Building Compact and Robust Deep Neural Networks with Toeplitz Matrices

Alexandre Araujo

PSL UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à Université Paris-Dauphine – PSL Research University

Building Compact and Robust Deep Neural Networks with Toeplitz Matrices

Soutenue par

Alexandre Araujo

Le 1er Juin 2021

École doctorale n°543

Sciences de la Décision, des Organisations, de la Société et de l'Echange

Spécialité

Informatique

Composition du jury :

Elisa FROMONT Professeur, IRISA Rennes	*Présidente du Jury*
Teddy FURON Chargé de recherche, INRIA Rennes	*Rapporteur*
Alain RAKOTOMAMONJY Professeur, CRITEO	*Rapporteur*
Krzysztof CHOROMANSKI Research Scientist, Google	*Examinateur*
Remi GRIBONVAL Directeur de recherche, INRIA Lyon	*Examinateur*
Jamal ATIF Professeur, Paris-Dauphine	*Directeur de thèse*
Yann CHEVALEYRE Professeur, Paris-Dauphine	*Co-Encadrant*
Benjamin NEGREVERGNE Maître de conférences, Paris-Dauphine	*Co-Encadrant*

Dauphine | PSL UNIVERSITÉ PARIS

Building Compact and Robust Deep Neural Networks with Toeplitz Matrices

by

Alexandre Araujo

Dissertation submitted in fulfillment of the requirements for the degree of *Doctor of Philosophy*

at

Université Paris-Dauphine – PSL Research University

under the joint supervision of

Benjamin Negrevergne, **Yann Chevaleyre** and **Jamal Atif**

Dédié à la mémoire de mon grand-père maternel

Jean Marchelie

1936 – 2021

Remerciements

Pour commencer, je souhaiterais remercier Teddy Furon et Alain Rakotomamonjy d'avoir accepté d'être examinateurs de cette thèse ainsi qu'Élisa Fromont, Rémi Gribonval et Krzysztof Choromanski de s'être intéressés à mes travaux de recherche et d'avoir accepté d'être membres du jury. Nos échanges pendant la relecture ainsi que la soutenance ont été très enrichissants.

J'éprouve une profonde gratitude envers mes trois encadrants de thèse, Jamal, Yann et Benjamin durant ces quatre dernières années où j'ai pu découvrir le monde de la recherche et me former à devenir un bon chercheur. En effet, le métier de chercheur est bien le plus beau métier du monde ! Merci d'avoir cru en moi et de m'avoir offert cette chance, merci pour tous ces échanges, désaccords, réunions, sessions de travail et enfin, merci de m'avoir toléré quand j'étais pénible. Vous m'avez vraiment offert un encadrement exceptionnel: peu de doctorants peuvent se targuer de pouvoir échanger avec leurs encadrants tous les jours !

Cette expérience a été d'autant plus riche grâce aux autres doctorants du laboratoire. Ainsi, je souhaite remercier Rafael et Laurent pour les collaborations réalisées ensemble, et également Geovani, Raphaël, Alexandre V., Éric, Virginie et Céline pour avoir participé à l'émulation et la bonne ambiance du labo. Merci également à Florian, Clément et Alexandre A. pour les échanges que l'on a pu avoir.

Cette thèse n'aurait pas été possible sans les financements de Wavestone. Ainsi, je souhaite remercier Cyril pour m'avoir donné cette opportunité ainsi que Nicolas pour nos longs échanges sur mes travaux de recherche. Également, je souhaite remercier David, Hugo et Cédric pour m'avoir aidé au cours de cette expérience.

Je souhaite également remercier ma famille, belle-famille et mes amis pour m'avoir toléré, accompagné, conseillé, encouragé et pour avoir essayé de comprendre ce que je faisais ces quatre dernières années. Désolé d'avoir autant travaillé pendant les week-ends, j'espère ne pas avoir été trop désagréable ou trop "dans mon monde" pendant cette période. Un grand merci à Othmane qui, à travers nos échanges, m'a donné l'idée de faire une thèse et de me lancer dans la recherche.

Pour finir, je souhaite exprimer ma profonde gratitude envers Chuthima, ma partenaire du quotidien. Merci pour tous les compromis faits pour me laisser réaliser cette thèse, pour le soutien quotidien, les encouragements et les relectures. La qualité de cette thèse est énormément due à ta présence à mes côtés.

Abstract

Deep neural networks are state-of-the-art in a wide variety of tasks, however, they exhibit important limitations which hinder their use and deployment in real-world applications. When developing and training neural networks, the accuracy should not be the only concern, neural networks must also be cost-effective and reliable. Although accurate, large neural networks often lack these properties. In this thesis, we leverage the properties of structured matrices from the Toeplitz family to build compact and secure neural networks. Our contributions are twofold.

First, we propose a new neural network architecture that is not only accurate but also compact and easy to train. The purpose of this contribution is to study deep diagonal-circulant neural networks, which are deep neural networks in which weight matrices are the product of diagonal and circulant ones. We perform a theoretical analysis of their expressivity and propose an initialization procedure and an intelligent use of nonlinearity functions to facilitate training. Furthermore, we show that these networks outperform recently introduced deep networks with other types of structured layers. We conduct a thorough experimental study to compare the performance of deep diagonal-circulant networks with state-of-the-art models based on structured matrices and with dense models. We show that our models achieve better accuracy than other structured approaches while requiring 2x fewer weights than the next best approach. Finally, we train compact and accurate deep diagonal-circulant networks on a real-world video classification dataset with over 3.8 million training examples.

Secondly, we propose an approach to build robust neural networks to adversarial examples. In this contribution, we introduce a new Lipschitz regularization for Convolutional Neural Networks that improves the robustness of neural networks. Lipschitz regularity is now established as a key property of modern deep learning with implications in training stability, generalization, robustness against adversarial examples, etc. However, computing the exact value of the Lipschitz constant of a neural network is known to be NP-hard. Recent attempts from the literature introduce upper bounds to approximate this constant that are either efficient but loose or accurate but computationally expensive. In this work, by leveraging the properties of doubly-block Toeplitz matrices, we introduce a new upper bound of the singular values of convolution layers that is both tight and easy to compute. Based on this result we devise an algorithm to train Lipschitz-regularized Convolutional Neural Networks.

Résumé

Les réseaux de neurones profonds sont considérés comme étant état de l'art dans une grande variété de tâches, mais ils présentent des limites importantes qui entravent leur utilisation et leur déploiement. Lors du développement et l'entraînement de réseaux de neurones, la précision ne devrait pas être la seule préoccupation, ils se doivent aussi d'être efficaces et sécurisés. Bien que précis, les réseaux de neurones dotés de nombreux paramètres n'ont souvent pas ces propriétés. Dans cette thèse, nous exploitons les propriétés des matrices structurées de la famille de Toeplitz pour construire des réseaux de neurones compacts et sécurisés. Nous réalisons deux contributions sur ces thématiques.

Premièrement, nous proposons une nouvelle architecture de réseau de neurones précise, mais également compacte et facile à entraîner. L'objectif de cette contribution est d'étudier les réseaux de neurones diagonaux-circulants, qui sont des réseaux de neurones profonds pour lesquels les matrices de poids sont le produit des matrices diagonales et circulantes. Nous effectuons une analyse théorique de leur expressivité et proposons une procédure d'initialisation et une utilisation intelligente des fonctions de non-linéarité qui facilitent leur entraînement. Nous montrons que nos modèles atteignent une meilleure précision que les autres approches structurées tout en nécessitant deux fois moins de paramètres. Enfin, nous entraînons des réseaux de neurones diagonaux-circulants sur un ensemble de données de classification vidéo qui contient plus de 3,8 millions d'exemples.

Deuxièmement, en plus d'être compacts et précis, les réseaux de neurones se doivent d'être sécurisés. Pour améliorer leur robustesse, nous proposons une nouvelle régularisation pour les réseaux convolutifs basée sur la constante de Lipschitz. La régularisation Lipschitz est maintenant établie comme une propriété clé de l'apprentissage profond avec des implications en stabilité, généralisation et robustesse contre les attaques adversariales, etc. Cependant, le calcul de la constante de Lipschitz d'un réseau de neurones est connu pour être un problème NP-complet. De récentes tentatives introduisent des bornes supérieures pour approximer cette constante qui sont soit efficaces, mais peu précises, soit précises mais coûteuses. Dans cette thèse, en exploitant les propriétés des matrices de Toeplitz à bloc de Toeplitz, nous introduisons une nouvelle borne supérieure de cette constante pour les couches convolutionnelles qui est à la fois précise et facile à calculer. Sur la base de ce résultat, nous concevons un algorithme pour entraîner des réseaux de neurones convolutifs avec une régularisation Lipschitz.

Table of Contents

List of Figures

List of Tables

Acronyms

a.k.a.	**A**lso **K**own **a**s
e.g.	**E**xempli **G**ratia
i.e.	**I**d **E**st
cf.	**C**on**f**er
s.t.	**S**uch **T**hat
i.i.d.	**I**dentically and **I**ndependently **D**istributed
p.d.f.	**P**robability **D**ensity **F**unction
ERM	**E**mpirical **R**isk **M**inimization
CNN	**C**onvolutional **N**eural **N**etwork
DCNN	**D**iagonal **C**irculant **N**eural **N**etwork
PGD	**P**rojected **G**radient **D**escent
C&W	**C**arlini and **W**agner
DFT	**D**iscrete **F**ourier **T**ransform
FFT	**F**ast **F**ourier **T**ransform
IFFT	**I**nverse **F**ast **F**ourier **T**ransform
SVD	**S**ingular **V**alue **D**ecomposition

List of Symbols

$\mathbb{N}$	The set of natural numbers
$\mathbb{R}, \mathbb{R}^n, \mathbb{R}^{n\times m}$	The set of real numbers, vectors and matrices
$\mathbb{C}, \mathbb{C}^n, \mathbb{C}^{n\times m}$	The set of complex numbers, vectors and matrices
$\mathbb{R}_+$, $\mathbb{R}_-$	The set of positive and negative real numbers respectivly
$[n]$	The set $\{x \in \mathbb{N} \mid 1 \leq x \leq n\}$
$[a, b]$	The set $\{x \in \mathbb{R} \mid a \leq x \leq b\}$
$\mathcal{I}_n$, $\mathcal{I}_n^+$	The sets $\{-n+1, \ldots, n-1\}$ and $\{0, \ldots, n-1\}$ respectively
$\mathbf{i} = \sqrt{-1}$	Imaginary number
$\mathfrak{R}(z), \mathfrak{I}(z)$	Real part and imaginary part of complex number z
$\lvert a + \mathbf{i}b\rvert$	Modulus of complex number $a + \mathbf{i}b$, *i.e.*, $\lvert a + \mathbf{i}b\rvert = \sqrt{a^2 + b^2}$
$\mathbb{1}_{[\text{Boolean expres.}]}$	Indicator function (equals 1 if expression is true 0 otherwise)
$\mathbf{M} = (a_{ij})$	Matrix $\mathbf{M}$ with (i, j)-entry a_{ij}
$\mathbf{x} = [\mathbf{x}_1 \ldots \mathbf{x}_n]$	Vector $\mathbf{x}$ of size n with $\mathbf{x}_i$ entries
$\mathbf{M}^\top, \mathbf{M}^*$	Transpose and conjugate transpose of matrix $\mathbf{M}$
$\mathbf{1}_n, \mathbf{0}_n$	n-vector of ones and n-vector of zeros
$\mathbf{e}^{(i)}$	i-th unit vector in $\mathbb{R}^n$
$\mathbf{I}_n, \mathbf{J}_n$	Identity and reflection matrix of size $n \times n$, *i.e.*, $\mathbf{J}^2 = \mathbf{I}$
$\mathbf{U}_n$	DFT matrix of size $n \times n$, *i.e.*, $\mathbf{U}_n = \left(e^{-(2\pi \mathbf{i} jk)/n}\right)_{j,k=0}^{n-1}$
$\lVert\mathbf{x}\rVert_p$	Norm p of vector $\mathbf{x}$, *i.e.*, $\lVert\mathbf{x}\rVert_p = (\sum_{i=0}^n \mathbf{x}_i^p)^{\frac{1}{p}}$
$\lVert\mathbf{x}\rVert_\infty$	Infinity Norm of vector $\mathbf{x}$, *i.e.*, $\lVert\mathbf{x}\rVert_\infty = \max_i \lvert\mathbf{x}_i\rvert$
$\lVert\mathbf{M}\rVert_p$	Norm p of matrix $\mathbf{M}$, *i.e.*, $\lVert\mathbf{M}\rVert_p = \sup_{\lVert\mathbf{x}\rVert_p \neq 0} \frac{\lVert\mathbf{M}\mathbf{x}\rVert_p}{\lVert\mathbf{x}\rVert_p}$
$\lVert\mathbf{M}\rVert_{\mathrm{F}}$	Frobenius Norm of matrix $\mathbf{M}$
$\sigma_1(\mathbf{M})$	Largest singular value of matrix $\mathbf{M}$, *i.e.* $\sigma_1(\mathbf{M}) = \lVert\mathbf{M}\rVert_2$
$\lambda_1(\mathbf{M})$	Largest eigenvalue of matrix $\mathbf{M}$
$\mathbf{M} \geq 0$	$\mathbf{M}$ is positive semidefinite, *i.e.* $\mathbf{x}^*\mathbf{M}\mathbf{x} \geq 0$ for all $\mathbf{x} \in \mathbb{C}^n$
$\mathbf{M} > 0$	$\mathbf{M}$ is positive definite, *i.e.* $\mathbf{x}^*\mathbf{M}\mathbf{x} > 0$ for all $\mathbf{x} \in \mathbb{C}^n$
$\mathbb{P}, \mathbb{E}$	Probability and expectation of a random variable
$\mathcal{N}$	Gaussian distribution

Chapter 1

Introduction

Contents

1.1 Context and Motivation

Since the dawn of computer science, researchers have been trying to emulate intelligence through computers. Alan Turing was the first, in a paper called *Computing Machinery and Intelligence* (Turing, 1950), to lay the foundation for what we now call *Artificial Intelligence*. In the last 20 years, with the surge in data collection and computing resources, the interest and use cases for Machine Learning have grown exponentially. More specifically, Deep Learning, a subfield of Machine Learning, consisting of training Deep Neural Networks on high-level data (images, sounds, texts) have shown great achievements, even outperforming humans on certain tasks.

