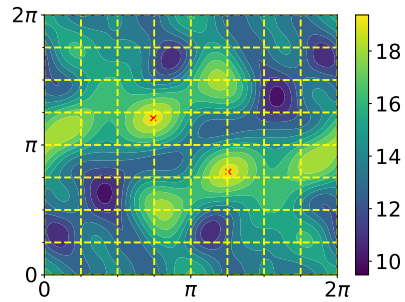
Precision of the Grid Search Algorithm

- We can characterize the error of the grid search algorithm with respect to the number of samples S.
- Let $f : [0, 2\pi]^2 \to \mathbb{C}$ be a two dimensional trigonometric polynomial.



$$\sup_{\omega_1, \omega_2 \in [0, 2\pi]^2} |f(\omega_1, \omega_2)|$$

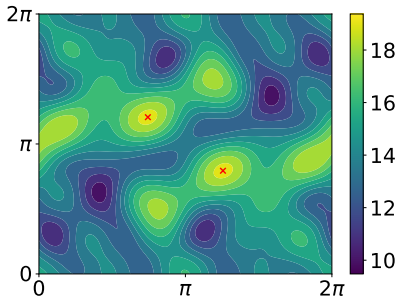
$\Longrightarrow$
discretization of
the search space



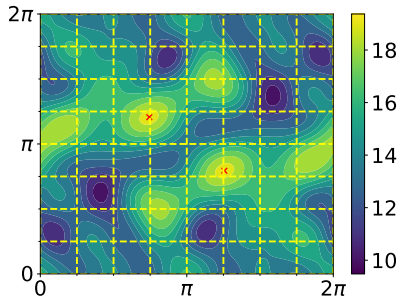
$$\max_{\omega_1', \omega_2' \in \Theta_S^2} \left| f(\omega_1', \omega_2') \right|$$

Precision of the Grid Search Algorithm

- We can characterize the error of the grid search algorithm with respect to the number of samples S.
- Let $f : [0, 2\pi]^2 \to \mathbb{C}$ be a two dimensional trigonometric polynomial.



$\Longrightarrow$
discretization of
the search space



$$\sup_{\omega_1,\omega_2 \in [0,2\pi]^2} |f(\omega_1, \omega_2)| \leq \frac{1}{(1-\alpha)} \max_{\omega'_1,\omega'_2 \in \Theta_S^2} \left|f(\omega'_1, \omega'_2)\right|$$

where $\alpha = 2d/S$ is the degree of the polynomial ([6])

We want to improve upon Adversarial Training:

$$\arg\min_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} \ell(h(\mathbf{x}_i + \tau^{\text{adv}}(\mathbf{x}_i, y_i; h)), y_i) + \underbrace{\lambda \sum_{j=1}^{p} \log\Big(\text{LipBound}(\mathbf{W}^{(j)})\Big)}_{\text{Our regularization}}$$

where λ is a user-defined parameter which controls the regularization.

$\Rightarrow$ Evaluation of the robustness with l_2 norm and several ϵ

Empirical Results

CIFAR10 Dataset
50K images
10 classes

	Accuracy	$\epsilon = 0.6$	$\epsilon = 0.8$
Baseline	**95**	0	0
AT	86	47	33
AT+LipReg	80	**54**	**43**
		+7	+10

Empirical Results

CIFAR10 Dataset
50K images
10 classes

	Accuracy	$\epsilon = 0.6$	$\epsilon = 0.8$
Baseline	**95**	0	0
AT	86	47	33
AT+LipReg	80	**54**	**43**
		+7	+10

ImageNet Dataset
1.2M images
1000 classes

	Accuracy	$\epsilon = 1$	$\epsilon = 2$
Baseline	**78**	0	0
AT	50	30	16
AT+LipReg	51	**33**	**20**
		+3	+4

Gradient masking and gradient obfuscation

Shattered Gradients are caused when a defense is nondifferentiable, introduces numeric instability, or otherwise causes a gradient to be nonexistent or incorrect

Stochastic Gradients are caused by randomized defenses, where either the network itself is randomized or the input is randomly transformed before being fed to the classifier, causing the gradients to become randomized

Exploding & Vanishing Gradients are often caused by defenses that consist of multiple iterations of neural network evaluation, feeding the output of one computation as the input of the next.

References

[1] Farzan Farnia et al. Generalizable adversarial training via spectral normalization. In International Conference on Learning Representations, 2019.

[2] Mahyar Fazlyab et al. Efficient and accurate estimation of lipschitz constants for deep neural networks. In Advances in Neural Information Processing Systems, 2019.

[3] Ian Goodfellow et al. Explaining and harnessing adversarial examples. In International Conference on Learning Representations, 2015.

[4] Robert Gray. Toeplitz and circulant matrices: A review. Foundations and Trends® in Communications and Information Theory, 2006.

[5] Jesús Gutiérrez et al. Block toeplitz matrices: Asymptotic results and applications. Foundations and Trends® in Communications and Information Theory, 2012.

References

[6] Luke Pfister and Yoram Bresler. Bounding multivariate trigonometric polynomials with applications to filter bank design. arXiv preprint arXiv:1802.09588, 2018.

[7] Aladin Virmaux et al. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In Advances in Neural Information Processing Systems. 2018.