One of the most remarkable breakthroughs of Deep Learning happened in 2012 during the ImageNet Large-Scale Visual Recognition Challenge (Russakovsky et al. 2015). The challenge aims at evaluating different algorithms for object detection and image classification. In 2012, Krizhevsky et al. obtained 1st place and beat every other participant by a 10.8% margin with a neural network architecture called **AlexNet**. The main reasons for this success are twofold. First, they used a convolutional neural network (CNN) with more than 60 million parameters which was one of the

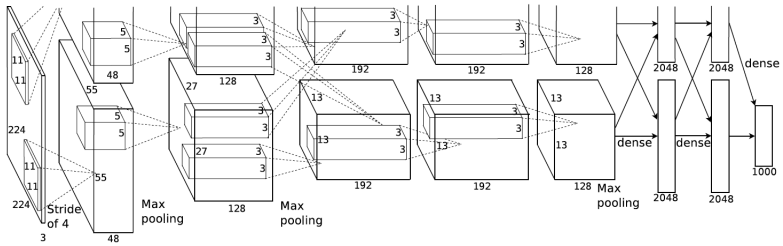


Figure 1.1: The neural network architecture (AlexNet) proposed by Krizhevsky et al. (2012) which won the ImageNet Large-Scale Visual Recognition Challenge in 2012.

largest models of the time. Secondly, they designed a specific architecture to exploit dual programmable graphics processing units (GPUs) to speed up the arithmetic operations, which enabled them to significantly reduce training time. Figure 1.1 shows the AlexNet architecture which consists of five convolution layers with two fully connected layers at the end.

Following this result, many architectures with an increasing number of parameters have been developed. This growth in the number of parameters has led to an substantial gains in accuracy, exceeding even human performance, on the ImageNet dataset (He et al. 2015). Table 1.1 shows a list of the different state-of-the-art architectures along with their size and accuracy. As we can see, the accuracy of the models generally improves at the cost of the model size. For computer vision models, Tan & Le (2019) have empirically shown that the relationship between model size and accuracy seems to obey a power law. This relationship has also been observed for neural networks designed for Natural Language Processing (NLP) (Kaplan et al. 2020; Rosenfeld et al. 2020) aided by the availability of large-scale datasets such as the Common Crawl dataset (Raffel et al. 2020) which constitutes nearly a trillion words.

As a result of their size and improved accuracy, deep neural networks now achieve state-of-the-art performances in a variety of domains such as image recognition (LeCun et al. 1998; Krizhevsky et al. 2012; He et al. 2016; Tan & Le, 2019), object detection (Liu et al. 2016; Redmon et al. 2016; Redmon & Farhadi, 2017), natural language processing (Merity et al. 2016; Vaswani et al. 2017; Radford et al. 2019; Brown et al. 2020), speech recognition (Hinton et al. 2012; Abdel-Hamid et al. 2014; Yu & Deng, 2016), games (Silver et al. 2017), etc. Specifically, computer vision and natural language processing models have achieved sufficient performance for being used in real-world applications such as autonomous vehicles (Sadat et al. 2019), translation (Bahdanau et al. 2015), vocal assistants (Li et al. 2017a), etc.

Authors	Models	#Params	TOP-5 Acc.
Krizhevsky et al. (2012)	AlexNet	61 M	84.7%
Simonyan & Zisserman (2014)	VGG	144 M	92.0%
He et al. (2016)	ResNet-152	60 M	93.8%
Szegedy et al. (2017)	Inception-ResNet-v2	56 M	95.1%
Xie et al. (2017)	ResNeXt-101	84 M	95.6%
Hu et al. (2018)	SENet	146 M	96.2%
Real et al. (2019)	AmoebaNet-A	469 M	96.7%
Huang et al. (2019)	AmoebaNet-B	556 M	97.0%

(a) Computer Vision Models

Authors	Models	#Params
Peters et al. (2018)	ELMo	94 M
Radford et al. (2018)	GPT	110 M
Devlin et al. (2019)	BERT	340 M
Yang et al. (2019)	XLNet (Large)	340 M
Liu et al. (2019)	RoBERTa (Large)	355 M
Radford et al. (2019)	GPT-2	1 B
Shoeybi et al. (2019)	MegatronLM	8 B
Raffel et al. (2020)	T5-11B	11 B
Rosset (2020)	T-NLG	17 B
Brown et al. (2020)	GPT-3	175 B
Fedus et al. (2021)	Switch Transformers	1 T

(b) Natural Language Processing Models

Table 1.1: Evolution of the number of parameters for Computer Vision and Natural Language Processing models developed in the years after AlexNet.

However, accuracy is not the only concern, when implemented in a critical decision process, neural networks need to be compact, cost-effective and secure. Although accurate, large neural networks often lack these properties. Indeed, training state-of-the-art models on computer vision or natural language processing tasks requires gigabytes of memory and can take several months on a single GPU (Krizhevsky et al. 2012; Brown et al. 2020). For example, the GPT-3 model proposed by Brown et al. (2020), culminates at 175 billion parameters and requires 355 years of training on a single GPU and $4 600 000 to train on a cloud-computing platform (Li, 2020). It has also been estimated by Strubell et al. (2019) that the training and development costs of the large Transformer model proposed by Vaswani et al. (2017) with neural

$$\begin{pmatrix} a & & & \\ & b & & \\ & & c & \\ & & & d \end{pmatrix} \qquad \begin{pmatrix} a & b & c & d \\ e & a & b & c \\ f & e & a & b \\ g & f & e & a \end{pmatrix} \qquad \begin{pmatrix} ae & af & ag & ah \\ be & bf & bg & bh \\ ce & cf & cg & ch \\ de & df & dg & dh \end{pmatrix} \qquad \begin{pmatrix} a & a^2 & a^3 & a^4 \\ b & b^2 & b^3 & b^4 \\ c & c^2 & c^3 & c^4 \\ d & d^2 & d^3 & d^4 \end{pmatrix}$$

diagonal Toeplitz Low Rank Vandermonde

Figure 1.2: Examples of structured matrices.

architecture search emits an estimated 284 019 kg of CO_2 whereas a human life will consume an average of 5 000 kg of CO_2 for one year. Furthermore, with the rise of smartphones and "Internet of things" devices (IoT) with limited computational and memory resources, neural networks also need to be efficient during the inference phase. In addition, with the growing concern over data privacy, methods such as *federated learning* are gaining ground. Federated learning involves training a model across multiple decentralized devices (*e.g.*, smartphones) with local data samples. This avoids the step of centralizing all users' data into one server, thus addressing, even modestly, the issue of data privacy. Thus, building compact and cost-effective neural networks have been an important goal in order to reduce training time, reduce cost and allow for faster research and development.

In addition to being compact and cost-effective, neural networks also need to be secure. Due to their high complexity and expressivity, large neural networks exhibit instability to small perturbations of their inputs. Unstable neural networks tend to be vulnerable to *adversarial examples*, *i.e.*, imperceptible variations of natural examples, crafted to deliberately mislead the models (Globerson & Roweis, 2006; Biggio et al. 2013; Szegedy et al. 2014). Figure 1.3 gives an example of an adversarial attack on an image. The small perturbation (center) is added to the original image (left) leading to an adversarial image (right). This behavior can cause serious security problems when neural networks are used for critical decision-making (*e.g.*, self-driving cars, predictive justice, etc.).

This thesis focuses on the problem of training neural networks which are not only accurate but also compact, easy to train, reliable and robust to adversarial examples.

1.2 Problem Statement and Contributions

Neural networks, which find their roots in the work of McCulloch & Pitts (1943) and Rosenblatt (1958), can be analytically described as a composition of multi-dimensional

Figure 1.3: Example of Adversarial Attack on an image.

linear functions interlaced with nonlinear functions (also called activation functions). More formally, a neural network is a function $N_\Omega : \mathbb{R}^n \to \mathbb{R}^m$ parameterized by a set of weights Ω of the form:

$$N_\Omega(\mathbf{x}) = \psi^{(p)} \circ \rho \circ \psi^{(p-1)} \cdots \circ \psi^{(2)} \circ \rho \circ \psi^{(1)}(\mathbf{x}) \ . \tag{1.1}$$

Here, p corresponds to the *depth* of the network (*i.e.*, the number of layers) and ρ is a nonlinear function. Each $\psi^{(i)}$ is a multi-dimensional linear function $\psi^{(i)} : \mathbf{x} \mapsto \mathbf{W}^{(i)}\mathbf{x} + \mathbf{b}^{(i)}$ parameterized by a weight matrix $\mathbf{W}^{(i)}$ and a bias vector $\mathbf{b}^{(i)}$ and Ω is the union of the parameters of all the layers.

Classical neural networks typically have a large number of parameters to train. If they have no restrictions on the weight matrices $\mathbf{W}^{(i)}$, the layers are said to be *fully connected*. Typically, fully connected neural networks have a large number of parameters. For example, a fully connected neural network with p layers and n neurons on each layer ($\mathbf{W}^{(i)} \in \mathbb{R}^{n \times n}$) will have $pn(n+1)$ parameters. Since the input and output dimensions are generally large (*e.g.*, ImageNet has an input dimension of $224^2 \times 3$ and an output of 1000), simple fully connected neural networks with few layers accumulate over hundreds of millions of parameters. Generally, this type of neural network has been shown to perform poorly due to a large search space or due to the important expressivity of the model which leads to overfitting[1]. Moreover, they are computationally expensive, which makes them impractical for a number of use cases (smartphones, IoT devices, etc.). To reduce the number of parameters on each layer, researchers have devised specific linear operations that reduce the number of parameters and have better properties for the problem at hand.

[1]For Machine Learning models, overfitting is a well understood phenomenon. However, it has been discovered that large deep neural networks exhibit a “double descent” phenomenon (see Spigler et al. (2019)), where the performance first gets worse (overfits) then gets better with longer training.

An example of widely used neural networks with specialized and more compact linear operations are *Convolutional Neural Networks* (CNN) (LeCun et al. 1998; Krizhevsky et al. 2012; He et al. 2016; Tan & Le, 2019) which achieve state-of-the-art results for computer vision tasks. Convolutional neural networks, which find their roots in the work of Fukushima & Miyake (1982), use specific weight matrices which encode the translation invariant property often desirable to process images. Whereas a classical linear layer with a dense matrix will have $n \times n$ parameters, a convolution layer only has $k \times k$ parameters where $k \ll n$ is the kernel size and is usually small (*e.g.*, 3 or 5 for classical convolution layers). A convolutional neural network is the most common type of *structured* neural networks. Indeed, the convolution operation can be represented by a structured matrix *i.e.*, a matrix that can be represented with less than n^2 parameters.

In addition to offering a more compact representation, the structure of certain matrices can be exploited to obtain better algorithms for the matrix-vector product, thus optimizing memory and computing operations. Based on the success of convolutional neural networks, researchers have studied and proposed other types of neural networks based on weight matrices with different structures (*e.g.*, Sindhwani et al. (2015) and Moczulski et al. (2016)). Figure 1.2 shows different types of structured matrices that have been used for deep learning. Although convolutional neural networks have been state-of-the-art for computer vision tasks, it remains unclear whether other types of structured networks can be beneficial to other types of applications and which type of structure can provide both accuracy and efficient computation.

The contributions of this thesis lie at the intersection of linear algebra, Fourier analysis and deep learning. As a result, we build compact and secure neural networks by leveraging the properties of structured matrices from the Toeplitz family. Hereafter, we summarize our contributions.

1.2.1 Training Compact Neural Networks

As a first contribution, we use circulant matrices, which are a particular type of matrix from the Toeplitz family, to devise a new compact architecture replacing fully connected neural networks. More precisely, we study deep diagonal-circulant neural networks, which are deep neural networks in which weight matrices are replaced by the product of diagonal and circulant ones. Besides making a theoretical analysis of their expressivity, we introduce principled techniques for training these models: we devise an initialization scheme and propose a smart use of nonlinearity functions in order to train deep diagonal-circulant networks. Furthermore, we show that these

networks outperform recently introduced deep networks with other types of structured layers. We conduct a thorough experimental study to compare the performance of these networks with state-of-the-art models. We show that our models achieve better accuracy than other structured approaches while requiring 2x fewer weights than the next best approach. Finally, we train accurate deep diagonal-circulant networks on a real-world video classification dataset with over 3.8 million training examples.

This contribution has been the subject of the following publications:

- *Training Compact Deep Learning Models for Video Classification using Circulant Matrices* in the **European Conference on Computer Vision Workshops on Video Classification**
- *Understanding and Training Deep Diagonal Circulant Neural Networks* in the **24th European Conference on Artificial Intelligence**.

1.2.2 Training Robust Neural Networks

As a second contribution, we build robust neural networks by studying the properties of the structure of convolutions. We devise a new upper bound on the largest singular value of convolution layers that is both tight and easy to compute. Our work is based on the result of Gray (2006) which states that an upper bound on the singular value of Toeplitz matrices can be computed from the inverse Fourier transform of the characteristic sequence of these matrices. From our analysis immediately follows an algorithm for bounding the Lipschitz constant of a convolution layer, and by extension the Lipschitz constant of the whole network. Finally, we illustrate our approach to adversarial robustness. Recent work has shown that empirical methods such as adversarial training offer poor generalization (Schmidt et al. 2018; Rice et al. 2020) and can be improved by applying Lipschitz regularization (Farnia et al. 2019). To illustrate the benefit of our new method, we train neural networks with Lipschitz regularization and show that it offers a significant improvement over adversarial training alone.

Additional joint contributions have also been made on the topic of robust neural networks. A first work studied the effectiveness of noise injection at training and inference time in neural networks to protect against adversarial attacks. In this work, we have shown that noise drawn from the Exponential family offers a provable protection against adversarial attacks. A follow-up work conducts a geometrical analysis of defense mechanisms designed to protect neural networks against several

types of attacks. This work shows that neural networks designed to be robust against one type of adversarial example offer poor protection against other types of attacks.

The contribution on adversarial robustness has been the subject of the following publications:

- *On Lipschitz Regularization of Convolutional Layers using Toeplitz Matrix Theory* in the **35th AAAI Conference on Artificial Intelligence**
- *Theoretical evidence for adversarial robustness through randomization* in the **Advances in Neural Information Processing Systems**.
- *Advocating for Multiple Defense Strategies against Adversarial Examples* in the **European Conference on Machine Learning Workshop for CyberSecurity**

Outline of the Thesis

This thesis is organized in six chapters. First, Chapter 2 gives an introduction to the theory of Toeplitz matrices and on supervised learning and neural networks. This chapter presents the necessary technical tools we will need for presenting the related work and for our contributions. Chapter 3 is dedicated to discussing the state-of-the-art approaches related to our contributions. The chapter is divided into two parts. First, we review some techniques to build compact neural networks with an important focus on techniques that use structured matrices. The second part focuses on presenting regularization methods for improving the robustness of neural networks. Chapter 4 and Chapter 5 constitute our main contributions. Chapter 4 presents results on compact neural networks built from diagonal and circulant matrices. Chapter 5 presents our new regularization scheme to improve the robustness of neural networks based on the properties of doubly-block Toeplitz matrices. Chapter 6 proposes a discussion and some perspectives on our contributions. Appendix B constitutes some complements to Chapter 4. It provides additional experiments on video classification with compact neural networks. Finally, Appendices C and D provide further work on the robustness of neural networks done during this Ph.D. thesis.

Chapter 2

Background

Contents

This chapter gives an overview on the theory of Toeplitz matrices and on supervised learning with neural networks. The first section describes the mathematical properties of Toeplitz matrices and some known theorems that we use in this thesis. A Toeplitz matrix, named after Otto Toeplitz, is a matrix in which each descending diagonal, from left to right, is constant. This simple property has led to interesting theoretical results and numerous applications. We will use a number of these results in the context of neural networks. The second section of this chapter is divided into four parts. First, we review notions of supervised learning which refer to the problem of

optimizing the parameters of a function in order to map an input to an output based on a series of input-output pairs. Then, we formally define neural networks and recall some of their properties. We pursue by introducing the concept of adversarial examples which we will use in Chapter 5. Finally, we present some recent theoretical results on neural networks that allow a better understanding of the contributions of this thesis.

2.1 A Primer on Circulant and Toeplitz Matrices

2.1.1 Properties of Circulant Matrices

A circulant matrix is a matrix in which each descending diagonal, from left to right, is constant and each row of the matrix is a cyclic right shift of the previous one:

$$\mathbf{C} = \begin{pmatrix} c_0 & c_{n-1} & c_{n-2} & \cdots & \cdots & c_1 \\ c_1 & c_0 & c_{n-1} & \ddots & & \vdots \\ c_2 & c_1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & c_{n-1} & c_{n-2} \\ \vdots & & \ddots & c_1 & c_0 & c_{n-1} \\ c_{n-1} & \cdots & \cdots & c_2 & c_1 & c_0 \end{pmatrix} . \tag{2.1}$$

The $n \times n$ circulant matrix $\mathbf{C}$ is fully determined by the sequence of scalars $\{c_h\}_{h \in \mathcal{I}_n^+}$ where $\mathcal{I}_n^+ = \{0, \ldots, n-1\}$. Furthermore, the (j, k) entry of $\mathbf{C}$ is given by

$$(\mathbf{C})_{j,k} = c_{(k-j) \bmod n} \ . \tag{2.2}$$

Algorithm 2.1 Matrix-vector product with a circulant matrix

```
procedure CIRCMUL(c, x)   ▷ first column of the circulant matrix C, vector x
    x̃ ← FFT(x)
    c̃ ← FFT(c)
    y ← IFFT(x̃ ⊙ c̃)                ▷ element-wise vector-vector product
    return y                        ▷ return the result of the product Cx
end procedure
```

In linear algebra, circulant matrices are important due to their numerous properties. Indeed, circulant matrices can be compactly represented in memory using only n values instead of n^2 values required for arbitrary matrices. In addition, algorithms

exist to speed-up the matrix-vector product operation from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log n)$. Finally, circulant matrices commute and are closed under the sum and products. All these properties can be demonstrated with the special diagonalization of circulant matrices with the matrix expansion of the Discrete Fourier Transform (DFT), *i.e.*, Fourier matrix, and an explicit formula of their eigenvalues. The Fourier matrix is of the form:

Definition 2.1 (Fourier Matrix)**.** *The Fourier matrix of order n is defined as follows:*

$$\mathbf{U}_n = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & z_n & z_n^2 & \cdots & z_n^{n-1} \\ 1 & z_n^2 & z_n^4 & \cdots & z_n^{2(n-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & z_n^{n-1} & z_n^{2(n-1)} & \cdots & z_n^{(n-1)(n-1)} \end{pmatrix} , \tag{2.3}$$

where $z_n = e^{-\frac{2\pi \mathbf{i}}{n}}$ is an nth root of unity.

The diagonalization of circulant matrices is given by the following theorem:

Theorem 2.1 (Davis (1979))**.** *The eigenvalues λ_k and the eigenvectors $\mathbf{y}^{(k)}$ of a circulant matrix $\mathbf{C} = \mathrm{circ}(\mathbf{c})$ with $\mathbf{c} \in \mathbb{R}^n$ are as follows:*

$$\lambda_k = \sum_{j \in \mathcal{I}_n^+} c_j e^{-\frac{2\pi \mathbf{i}}{n} jk} \quad \Leftrightarrow \quad \lambda_k = (\mathbf{U}_n \mathbf{c})_k \ , \tag{2.4}$$

and

$$\mathbf{y}^{(k)} = \frac{1}{\sqrt{n}} \Big(1, e^{-\frac{2\pi \mathbf{i} k}{n}}, \ldots, e^{-\frac{2\pi \mathbf{i} k(n-1)}{n}}\Big)^\top \ . \tag{2.5}$$

Furthermore, the circulant matrix $\mathbf{C}$ can be expressed in the form

$$\mathbf{C} = \frac{1}{n} \mathbf{U}_n^* \mathrm{diag}(\mathbf{U}_n \mathbf{c}) \mathbf{U}_n \ . \tag{2.6}$$

Based on this decomposition, we can state several properties of circulant matrices:

- **Matrix-vector product**: Let $\mathbf{x} \in \mathbb{R}^n$ an arbitrary vector then the product $\mathbf{Cx}$ can be expanded as follows:

$$\mathbf{Cx} = \frac{1}{n} \mathbf{U}_n^* \mathrm{diag}(\mathbf{U}_n \mathbf{c}) \mathbf{U}_n \mathbf{x} \tag{2.7}$$

$$= \frac{1}{n} \mathbf{U}_n^* \big((\mathbf{U}_n \mathbf{c}) \odot (\mathbf{U}_n \mathbf{x})\big) \tag{2.8}$$

where $\odot$ is the element-wise vector multiplication. Thus, the matrix-vector product $\mathbf{Cx}$ can be reduced to an element-wise multiplication between the characteristic vector $\mathbf{c}$ and the vector $\mathbf{x}$ in the Fourier domain. Furthermore, the multiplication between the Fourier matrix $\mathbf{U}_n$ and a vector can be efficiently computed with the *Fast Fourier Transform* (FFT) algorithms (Cooley & Tukey, 1965). Algorithm 2.1 details the steps required to perform the $\mathcal{O}(n \log n)$ multiplication between a circulant matrix and a vector.

- **Closeness under sum**: Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, $\mathbf{X} = \text{circ}(\mathbf{x})$ and $\mathbf{Y} = \text{circ}(\mathbf{y})$ then, $\mathbf{Z} = \mathbf{X} + \mathbf{Y}$ is also a circulant matrix with $\mathbf{Z} = \text{circ}(\mathbf{x} + \mathbf{y})$:

$$\begin{aligned}
\mathbf{X} + \mathbf{Y} &= \left(\frac{1}{n}\mathbf{U}_n^* \text{diag}(\mathbf{U}_n\mathbf{x})\mathbf{U}_n\right) + \left(\frac{1}{n}\mathbf{U}_n^* \text{diag}(\mathbf{U}_n\mathbf{y})\mathbf{U}_n\right) && (2.9)\\
&= \frac{1}{n}\mathbf{U}_n^*(\text{diag}(\mathbf{U}_n\mathbf{x})\mathbf{U}_n + \text{diag}(\mathbf{U}_n\mathbf{y})\mathbf{U}_n) && (2.10)\\
&= \frac{1}{n}\mathbf{U}_n^*(\text{diag}(\mathbf{U}_n\mathbf{x}) + \text{diag}(\mathbf{U}_n\mathbf{y}))\mathbf{U}_n && (2.11)\\
&= \frac{1}{n}\mathbf{U}_n^*(\text{diag}(\mathbf{U}_n(\mathbf{x} + \mathbf{y})))\mathbf{U}_n && (2.12)\\
&= \text{circ}(\mathbf{x} + \mathbf{y}) && (2.13)
\end{aligned}$$

- **Closeness under product**: Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, $\mathbf{X} = \text{circ}(\mathbf{x})$ and $\mathbf{Y} = \text{circ}(\mathbf{y})$ then, $\mathbf{Z} = \mathbf{XY}$ is also a circulant matrix with $\mathbf{Z} = \text{circ}(\mathbf{x} \odot \mathbf{y})$:

$$\begin{aligned}
\mathbf{XY} &= \left(\frac{1}{n}\mathbf{U}_n^* \text{diag}(\mathbf{U}_n\mathbf{x})\mathbf{U}_n\right)\left(\frac{1}{n}\mathbf{U}_n^* \text{diag}(\mathbf{U}_n\mathbf{y})\mathbf{U}_n\right) && (2.14)\\
&= \frac{1}{n^2}\mathbf{U}_n^* \text{diag}(\mathbf{U}_n\mathbf{x})\mathbf{U}_n\mathbf{U}_n^* \text{diag}(\mathbf{U}_n\mathbf{y})\mathbf{U}_n && (2.15)\\
&= \frac{1}{n^2}\mathbf{U}_n^* \text{diag}(\mathbf{U}_n\mathbf{x})(n\mathbf{I})\text{diag}(\mathbf{U}_n\mathbf{y})\mathbf{U}_n && (2.16)\\
&= \frac{1}{n}\mathbf{U}_n^* \text{diag}(\mathbf{U}_n\mathbf{x})\text{diag}(\mathbf{U}_n\mathbf{y})\mathbf{U}_n && (2.17)\\
&= \frac{1}{n}\mathbf{U}_n^* \text{diag}(\mathbf{U}_n(\mathbf{x} \odot \mathbf{y}))\mathbf{U}_n && (2.18)\\
&= \text{circ}(\mathbf{x} \odot \mathbf{y}) && (2.19)
\end{aligned}$$

In this thesis, we also make use of specific type of circulant matrices called *f-circulant matrices* which are one of the building blocks of *low displacement rank operators* presented in Section 2.1.4 and also enjoy compact representation and fast matrix-vector product. An f-unit-circulant matrix is defined as follows:

Definition 2.2 (f-circulant matrix)**.** *Given a vector* $\mathbf{x}$ *and a scalar* f*, the* f*-circulant matrix,* $\mathbf{Z}_f(\mathbf{x})$*, is defined as follows:*

$$\mathbf{Z}_f(\mathbf{x}) \triangleq \begin{pmatrix} \mathbf{x}_0 & f\mathbf{x}_{n-1} & f\mathbf{x}_{n-2} & \cdots & \cdots & f\mathbf{x}_1 \\ \mathbf{x}_1 & \mathbf{x}_0 & f\mathbf{x}_{n-1} & \ddots & & \vdots \\ \mathbf{x}_2 & \mathbf{x}_1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & f\mathbf{x}_{n-1} & f\mathbf{x}_{n-2} \\ \vdots & & \ddots & \mathbf{x}_1 & \mathbf{x}_0 & f\mathbf{x}_{n-1} \\ \mathbf{x}_{n-1} & \cdots & \cdots & \mathbf{x}_2 & \mathbf{x}_1 & \mathbf{x}_0 \end{pmatrix}. \tag{2.20}$$

We denote $\mathbf{Z}_f$ the f-unit-circulant, defined by the vector $(0, 1, \ldots, 0)^\top$, a matrix of the form:

$$\mathbf{Z}_f = \begin{pmatrix} 0 & 0 & 0 & \cdots & \cdots & f \\ 1 & 0 & 0 & \ddots & & \vdots \\ 0 & 1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 & 0 \\ \vdots & & \ddots & 1 & 0 & 0 \\ 0 & \cdots & \cdots & 0 & 1 & 0 \end{pmatrix}. \tag{2.21}$$

The matrix-vector product $\mathbf{Z}_f\mathbf{x}$ scales the last element by f and makes a circular shift on the components of the vector $\mathbf{x}$ by one resulting in $\mathbf{Z}_f\mathbf{x} = (f\mathbf{x}_{n-1}, \mathbf{x}_0, \ldots, \mathbf{x}_{n-2})^\top$.

2.1.2 A Fourier Representation of Toeplitz Matrices

Toeplitz matrices generalize circulant matrices by relaxing the cyclic right shift on the rows. Therefore, a Toeplitz matrix is a matrix in which each descending diagonal, from left to right, is constant, *i.e.*, a matrix of the form:

$$\mathbf{A} = \begin{pmatrix} a_0 & a_{-1} & a_2 & \cdots & \cdots & a_{-n+1} \\ a_1 & a_0 & a_1 & \ddots & & \vdots \\ a_2 & a_1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & a_{-1} & a_{-2} \\ \vdots & & \ddots & a_1 & a_0 & a_{-1} \\ a_{n-1} & \cdots & \cdots & a_2 & a_1 & a_0 \end{pmatrix}. \tag{2.22}$$

The $n \times n$ Toeplitz matrix $\mathbf{A}$ is fully determined by a two-sided sequence of scalars $\{a_h\}_{h \in \mathcal{I}_n}$ where $\mathcal{I}_n = \{-n+1, \ldots, n-1\}$ and the (j,k) entry of $\mathbf{A}$ is given by

$$(\mathbf{A})_{j,k} = a_{k-j} \quad . \tag{2.23}$$

Similarly to their circulant counterpart, Toeplitz matrices can be represented compactly in memory using only $2n-1$ values instead of n^2 values required for arbitrary ones. Toeplitz matrices have been extensively studied in the context of operator and spectral theory (Grenander et al. 1958; Widom, 1965; Böttcher & Silbermann, 2012). One important result regarding Toeplitz matrices is Szegö's theorem (Szegö, 1915) which describes the asymptotic behavior of the determinant of large Toeplitz matrices. Because Toeplitz matrices do not have a closed-form expression for their eigenvalues, studying their spectrum is not as straightforward as their circulant counterpart. In order to devise results on Toeplitz matrices, Grenander et al. (1958) introduced a representation based on the Fourier transform. Indeed, Toeplitz matrices can be generated from a 2π-periodic function where the values of the Toeplitz matrix are the Fourier coefficients of this *generating function*. The spectrum of Toeplitz matrices can be described precisely from the properties of their generating functions. This representation of Toeplitz matrices has been widely studied in contexts such as signal processing, trigonometric moment problems, integral equations and elliptic partial differential equations with boundary conditions, etc. (Parter, 1961; Widom, 1965; Avram, 1988; Serra, 1997; Tilli, 1997, 1998; Tyrtyshnikov & Zamarashkin, 1998)

The Fourier representation of Toeplitz matrices can be described as follows. Let $\{a_h\}_{h \in \mathcal{I}_n}$ be the characteristic sequence of the Toeplitz matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$. Then, the trigonometric polynomial $f : \mathbb{R} \to \mathbb{C}$ of the form

$$f(\omega) = \sum_{h \in \mathcal{I}_n} a_h e^{\mathbf{i} h \omega} \tag{2.24}$$

is the *inverse Fourier transform* of the sequence $\{a_h\}_{h \in \mathcal{I}_n}$. From this function, one can recover the sequence $\{a_h\}_{h \in \mathcal{I}_n}$ using the standard Fourier transform:

$$a_h = \frac{1}{2\pi} \int_0^{2\pi} e^{-\mathbf{i} h \omega} f(\omega) \, \mathrm{d}\omega \quad . \tag{2.25}$$

We can, now, define an operator $\mathbf{T}$ mapping integrable functions to Toeplitz matrices:

$$\mathbf{T}_n(f) \triangleq \left(\frac{1}{2\pi} \int_0^{2\pi} e^{-\mathbf{i}(i-j)\omega} f(\omega) \, \mathrm{d}\omega \right)_{i,j \in \mathcal{I}_n^+} \quad . \tag{2.26}$$

In the following, when it is clear from context, we will write $\mathbf{T}(f)$ instead of $\mathbf{T}_n(f)$.

2.1.3 Block Circulant, Block Toeplitz and the Convolution Operator

Block Toeplitz and Block Circulant Matrices

We can adapt the structure of circulant matrices and their properties to block matrices. A block circulant matrix is a matrix where each block is repeated identically along diagonals and each row of blocks is a cyclic right shift of the previous one. Therefore, an $nm \times nm$ block circulant matrix $\mathbf{A}$ is fully determined by a sequence of blocks $\{\mathsf{A}^{(h)}\}_{h\in\mathcal{I}_n^+}$ and where each block $\mathsf{A}^{(h)}$ is an $m \times m$ matrix. The block circulant matrix $\mathbf{A} = \left(\mathsf{A}^{((k-j) \bmod n)}\right)_{j,k\in\mathcal{I}_n^+}$ is given by

$$\mathbf{A} = \begin{pmatrix} \mathsf{A}^{(0)} & \mathsf{A}^{(n-1)} & \mathsf{A}^{(n-2)} & \cdots & \cdots & \mathsf{A}^{(1)} \\ \mathsf{A}^{(1)} & \mathsf{A}^{(0)} & \mathsf{A}^{(n-1)} & \ddots & & \vdots \\ \mathsf{A}^{(2)} & \mathsf{A}^{(1)} & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \mathsf{A}^{(n-1)} & \mathsf{A}^{(n-2)} \\ \vdots & & \ddots & \mathsf{A}^{(1)} & \mathsf{A}^{(0)} & \mathsf{A}^{(n-1)} \\ \mathsf{A}^{(n-1)} & \cdots & \cdots & \mathsf{A}^{(2)} & \mathsf{A}^{(1)} & \mathsf{A}^{(0)} \end{pmatrix} . \tag{2.27}$$

The diagonalization of circulant matrices can be extended to block circulant matrices where the diagonalization is done by blocks and the unit matrix is the Kronecker product of the Fourier matrix with the identity. The following theorem describes this block diagonalization:

Theorem 2.2 (Gutiérrez Gutiérrez & Crespo (2012)). *Let $\mathbf{A}$ be an $n^2 \times n^2$ block circulant matrix defined by the sequence of blocks $\{\mathsf{A}^{(h)}\}_{h\in\mathcal{I}_n^+}$, then:*

$$\mathbf{A} = \frac{1}{n}(\mathbf{U}_n \otimes \mathbf{I}_n)^* \mathrm{bdiag}(\Psi^{(0)}, \cdots, \Psi^{(n-1)})(\mathbf{U}_n \otimes \mathbf{I}_n) \ , \tag{2.28}$$

where $\otimes$ is the Kronecker product, bdiag *is the block diagonal operator, $\mathbf{U}_n$ is the Fourier matrix of size $n \times n$ and $\Psi^{(0)}, \ldots, \Psi^{(n-1)}$ are blocks determined as follows:*

$$\begin{pmatrix} \Psi^{(0)} \\ \Psi^{(1)} \\ \vdots \\ \Psi^{(n-1)} \end{pmatrix} = (\mathbf{U}_n \otimes \mathbf{I}_n) \begin{pmatrix} \mathsf{A}^{(0)} \\ \mathsf{A}(1) \\ \vdots \\ \mathsf{A}^{(n-1)} \end{pmatrix} . \tag{2.29}$$

One can remark that when the blocks are of size 1×1, *i.e.*, scalars, this theorem coincides with Equation (2.6) of Theorem 2.1. Although interesting, this representation does not provide a closed form expression of the eigenvalues of the block circulant matrix. However, in the special case where the blocks are also circulant matrices – the matrix is called a doubly-block circulant matrix – then we can extend the diagonalization and get a closed form of the eigenvalues of doubly-block circulant matrices. First, we can remark that if the blocks $\mathsf{A}^{(0)}, \dots, \mathsf{A}^{(n-1)}$ are circulant matrices then the blocks $\boldsymbol{\Psi}^{(0)}, \dots, \boldsymbol{\Psi}^{(n-1)}$ are also circulant matrices because they are linear combinations of circulant matrices which are closed under the sum and products. Therefore, using Theorem 2.1 for each block of the block diagonal independently, we have:

$$\operatorname{bdiag}\Big(\boldsymbol{\Psi}^{(0)}, \cdots, \boldsymbol{\Psi}^{(n-1)}\Big) = \frac{1}{n}(\mathbf{I} \otimes \mathbf{U}_n)^* \boldsymbol{\Lambda} (\mathbf{I} \otimes \mathbf{U}_n) \ , \tag{2.30}$$

where $\boldsymbol{\Lambda} = \operatorname{diag}\Big((\mathbf{U}_n \psi^{(0)}, \dots, \mathbf{U}_n \psi^{(n-1)})\Big)$ and the vectors $\psi^{(0)}, \dots, \psi^{(n-1)}$ are the characteristic vectors of the circulant matrices $\boldsymbol{\Psi}^{(0)}, \dots, \boldsymbol{\Psi}^{(n-1)}$ respectively. By combining Equation (2.28) and Equation (2.30), we obtain the eigenvalues decomposition of a doubly-block circulant matrix. Given a doubly-block circulant matrix $\mathbf{A}$, we have:

$$\mathbf{A} = \frac{1}{n^2}(\mathbf{U}_n \otimes \mathbf{U}_n)^* \boldsymbol{\Lambda} (\mathbf{U}_n \otimes \mathbf{U}_n) \ . \tag{2.31}$$

This decomposition makes it possible to express the eigenvalues of a doubly-block circulant matrix with the characteristic vectors of the circulant matrices composing it. Furthermore, one can note that the eigenvectors are independent of the values of the matrix and can be expressed with the Fourier matrix.

Akin to circulant and block circulant matrices, we can extend the block structure to Toeplitz matrices. An $nm \times nm$ block Toeplitz matrix $\mathbf{B}$ is fully determined by a two-sided sequence of blocks $\{\mathsf{B}^{(j)}\}_{h \in \mathcal{I}_n}$ and where each block $\mathsf{B}^{(h)}$ is an $m \times m$ matrix. The block Toeplitz matrix $\mathbf{B} = \Big(\mathsf{B}^{(k-j)}\Big)_{j,k \in \mathcal{I}_n^+}$ is given by

$$\mathbf{B} = \begin{pmatrix} \mathsf{B}^{(0)} & \mathsf{B}^{(-1)} & \mathsf{B}^{(-2)} & \dots & \dots & \mathsf{B}^{(-n+1)} \\ \mathsf{B}^{(1)} & \mathsf{B}^{(0)} & \mathsf{B}^{(-1)} & \ddots & & \vdots \\ \mathsf{B}^{(2)} & \mathsf{B}^{(1)} & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \mathsf{B}^{(-1)} & \mathsf{B}^{(-2)} \\ \vdots & & \ddots & \mathsf{B}^{(1)} & \mathsf{B}^{(0)} & \mathsf{B}^{(-1)} \\ \mathsf{B}^{(n-1)} & \dots & \dots & \mathsf{B}^{(2)} & \mathsf{B}^{(1)} & \mathsf{B}^{(0)} \end{pmatrix} . \tag{2.32}$$

Block Toeplitz and doubly-block Toeplitz matrices (block Toeplitz matrix where the blocks are also Toeplitz) do not have a block diagonalization nor a closed-form expression for their eigenvalues. However, the Toeplitz operator defined in Equation (2.26) can be extended to block Toeplitz and doubly-block Toeplitz matrices. For block Toeplitz matrices, the trigonometric polynomial that *generates* the block Toeplitz matrix $\mathbf{B}$ can be defined as follows:

$$F_{\mathbf{B}}(\omega) \triangleq \sum_{h \in \mathcal{I}_n} \mathsf{B}^{(h)} e^{\mathbf{i}h\omega} \quad . \tag{2.33}$$

The function $F_{\mathbf{B}}$ is said to be the *generating function* of the block matrix $\mathbf{B}$. To recover the block Toeplitz matrix from its generating function, we use the Toeplitz operator defined in Equation (2.26); therefore by construction, we have $\mathbf{T}_n(F_{\mathbf{B}}) = \mathbf{B}$.

Relation with the Convolution Operator

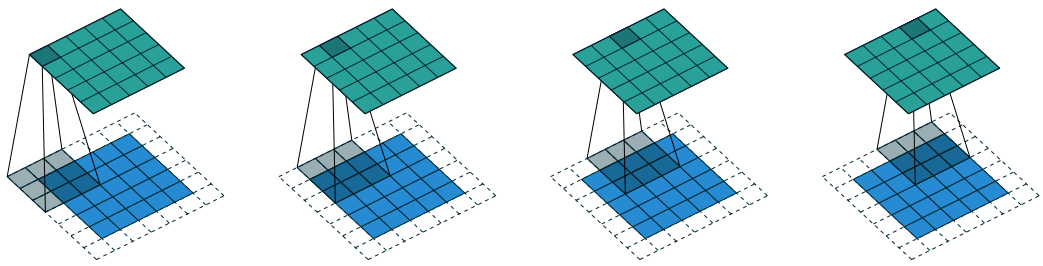


Figure 2.1: A convolution: a kernel sliding over an image and acting as a filter. Illustration taken from Dumoulin & Visin (2016).

Doubly-block circulant and doubly-block Toeplitz matrices are interesting structure due to their relation to 2-dimensional convolutions. We recall that a discrete convolution can be seen as a kernel sliding over the image and acting as a filter. Figure 2.1 illustrates the convolution operation with the image (blue), the kernel (gray) and the resulting operation (green). It has been shown by Jain (1989) that the result of a discrete 2d-convolution can be obtained by applying a doubly-block Toeplitz matrix to the input signal. A doubly-block Toeplitz matrix is a block Toeplitz matrix where the blocks are also Toeplitz. To illustrate, we consider a

discrete convolution between a 2-dimensional signal $\mathbf{x}$ and a kernel $\mathbf{K}$ where the kernel is defined as follows:

$$\mathbf{K} = \begin{pmatrix} k_0 & k_1 & k_2 \\ k_3 & k_4 & k_5 \\ k_6 & k_7 & k_8 \end{pmatrix}$$

then, the doubly-block Toeplitz matrix $\mathbf{M}$ that performs the convolution can be represented as:

$$\mathbf{M} = \begin{pmatrix} \mathsf{T}^{(0)} & \mathsf{T}^{(1)} & & 0 \\ \mathsf{T}^{(2)} & \mathsf{T}^{(0)} & \ddots & \\ & \ddots & \mathsf{T}^{(0)} & \mathsf{T}^{(1)} \\ 0 & & \mathsf{T}^{(2)} & \mathsf{T}^{(0)} \end{pmatrix} .$$

where $\mathsf{T}^{(j)}$ are Toeplitz matrices and the values of the kernel $\mathbf{K}$ are distributed in the Toeplitz blocks as follows:

$$\mathsf{T}^{(0)} = \begin{pmatrix} k_4 & k_3 & & & 0 \\ k_5 & k_4 & k_3 & & \\ & k_5 & \ddots & \ddots & \\ & & \ddots & k_4 & k_3 \\ 0 & & & k_5 & k_4 \end{pmatrix} \qquad \mathsf{T}^{(1)} = \begin{pmatrix} k_7 & k_6 & & & 0 \\ k_8 & k_7 & k_6 & & \\ & k_8 & \ddots & \ddots & \\ & & \ddots & k_7 & k_6 \\ 0 & & & k_8 & k_7 \end{pmatrix}$$

$$\mathsf{T}^{(2)} = \begin{pmatrix} k_1 & k_0 & & & 0 \\ k_2 & k_1 & k_0 & & \\ & k_2 & \ddots & \ddots & \\ & & \ddots & k_1 & k_0 \\ 0 & & & k_2 & k_1 \end{pmatrix}$$

In practice, the signal often can have multiple channels (*e.g.*, images have 3 channels corresponding to the colors red, green and blue). Let us denote c_{in} and c_{out} the number of the input and output channels respectively. Then the input signal has a dimension of $c_{in} \times n \times n$, performed by a kernel of size $c_{out} \times c_{in} \times k \times k$, and outputs a signal of size $c_{out} \times m \times m$ with $m = n - k + 2p + 1$ where p corresponds to the padding. The matrix for the multi-channel convolution is the concatenation of $c_{out} \cdot c_{in}$ doubly-block Toeplitz matrices.

$$\begin{pmatrix} \mathbf{h}_{n-1} & \cdots & \mathbf{h}_1 & \mathbf{h}_0 \\ \vdots & \ddots & \mathbf{h}_0 & \mathbf{h}_{-1} \\ \mathbf{h}_1 & \ddots & \ddots & \vdots \\ \mathbf{h}_0 & \mathbf{h}_{-1} & \cdots & \mathbf{h}_{-n+1} \end{pmatrix} \quad \begin{pmatrix} 1 & \mathbf{v}_0 & \cdots & \mathbf{v}_0^{n-1} \\ 1 & \mathbf{v}_1 & \cdots & \mathbf{v}_1^{n-1} \\ 1 & \vdots & & \vdots \\ 1 & \mathbf{v}_{n-1} & \cdots & \mathbf{v}_{n-1}^{n-1} \end{pmatrix} \quad \begin{pmatrix} \frac{1}{\mathbf{u}_0-\mathbf{v}_0} & \cdots & \frac{1}{\mathbf{u}_0-\mathbf{v}_{n-1}} \\ \frac{1}{\mathbf{u}_1-\mathbf{v}_0} & \cdots & \frac{1}{\mathbf{u}_1-\mathbf{v}_{n-1}} \\ \vdots & \cdots & \vdots \\ \frac{1}{\mathbf{u}_{n-1}-\mathbf{v}_0} & \cdots & \frac{1}{\mathbf{u}_{n-1}-\mathbf{v}_{n-1}} \end{pmatrix}$$

Hankel Vandermonde Cauchy

Figure 2.2: Representation of Hankel, Vandermonde and Cauchy matrices

2.1.4 LDR: General Framework for Structured Matrices

In this thesis, we mainly focus on structured transforms from the Toeplitz family. The properties of matrices from the Toeplitz family presented in the previous section make them good candidates for applications in the context of signal processing and deep neural network. However, other structured matrices with other properties have been considered. In this section, we briefly present these families of structured matrices and introduce LDR, a more general framework to capture all structured matrices. Below a description of some known structured matrices:

- **Hankel matrix**: A Hankel matrix has constant values along each of its anti-diagonals.
- **Vandermonde matrix**: A Vandermonde matrix is a matrix where each term follows a geometric progression. A very important special case is the complex matrix associated with the Discrete Fourier transform (DFT) presented in Equation (2.3) which has a Vandermonde structure.
- **Cauchy matrix**: A Cauchy matrix is an $m \times n$ matrix with elements a_{ij} such that $a_{ij} = (\mathbf{u}_i - \mathbf{v}_j)^{-1}$ with $\mathbf{u}_i - \mathbf{v}_j \neq 0$, $i \in \{0, \ldots, m-1\}$ and $j \in \{0, \ldots, n-1\}$.

Figure 2.2 shows the representation of the parameters sharing of Hankel, Vandermonde and Cauchy matrices. The study of these matrices with those from the Toeplitz family can be unified thought to the concept of *Low Displacement Rank* (LDR) initially proposed by Kailath et al. (1979). Although these matrices appear to have very different kinds of structure, they can be all associated with a specific displacement operator $\nabla_{\mathbf{A},\mathbf{B}} : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ which takes a matrix, $\mathbf{M}$, and outputs a low rank matrix $\nabla_{\mathbf{A},\mathbf{B}}(\mathbf{M})$ such that $\text{rank}(\nabla_{\mathbf{A},\mathbf{B}}(\mathbf{M})) \ll \min(m, n)$.

More formally, two displacement operators can be defined as follows (for simplification, we consider $m = n$):

Operator Matrices $\mathbf{A}$	$\mathbf{B}$	Class of structured matrices $\mathbf{M}$	Rank of $\nabla_{\mathbf{A},\mathbf{B}}(\mathbf{M})$
$\mathbf{Z}_1$	$\mathbf{Z}_{-1}$	Toeplitz	≤ 2
$\mathbf{Z}_1$	$\mathbf{Z}_0^\top$	Hankel	≤ 2
$\mathbf{Z}_0 + \mathbf{Z}_0^\top$	$\mathbf{Z}_0 + \mathbf{Z}_0^\top$	Toeplitz + Hankel	≤ 4
$\mathrm{diag}(\mathbf{v})$	$\mathbf{Z}_0$	Vandermonde	≤ 1
$\mathbf{Z}_0$	$\mathrm{diag}(\mathbf{v})$	Inverse of Vandermonde	≤ 1
$\mathrm{diag}(\mathbf{u})$	$\mathrm{diag}(\mathbf{v})$	Cauchy	≤ 1
$\mathrm{diag}(\mathbf{v})$	$\mathrm{diag}(\mathbf{u})$	Inverse of Cauchy	≤ 1

Table 2.1: Displacing Matrices Associated with Families of Structured Matrices.

Definition 2.3 (*Sylvester* & *Stein* displacement operators)**.** *Let* $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$*, the* Sylvester *displacement operator* $\nabla_{\mathbf{A},\mathbf{B}} : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ *is defined as follows:*

$$\nabla_{\mathbf{A},\mathbf{B}}(\mathbf{M}) \triangleq \mathbf{AM} - \mathbf{MB} \tag{2.34}$$

The Stein *displacement operator* $\triangle_{\mathbf{A},\mathbf{B}} : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ *is defined as follows:*

$$\triangle_{\mathbf{A},\mathbf{B}}(\mathbf{M}) \triangleq \mathbf{M} - \mathbf{AMB} \tag{2.35}$$

where $\nabla_{\mathbf{A},\mathbf{B}}(\mathbf{M}) = \mathbf{A}\,\triangle_{\mathbf{A}^{-1},\mathbf{B}}(\mathbf{M})$ *if the operator matrix* $\mathbf{A}$ *is non-singular, and* $\nabla_{\mathbf{A},\mathbf{B}}(\mathbf{M}) = -\triangle_{\mathbf{A},\mathbf{B}^{-1}}(\mathbf{M})\mathbf{B}$ *if the operator matrix* $\mathbf{B}$ *is non-singular.*

Based on this definition, if $\mathbf{M}$ is a structured matrix, there exist operator matrices $\mathbf{A}$ and $\mathbf{B}$ such that $\nabla_{\mathbf{A},\mathbf{B}}(\mathbf{M})$ is low rank. In particular, $\mathbf{A}$ and $\mathbf{B}$ can be chosen to be diagonal or f-unit-circulant matrices (see Definition 2.2) for several classes of structured matrices. Table 2.1 shows some specific choices of operators for the four classes of structured matrices presented above as well as other types of structured matrices from the same family. We now define the matrices that can be considered structured with respect to the Sylvester or Stein operator.

Definition 2.4 (L-like matrices Pan (2001))**.** *For an* $n \times n$ *matrix* $\mathbf{M}$ *and an associated operator* $\nabla_{\mathbf{A},\mathbf{B}}$ *(or* $\triangle_{\mathbf{A},\mathbf{B}}$*), the value* $r = \mathrm{rank}(\nabla_{\mathbf{A},\mathbf{B}}(\mathbf{M}))$ *(or* $r = \mathrm{rank}(\triangle_{\mathbf{A},\mathbf{B}}(\mathbf{M}))$*) is called the* displacement rank*. If the value of* r *is small relative to* n *as* n *grows large, then we call the matrix* $\mathbf{M}$ L-like *having a structure of type* L*. For example, in the case where the operator* $\nabla_{\mathbf{A},\mathbf{B}}$ *is associated with Toeplitz matrices (i.e.,* $\mathbf{A} = \mathbf{Z}_1$ *and* $\mathbf{B} = \mathbf{Z}_{-1}$*, see Table 2.1), we call the matrix* $\mathbf{M}$*,* Toeplitz-Like*.*

An important result allows us to express structured matrices with low-displacement rank directly as a function of their low displacement generators. This result can then be used to decompose structured matrices and define efficient algorithms for matrix-vector products.

Theorem 2.3 (Krylov Decomposition Pan & Wang (2003) and Sindhwani et al. (2015))**.** *If an $n \times n$ matrix $\mathbf{M}$ is such that $\triangle_{\mathbf{A},\mathbf{B}}(\mathbf{M}) = \mathbf{G}\mathbf{H}^\top$ where $\mathbf{G} = (\mathbf{g}^{(1)} \dots \mathbf{g}^{(r)}), \mathbf{H} = (\mathbf{h}^{(1)} \dots \mathbf{h}^{(r)}) \in \mathbb{R}^{n \times r}$ and the operator matrices satisfy: $\mathbf{A}^n = a\mathbf{I}$, $\mathbf{B}^n = b\mathbf{I}$ for some scalars a, b, then $\mathbf{M}$ can be expressed as:*

$$\mathbf{M} = \frac{1}{1-ab} \sum_{j=1}^{r} \text{krylov}(\mathbf{A}, \mathbf{g}^{(j)}) \ \text{krylov}(\mathbf{B}^\top, \mathbf{h}^{(j)})^\top \tag{2.36}$$

where $\text{krylov}(\mathbf{A}, \mathbf{v})$ *is defined by:*

$$\text{krylov}(\mathbf{A}, \mathbf{v}) = [\mathbf{v} \quad \mathbf{A}\mathbf{v} \quad \mathbf{A}^2\mathbf{v} \quad \dots \quad \mathbf{A}^{n-1}\mathbf{v}] \tag{2.37}$$

In the case of *Toeplitz-like matrices*, the above theorem can be simplified as follows:

Theorem 2.4 (Toeplitz-like matrix decomposition Pan (2001))**.** *If an $n \times n$ matrix $\mathbf{M}$ satisfies $\nabla_{\mathbf{Z}_1,\mathbf{Z}_{-1}}(\mathbf{M}) = \mathbf{G}\mathbf{H}^\top$ ($\mathbf{M}$ is Toeplitz-like) where $\mathbf{G} = (\mathbf{g}^{(1)} \dots \mathbf{g}^{(r)}), \mathbf{H} = (\mathbf{h}^{(1)} \dots \mathbf{h}^{(r)}) \in \mathbb{R}^{n \times r}$, then $\mathbf{M}$ can be written as:*

$$\mathbf{M} = \frac{1}{2} \sum_{j=1}^{r} \mathbf{Z}_1(\mathbf{g}^{(j)}) \mathbf{Z}_{-1}(\mathbf{J}_n \mathbf{h}^{(j)}) \tag{2.38}$$

where $\mathbf{J}_n$ is the reflection of the $n \times n$ identity matrix and $\mathbf{Z}_1$ and $\mathbf{Z}_{-1}$ are f-unit-circulant matrices (see Definition 2.2).

In the next chapter, we will see how this general framework have been used in the context of compact neural networks.

2.2 Supervised Learning and Neural Networks

2.2.1 Introduction to Supervised Learning

Supervised learning consists in learning a function that maps an input to an output based on input-output pairs. For example, one could learn to "predict" if a fruit will be tasty based on its features (*e.g.*, size, weight, color, consistency, etc.). These features

are used as inputs to the function and the function outputs a value characterizing the taste of the fruit.

In the following, we will formalize the learning problem described above with the *statistical learning framework*. First, let us define the domain space $\mathcal{X}$ which corresponds to the set of inputs that we wish to label. Let us denote the label space $\mathcal{Y}$ and a finite sequence of pairs $\mathcal{S} = \left\{\left(\mathbf{x}^{(1)}, y^{(1)}\right) \dots \left(\mathbf{x}^{(m)}, y^{(m)}\right)\right\}$ in $\mathcal{X} \times \mathcal{Y}$. Such pairs *i.e.*, labeled examples, are called *training examples* and the set $\mathcal{S}$ is called the *training set*. We denote $\mathcal{D}$ the *joint distribution* over $\mathcal{X} \times \mathcal{Y}$. The main objective of the task at hand is to output a function $h : \mathcal{X} \to \mathcal{Y}$ that maps the input $\mathbf{x} \in \mathcal{X}$ to the output $y \in \mathcal{Y}$. This function is called the *hypothesis* or the *classifier*. Given the probability distribution $\mathcal{D}$, we aim to measure how *likely* the hypothesis h makes an error when labeled points are randomly drawn from the distribution $\mathcal{D}$. Let us define the true error or *risk* of the hypothesis h that we wish to minimize:

$$R_{\mathcal{D}}(h) \triangleq \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\big[L\big(h(\mathbf{x}), y\big)\big] \ . \tag{2.39}$$

where $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ is a *loss function* which measures the correctness of the hypothesis. For example, for classification problems, we can use the 0-1 loss defined as:

$$L(h(\mathbf{x}), y) \triangleq \mathbb{1}_{\left[h(\mathbf{x})\neq y\right]} \tag{2.40}$$

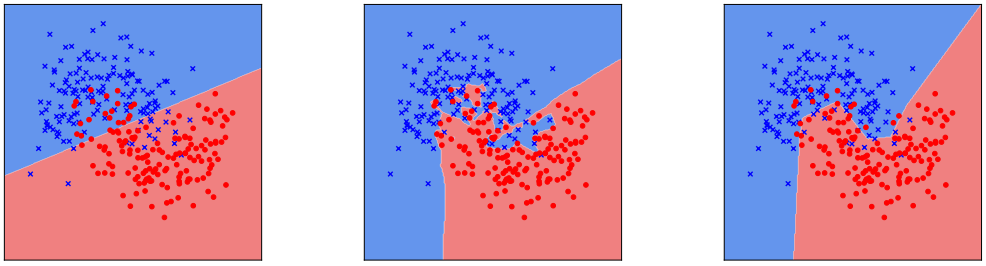
However, in practice, the joint probability distribution $\mathcal{D}$ is unknown; therefore, the true error is not directly available to the learner. The learner only has access to the training data, $\mathcal{S}$, and can calculate the *empirical error*, *i.e.*, the error over the training samples. We define the *empirical risk* as follows:

$$R_{\mathcal{S}}(h) \triangleq \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{x},y)\in\mathcal{S}} L\big(h(\mathbf{x}), y\big) \ . \tag{2.41}$$

The learning paradigm which consists in minimizing this value is called *Empirical Risk Minimization* denoted ERM.

We use the ERM paradigm as a surrogate to find a hypothesis h that minimizes the true risk $R_{\mathcal{D}}$. However, all hypotheses that minimize the empirical error do not necessarily minimize the true risk. For example, consider the following function:

$$h_c(\mathbf{x}) = \begin{cases} y^{(i)} & \text{if } \exists i \in [m] \text{ s.t. } \mathbf{x}^{(i)} = \mathbf{x} \\ c & \text{otherwise} \end{cases} \tag{2.42}$$



(a) Underfitting (b) Overfitting (c) Good fit

Figure 2.3: Decision boundary of three classifiers with different complexity for the same set of samples.

Clearly, this function, for any training, $\mathcal{S}$ and a binary target, will have $R_{\mathcal{S}}(h_1) = R_{\mathcal{S}}(h_2) = 0$, whereas one the two functions will have a true risk $\geq \frac{1}{2}$ (under the reasonable assumption that $\mathcal{S}$ is a negligible set with respect to $\mathcal{D}$). The phenomenon, called *overfitting*, happens when the classifier fits the training data "too well" but will likely have a high error on unseen data. One possible solution to this phenomenon is to apply ERM with a restricted search space to prevent the learning algorithm to output a function such as h_c in Equation (2.42). We call this set the *hypothesis class* and is denoted $\mathcal{H}$. Each $h \in \mathcal{H}$ is a function mapping from $\mathcal{X}$ to $\mathcal{Y}$. We call $\text{ERM}_{\mathcal{H}}$, the set of learned hypotheses that uses the ERM paradigm over the hypothesis class $\mathcal{H}$ and a training data $\mathcal{S}$. Formally,

$$\text{ERM}_{\mathcal{H}}(\mathcal{S}) = \arg\min_{h \in \mathcal{H}} R_{\mathcal{S}}(h) \ . \tag{2.43}$$

For a training sample $\mathcal{S}$, we denote $h_{\mathcal{S}} \in \text{ERM}_{\mathcal{H}}(\mathcal{S})$, one solution of applying $\text{ERM}_{\mathcal{H}}$ on the set $\mathcal{S}$, if there exist multiple hypotheses with minimal error on the training sample, then the minimization problem returns an arbitrary one. In practice, the hypothesis class is chosen on the basis of an assumption about the relationship between the data and its label. For example, if the relation between the data and its label is supposedly linear, then the hypothesis class can be the set of all linear functions. This kind of restriction is called the *inductive bias* because the learner is *biased* towards a particular set of predictors.

The ERM paradigm assumes that a hypothesis $h_{\mathcal{S}}$ that minimizes the risk $R_{\mathcal{S}}$ will also minimize the true risk $R_{\mathcal{D}}$. To verify that this assumption is correct, we need to ensure that all hypotheses in the hypothesis class $\mathcal{H}$ are good approximators of

their true risk. We say that a hypothesis class has the *uniform convergence property* if there exists a function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ such that for every $\epsilon, \delta \in (0,1)$, if $\mathcal{S}$ is a sample of size $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ examples drawn *independently and identically* according to $\mathcal{D}$, then, with probability of at least $1 - \delta$:

$$\forall h \in \mathcal{H}, \quad |R_{\mathcal{D}}(h) - R_{\mathcal{S}}(h)| \leq \epsilon \ , \tag{2.44}$$

where the function $m_{\mathcal{H}}$, called *the sample complexity*, measures the minimal number of examples needed to ensure that with probability of at least $1 - \delta$, Equation (2.44) holds. The i.i.d. assumption is common in statistical learning theory. It is easy to see that the error between $R_{\mathcal{S}}(h)$ and $R_{\mathcal{D}}(h)$ is dependent on the representativeness of the sample $\mathcal{S}$ with respect to the underlying distribution $\mathcal{D}$. Therefore, the parameter δ characterizes the probability of having a nonrepresentative sample. The quantity $1 - \delta$ is the *confidence parameter* of the prediction.

The following result gives the sample complexity measure the maximum value for which the hypothesis class has the *uniform convergence property.*

Theorem 2.5 (Shalev-Shwartz & Ben-David (2014)). *Let $\mathcal{H}$ be a* finite *hypothesis class, then $\mathcal{H}$ enjoys the uniform convergence property with sample complexity:*

$$m_{\mathcal{H}}(\epsilon, \delta) \leq \left\lceil \frac{\log(2|\mathcal{H}|/\delta)}{2\epsilon^2} \right\rceil \ , \tag{2.45}$$

where $|\mathcal{H}|$ is the cardinal of the set $\mathcal{H}$. Note that we can also write the bound from Theorem 2.5 as follows: for all $h \in \mathcal{H}$, we have

$$\underbrace{R_{\mathcal{D}}(h)}_{\text{Error}} \leq \underbrace{R_{\mathcal{S}}(h)}_{\text{Estimation Error}} + \underbrace{\sqrt{\frac{\log(|\mathcal{H}|/\delta)}{2m_{\mathcal{H}}(\epsilon, \delta)}}}_{\text{Complexity penality}} \ , \tag{2.46}$$

with probability $1 - \delta$. This bound is called a *generalization bound* and consists in bounding the true error by the empirical error and a complexity penalty. In the above theorem, the condition on the finiteness of the hypothesis class might by too strong. For example, if we consider the set of linear functions parameterized by a set of real-valued parameters the hypothesis class is infinite and the theorem above does not apply. To characterize the learnability of infinite hypothesis classes, several complexity measures have been proposed. One of the first, discovered by Vapnik & Chervonenkis (2015), relies on a combinatorial notion called the Vapnik-Chervonenkis dimension (VC-dimension). They showed that having a finite VC-dimension is a

necessary and sufficient condition for the uniform convergence property. In the same vein, the Rademacher complexity (Koltchinskii & Panchenko, 2000), measures the richness of the class of real-valued functions with respect to a probability distribution. In Section 2.2.4, we will study recent generalization bounds specific to neural networks where the complexity penalty is dependent on the Lipschitz constant of the weights matrices.

A fundamental question of the ERM paradigm remains: *how to choose the correct hypothesis class for which $ERM_{\mathcal{H}}$ will not lead to overfitting?* We answer this question by decomposing the true risk into two different components as follows:

$$R_{\mathcal{D}}(h_{\mathcal{S}}) = \underbrace{\left[\min_{h \in \mathcal{H}} R_{\mathcal{D}}(h)\right]}_{\text{Approximation Error}} + \underbrace{\left[R_{\mathcal{D}}(h_{\mathcal{S}}) - \min_{h \in \mathcal{H}} R_{\mathcal{D}}(h)\right]}_{\text{Estimation Error}} \tag{2.47}$$

- **Approximation Error**: The approximation error corresponds to the minimum risk achievable by a classifier in the given hypothesis class. Intuitively, this error measures the quality of the hypothesis class and therefore the quality of the prior knowledge. Enlarging the hypothesis class, *i.e.*, allowing more complex functions, can decrease the approximation error.

- **Estimation Error**: The estimation error is the difference between the approximation error and the error made by the ERM predictor. Recall that the empirical risk is only an estimate of the true risk. This error is dependent on the sample size and the complexity of the hypothesis class.

Recall that the main goal is to minimize the true risk $R_{\mathcal{D}}(h_{\mathcal{S}})$, however, Equation (2.47) shows a trade-off called the *bias-complexity trade-off*. The trade-off is as follows: if we choose a large and complex hypothesis space, we reduce the approximation error but at the same time we can increase the estimation error because a complex hypothesis space might lead to overfitting. Conversely, choosing a small hypothesis space might reduce the estimation error but increase the approximation error leading to *underfitting*. We can illustrate the *overfitting* and *underfitting* phenomenons with Figure 2.3 which shows the decision boundary of 3 classifiers for the same set of samples. Figure 2.3a shows a classifier which *underfits* the data, meaning the decision boundary is not complex enough to separate the data correctly. Figure 2.3b shows a classifier that almost perfectly fits the training data but is likely to have a higher error rate on the unseen data. Finally, Figure 2.3c shows a classifier that seems to have a good compromise between the two.

As seen above, defining a small hypothesis class might lead to underfitting and a large hypothesis class might lead to overfitting. A good way to balance the trade-off would be to minimize the empirical risk while also minimizing the complexity of the hypothesis class. Let us define a *regularization* function $r : \mathcal{H} \to \mathbb{R}$ which takes a hypothesis as input and measures the "complexity" of the hypothesis. We could now update the learning rule as follows:

$$\arg\min_{h \in \mathcal{H}} [R_{\mathcal{S}}(h) + r(h)] \tag{2.48}$$

This learning rule minimizes the empirical risk $R_{\mathcal{S}}(h)$ and a well chosen regularization function r. If $r(\,\cdot\,)$ is carefully chosen, this prevent overfitting and improve generalization on unseen data. This learning rule is closely related to *Structural Minimization Paradigm* (SRM) (Shalev-Shwartz & Ben-David, 2014). In the next section, we will present a classical regularization function for neural networks and we will introduce a new regularization scheme in Chapter 5.

2.2.2 Preliminaries on Neural Networks

Neural networks, which find their roots in the work of McCulloch & Pitts (1943) and Rosenblatt (1958), can be analytically described as a composition of linear functions interlaced with nonlinear functions (also called activation functions). A feedforward neural network can be defined as follows:

Definition 2.5 (Neural Network). *Given a depth $p \in \mathbb{N}$, let $w = \{w^{(i)}\}_{i \in [p+1]}$ and $b = \{b^{(i)}\}_{i \in [p]}$ be sequences of integers, $\Omega = \left\{\left(\mathbf{W}^{(i)}, \mathbf{b}^{(i)}\right)\right\}_{i \in [p]}$ a set of weights matrices and bias vectors such that $\mathbf{W}^{(i)} \in \mathbb{R}^{w^{(i)} \times w^{(i+1)}}$ and $\mathbf{b}^{(i)} \in \mathbb{R}^{b^{(i)}}$ and a sequence of activation functions $\rho = \{\rho_i\}_{i \in [p]}$. Let $\mathcal{X} \subset \mathbb{R}^{w^{(1)}}$ and $\mathcal{Y} \subset \mathbb{R}^{w^{(p+1)}}$ be the input and output spaces respectively. $w^{(1)}$ and $w^{(p)}$ refer to the input and output dimension respectively. A neural network is a function $N_\Omega^\rho : \mathcal{X} \to \mathcal{Y}$ such that*

$$N_\Omega^\rho(\mathbf{x}) \triangleq \phi^{\rho_p}_{\mathbf{W}^{(p)}, \mathbf{b}^{(p)}} \circ \cdots \circ \phi^{\rho_1}_{\mathbf{W}^{(1)}, \mathbf{b}^{(1)}}(\mathbf{x}) \tag{2.49}$$

where $\phi^{\rho_i}_{\mathbf{W}^{(i)}, \mathbf{b}^{(i)}} : \mathbb{R}^{w^{(i)}} \to \mathbb{R}^{w^{(i+1)}}$ (also called layer) is a function parameterized by the weight matrix $\mathbf{W}^{(i)}$, the bias vector $\mathbf{b}^{(i)}$ and the activation function ρ_i. $\phi^{\rho_i}_{\mathbf{W}^{(i)}, \mathbf{b}^{(i)}}$: is defined as follows:

$$\phi^{\rho_i}_{\mathbf{W}^{(i)}, \mathbf{b}^{(i)}}(\mathbf{x}) \triangleq \rho_i\left(\mathbf{W}^{(i)}\mathbf{x} + \mathbf{b}^{(i)}\right) \ , \tag{2.50}$$

and ρ_p is identity function.

Based on this definition, for a given training set $\mathcal{S} \subset \mathcal{X} \times \mathcal{Y}$, a set of activation functions ρ, a set of weights and biases Ω and a loss function $L : \mathcal{Y} \times [k] \to \mathbb{R}_+$, the ERM learning paradigm for neural networks is given by

$$\arg\min_{\Omega} \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{x},y)\in\mathcal{S}} L(N_\Omega^\rho(\mathbf{x}), y) \tag{2.51}$$

For classification problems, the zero-one loss is non-convex, and finding a near optimal solution is an NP-hard problem (Ben-David et al. 2003; Feldman et al. 2012). Instead, a common approach is to use a surrogate loss such as the logistic loss multiclass function and estimate the parameters by maximizing the *likelihood* over the data. This loss $L : \mathcal{Y} \times [k]$, is defined as follows:

$$L(N_\Omega^\rho(\mathbf{x}), y) = -\log\left(\frac{e^{\left(N_\Omega^\rho(\mathbf{x})\right)_y}}{\sum_{j\in[k]} e^{\left(N_\Omega^\rho(\mathbf{x})\right)_j}}\right) \tag{2.52}$$

The generic approach for minimizing the empirical risk in Equation (2.51) is by *gradient descent* with the *backpropagation* algorithm (Rumelhart et al. 1986) which consists in computing the gradient with the chain rule.

As seen in the previous section, the SRM paradigm minimizes two terms, the empirical risk and a weight function measuring the "complexity" of the hypothesis. It has been shown that the ℓ_2 norm of the weights of a network can be used as a measure of complexity of the network (Hinton, 1987). This *regularization*, also called *weight decay*, prevents weights from growing too large. The SRM learning algorithm can then be expressed as follows:

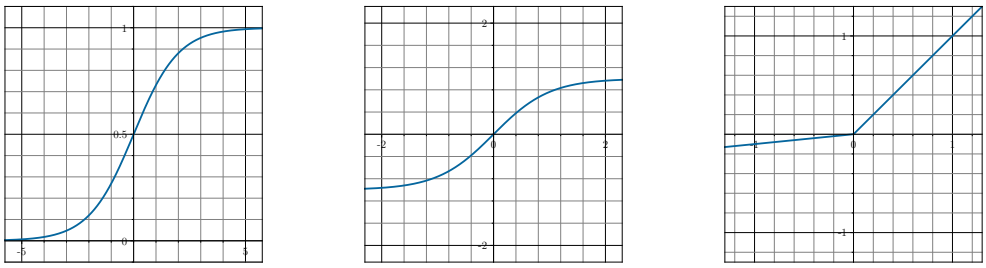
$$\arg\min_{\Omega} \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{x},y)\in\mathcal{S}} L(N_\Omega^\rho(\mathbf{x}), y) + \lambda \sum_{(\mathbf{W},\mathbf{b})\in\Omega} (\|\mathbf{W}\|_{\mathrm{F}} + \|\mathbf{b}\|_2) \tag{2.53}$$

where $\lambda > 0$ is the regularization parameter.

Choosing the right activation function has been an active area of research. Hereafter, we present three common activation functions used by practitioners.

- **Sigmoid activation**

$$\rho(x) = \frac{1}{1+e^{-x}}$$



(a) Sigmoid Activation (b) Tanh Activation (c) Leaky-ReLU Activation

Figure 2.4: Graphical representation of three common activation functions

The sigmoid activation function is one of the first continuous nonlinear functions to be used in the context of neural networks. It takes a real value as input and outputs another value between 0 and 1.

- **Hyperbolic Tangent activation**

$$\rho(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

The hyperbolic tangent activation function is similar to the sigmoid activation function but instead of returning between 0 and 1, the function returns values between -1 and 1.

- **Rectified Linear activation (ReLU)** (Nair & Hinton, 2010)

$$\rho(x) = \max(0, x)$$

The ReLU activation was proposed to avoid the *vanishing gradient problem*. The vanishing gradient problem, discovered by Bengio et al. (1994), occurs with hyperbolic tangent or sigmoid activation when the magnitude of the input values are almost saturated at -1 or 1, in this case the gradient is close to 0 and difficulties of optimization and convergence occur. The ReLU activation addresses this problem due to the simple values of its gradients which are either 0 or 1 on $\mathbb{R}_-$ or $\mathbb{R}_+$ respectively. Furthermore, it has the advantage to be less computationally expensive than tanh and sigmoid functions because it involves simpler mathematical operations.

- **Leaky Rectified Linear activation (Leaky-ReLU)** (Maas et al. 2013)

$$\rho(x) = \max(\alpha, x)$$

More recently, the Leaky-ReLU ($\alpha > 0$) activation function was proposed. It introduces the parameter α which characterizes the slope on $\mathbb{R}_-$. The advantage of Leaky-ReLU over the ReLU nonlinear activation is that it prevents sparse gradient, which facilitates convergence of neural networks.

Figure 2.4 presents the graphical representation of the activation functions presented above. In this thesis, we will use the Leaky-ReLU function with different α when we train deep neural networks. We simplify the notation N_Ω^ρ with N_Ω.

2.2.3 Adversarial Attacks & Robustness of Neural Networks

As seen in the introduction (Chapter 1), deep neural networks achieve state-of-the-art performances in a variety of domains such as natural language processing (Radford et al. 2019), image recognition (He et al. 2016) and speech recognition (Hinton et al. 2012). However, it has been shown that such neural networks are vulnerable to *adversarial examples*, *i.e.*, imperceptible variations of the natural examples, crafted to deliberately mislead the models (Globerson & Roweis, 2006; Biggio et al. 2013; Szegedy et al. 2014). Because it is difficult to characterize the space of visually imperceptible variations of a natural image, existing adversarial attacks use ℓ_p norms as surrogate measures. We can formally define an adversarial example as follows:

Definition 2.6 (Adversarial Pertubation)**.** *Given an example* $\mathbf{x}$ *and its predicted label* y*,* k *number of classes, a trained neural network* N_Ω *with* $\arg\max_{i\in[k]} \left(N_\Omega(\mathbf{x})\right)_i = y$ *and a radius* $\epsilon \in \mathbb{R}$*, an adversarial perturbation is a vector* $\boldsymbol{\tau} \in \mathcal{X}$ *such that:*

$$\begin{aligned} &\arg\max_{i\in[k]} \left(N_\Omega(\mathbf{x}+\boldsymbol{\tau})\right)_i \neq y \ , \\ &s.t.\ \|\boldsymbol{\tau}\|_p \leq \epsilon \end{aligned} \tag{2.54}$$

where ϵ *is a small value defined by the attacker.*

Note that this definition assumes that the attacker (the person crafting the attack) has access to the parameters of the model. Typically, an attack method is either *white-box* (complete knowledge of the model and its parameters) or *black-box* (no knowledge of the model). It is possible to consider that the white-box setting admits too strong assumptions because a model and its parameters could very well be hidden

from the public. In general, it is safer to assume that the adversary has complete knowledge of the model and its defense. This principle is known in the field of security as the Shannon's maxim (Shannon, 1949). Therefore, in this thesis, we only consider defenses against white-box attacks.

Implementing Adversarial Attacks

Since the discovery of adversarial perturbations, a variety of procedures, *a.k.a. adversarial attacks*, have been developed to generate adversarial examples. FGSM (Goodfellow et al. 2015), PGD (Madry et al. 2018) and (Carlini & Wagner, 2017) to name a few, are the most popular ones. To find the best perturbation $\boldsymbol{\tau}$, existing attacks can adopt one of the two following strategies:

Loss maximization. In this scenario, the procedure maximizes the loss objective function $L(N_\Omega(\mathbf{x}+\boldsymbol{\tau}), y)$, under the constraint that the ℓ_p norm of the perturbation remains bounded by some value ϵ, as follows:

$$\underset{\boldsymbol{\tau}:\|\boldsymbol{\tau}\|_p \leq \epsilon}{\arg\max}\, L(N_\Omega(\mathbf{x}+\boldsymbol{\tau}), y) \quad . \tag{2.55}$$

The typical value of ϵ depends on the norm $\|\cdot\|_p$ considered in the problem setting. The current state-of-the-art method to solve Equation (2.55) is based on a projected gradient descent (PGD) (Madry et al. 2018) of radius ϵ. Given a budget ϵ, it recursively computes

$$\mathbf{x}^{(t+1)} = \prod_{\mathcal{B}_p(\mathbf{x},\epsilon)} \left(\mathbf{x}^{(t)} + \alpha \underset{\boldsymbol{\tau}:\|\boldsymbol{\tau}\|_p \leq 1}{\arg\max}\, \boldsymbol{\tau}^\top \nabla_{\mathbf{x}^{(t)}} L\Big(N_\Omega(\mathbf{x}^{(t)}), y\Big) \right) \tag{2.56}$$

where $\mathcal{B}_p(\mathbf{x},\epsilon) = \{\mathbf{x}+\boldsymbol{\tau} : \|\boldsymbol{\tau}\|_p \leq \epsilon\}$ is the ball of norm p with radius ϵ, centered at $\mathbf{x}$, α is a gradient step size, and $\prod_{\mathcal{B}}$ is the projection operator on the ball $\mathcal{B}$. The PGD attack is currently used in the literature with $p=2$ and $p=\infty$. The attack with the norm $p=\infty$ is state-of-the-art for the loss maximization problem.

Perturbation minimization. This type of procedure searches for the perturbation with the minimal ℓ_p norm, under the constraint that $L(N_\Omega(\mathbf{x}+\boldsymbol{\tau}), y)$ is bigger than a given bound c:

$$\begin{aligned} &\arg\min_{\boldsymbol{\tau}} \|\boldsymbol{\tau}\|_p \ , \\ &s.t.\ L(N_\Omega(\mathbf{x}+\boldsymbol{\tau}), y) \geq c \end{aligned} \tag{2.57}$$

The value of c is typically chosen depending on the loss function L. For example, if L is the 0-1 loss, any $c > 0$ is acceptable. Equation (2.57) has been tackled by Carlini & Wagner (2017), leading to the following method, denoted C&W attack in the rest of the chapter. It aims at solving the following Lagrangian relaxation of Equation (2.57):

$$\arg\min_{\boldsymbol{\tau}} \|\boldsymbol{\tau}\|_p + \lambda g(\mathbf{x}+\boldsymbol{\tau}) \ , \tag{2.58}$$

where $g(\mathbf{x}+\boldsymbol{\tau}) < 0$ if and only if $L(N_\Omega(\mathbf{x}+\boldsymbol{\tau}), y) \geq c$. The authors use a binary search to optimize the constant c, and gradient descent to compute an approximated solution. The C&W attack is currently used in the literature with $p \in \{1, 2, \infty\}$ and is state-of-the-art with $p = 2$ for the perturbation minimization problem if we consider that the attacker has unlimited computing power.

Defending against Adversarial Attacks

Given the security risks that adversarial attacks pose, it is important to design defenses to protect neural networks against these kinds of attacks. Adversarial Training was introduced by Goodfellow et al. (2015) and later improved by Madry et al. (2018) as a first defense mechanism to train robust neural networks. It consists in augmenting training batches with adversarial examples generated during the training procedure. The structural risk minimization paradigm is thus replaced by the following min max problem, where the classifier tries to minimize the expected loss under the maximum perturbation of its input:

$$\min_{\Omega} \max_{\boldsymbol{\tau}:\|\boldsymbol{\tau}\|\leq\epsilon} \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{x},y)\in\mathcal{S}} L(N_\Omega(\mathbf{x}+\boldsymbol{\tau}), y) + \lambda \sum_{(\mathbf{W},\mathbf{b})\in\Omega} (\|\mathbf{W}\|_{\mathrm{F}} + \|\mathbf{b}\|_2) \ . \tag{2.59}$$

Although adversarial training lacks formal guarantees, it is one of the few techniques that proves to be empirically very effective.

2.2.4 Recent Results on the Theory of Neural Networks

In this section, we give recent generalization bounds for neural networks. Neural networks have the astonishing property of providing a low error rate on unseen data although they have more parameters than the number of training samples and therefore have the capabilities to fit random labels (Zhang et al. 2017). In this context, traditional approaches of statistical learning fail to explain why large neural networks generalize well in practice.

Harvey et al. (2017) have introduced a generalization bound of neural networks with the VC-dimension as a complexity measure of the hypothesis class. They improved over previous bounds (Bartlett et al. 1998; Anthony & Bartlett, 1999) by showing that the VC-dimension of a p-layer feedforward neural network is equal to the depth times the number of parameters. Unfortunately, this kind of bound with such a complexity measure is of little help to better understand the generalization capabilities of neural networks.

More recently, Bartlett et al. (2017) have proposed to use a *scale-sensitive* complexity measure instead of combinatorial ones (*i.e.*, VC-dimension) which can work with real-valued function classes and are sensitive to their magnitudes. They proposed to use the product of the spectral norms of the weight matrices (*i.e.*, the Lipschitz constant of the weight matrices) of the network to this scale-sensitive complexity measure. In addition, they investigated the margins and show that normalizing these Lipschitz constants by the margin allows to better control their excess risk (the test error minus the training error) across training epochs. The margin has been previously studied in relationship to generalization by Langford & Shawe-Taylor (2002) and more recently by Neyshabur et al. (2018).

In what follows, we present generalization bounds for neural networks which are independent of the number of parameters of the network and use as a complexity penalty the Lipschitz constant of the weight matrices. In addition, following the Section 2.2.3 on Adversarial Attacks, we present the work of Farnia et al. (2019) which introduce *adversarial risk* and *empirical adversarial risk* and present an *adversarial generalization bound* similar to the one proposed by Bartlett et al. (2017).

First, let us formally define the Lipschitz constant of a function as well as the spectral norm of a matrix. We will use these notions in the following and later in the thesis. Formally, the Lipschitz constant of a function is defined as follows:

Definition 2.7 (Lipschitz Constant). *The Lipschitz constant with respect to the ℓ_p-norm of a Lipschitz continuous function $f : \mathbb{R}^n \to \mathbb{R}^m$ is defined as follows:*

$$\mathrm{Lip}_p(f) \triangleq \sup_{\substack{\mathbf{x},\mathbf{y}\in\mathbb{R}^n \\ \mathbf{x}\neq\mathbf{y}}} \frac{\|f(\mathbf{x}) - f(\mathbf{y})\|_p}{\|\mathbf{x} - \mathbf{y}\|_p} \ . \tag{2.60}$$

In the following of this thesis, we denote $\mathrm{Lip}_2(f)$ by $\mathrm{Lip}(f)$ for simplicity and if $\mathrm{Lip}_p(f) = k$, we denote the function f as k-Lipschitz. The spectral norm of a matrix $\mathbf{W}$, which is equivalent to the Lipschitz constant of the function $\mathbf{x} \mapsto \mathbf{W}\mathbf{x}$, is defined as follows:

Definition 2.8 (Spectral norm). *Given a matrix $\mathbf{W}$, the spectral norm of $\mathbf{W}$ denoted $\|\mathbf{W}\|_2$ is defined as:*

$$\|\mathbf{W}\|_2 \triangleq \sup_{\substack{\mathbf{x}\in\mathbb{R}^n \\ \mathbf{x}\neq\mathbf{0}_n}} \frac{\|\mathbf{W}\mathbf{x}\|_2}{\|\mathbf{x}\|_2} \ . \tag{2.61}$$

Note that the spectral norm also corresponds to the largest singular value of the matrix denoted $\sigma_1(\mathbf{W})$.

Before presenting the bound from Bartlett et al. (2017), let us introduce and recall some notations. Let N_Ω be a neural network parameterized by Ω as in the Definition 2.5. Let us recall the risk with respect to the neural network N_Ω and a distribution $\mathcal{D}$ as in Equation (2.39):

$$R_{\mathcal{D}}(N_\Omega) = \mathbb{P}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\arg\max_{i\in[k]}(N_\Omega(\mathbf{x}))_i \neq y\right] \ . \tag{2.62}$$

Bartlett et al. (2017) extended the notion of risk with a *margin operator* $M : \mathbb{R}^k \times [k] \to \mathbb{R}$ defined as $M(\mathbf{v}, j) \triangleq \mathbf{v}_j - \max_{i\neq j} \mathbf{v}_i$ and an extension to the 0-1 loss called the *ramp loss* $L_\gamma : \mathbb{R} \to \mathbb{R}_+$ as:

$$L_\gamma(r) \triangleq \begin{cases} 0 & r < -\gamma, \\ 1 + r/\gamma & r \in [-\gamma, 0], \\ 1 & r > 0, \end{cases} \tag{2.63}$$

Now, we can define the *margin risk* as

$$R_{\gamma,\mathcal{D}}(N_\Omega) \triangleq \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\big[L_\gamma\big(-M(N_\Omega(\mathbf{x}), y)\big)\big] \ , \tag{2.64}$$

and the *empirical margin risk* as

$$R_{\gamma,\mathcal{S}}(N_\Omega) \triangleq \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{x},y)\in\mathcal{S}} L_\gamma\big(-M(N_\Omega(\mathbf{x}),y)\big) \ . \tag{2.65}$$

Note that the margin risk and the empirical margin risk upper bound the risk and empirical risk. The generalization bound proposed by Bartlett et al. (2017) for neural networks is stated as follows:

Theorem 2.6 (Bartlett et al. (2017)). *Let $(\rho^{(1)},\dots,\rho^{(p)})$ be nonlinearities where $\forall i \in [p], \mathrm{Lip}\big(\rho^{(i)}\big) < \infty$ and $\rho^{(i)}(0) = 0$. Let $w^{(1)},\dots,w^{(p+1)}$ be integers such that $\mathbf{W}^{(i)} \in \mathbb{R}^{w^{(i)}\times w^{(i+1)}}$ and let $W = \max_i w^{(i)}$. Let $\mathbf{X}$ a matrix where the rows of $\mathbf{X}$ are the input data $\mathbf{x}^{(1)},\dots,\mathbf{x}^{(m)} \in \mathcal{S}$. Let $N_\Omega : \mathbb{R}^{w^{(1)}} \to \mathbb{R}^{w^{(p+1)}}$ be a neural network parameterized by Ω as in the Definition 2.5 where $\big(\mathbf{W}^{(1)},\dots,\mathbf{W}^{(p)}\big)$ are the weights matrices. Then for every margin $\gamma > 0$, the following bound applies:*

$$R_{\mathcal{D}}(N_\Omega) \le R_{\gamma,\mathcal{S}}(N_\Omega) + \widetilde{\mathcal{O}}\left(\frac{\|\mathbf{X}\|_{\mathrm{F}}\mathcal{R}_\Omega}{\gamma|\mathcal{S}|}\ln(W) + \sqrt{\frac{\ln(1/\delta)}{|\mathcal{S}|}}\right) \tag{2.66}$$

with probability at least $1-\delta$, where the spectral complexity *$\mathcal{R}_\Omega$ is defined as*

$$\mathcal{R}_\Omega = \left(\prod_{i=1}^{p} \mathrm{Lip}\big(\rho^{(i)}\big)\big\|\mathbf{W}^{(i)}\big\|_2\right)\left(\sum_{i=1}^{p}\frac{\big\|\mathbf{W}^{(i)\top}\big\|_{2,1}^{2/3}}{\|\mathbf{W}^{(i)}\|_2^{2/3}}\right)^{3/2} \tag{2.67}$$

where $\widetilde{\mathcal{O}}(\,\cdot\,)$ ignores logarithmic factors and the norm $\|\,\cdot\,\|_{p,q}$ is defined by $\|\mathbf{W}\|_{p,q} \triangleq \left(\sum_{j=1}^{n}(\sum_{i=1}^{m}|a_{ij}|^p)^{\frac{q}{p}}\right)^{\frac{1}{q}}$.

As we stated in the introduction, the generalization of neural networks is important but should not be the only metric to consider. Indeed, a neural network that performs well on natural data could be vulnerable to adversarial attacks. We saw in the previous section that *adversarial training* is a technique that successfully improves the robustness of neural networks by learning on adversarial examples instead of natural ones. In the following, we present recent results devised by Farnia et al. (2019) on the generalization capabilities of neural networks trained with adversarial training. First, let us define the *adversarial margin risk* as:

$$R^{\mathrm{adv}}_{\gamma,\mathcal{D}}(N_\Omega) \triangleq \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\ L_\gamma\big(-M(N_\Omega(\mathbf{x}+\boldsymbol{\tau}^{\mathrm{adv}}_\Omega(\mathbf{x})),y)\big) \ , \tag{2.68}$$

where $\boldsymbol{\tau}_{\Omega}^{\text{adv}}(\mathbf{x})$ is an adversarial perturbation following the loss maximization strategy presented above. The *adversarial empirical margin risk* $R_{\gamma,\mathcal{S}}^{\text{adv}}$ is defined similarly as in Equation (2.65). The adversarial generalization bound is stated as follows:

Theorem 2.7 (Farnia et al. (2019)). *Let $(\rho^{(1)}, \ldots, \rho^{(p)})$ be nonlinearities where $\forall i \in [p], \text{Lip}\left(\rho^{(i)}\right) = 1$ and $\rho^{(i)}(0) = 0$. Let $b = \max_{\mathbf{x}\in\mathcal{S}} \|\mathbf{x}\|_2$. Let $\mathcal{X} \subset \mathbb{R}^{w^{(1)}}$ and $\mathcal{Y} \subset \mathbb{R}^{w^{(p+1)}}$ be the input and output spaces respectively. Let $N_\Omega : \mathcal{X} \to \mathcal{Y}$ be a neural network parameterized by Ω of depth p and of largest width $W = \max_i w^{(i)}$ following the Definition 2.5. Assume that for a constant $c_1 \geq 1$ the weights matrices satisfy:*

$$\forall i, \quad \frac{1}{c_1} \leq \frac{\left\|\mathbf{W}^{(i)}\right\|_2}{\prod_{j=1}^{p}(\|\mathbf{W}^{(j)}\|_2)^{1/p}} \leq c_1 \tag{2.69}$$

and that $c_2 \leq \|\nabla_{\mathbf{x}} L(N_\Omega(\mathbf{x}), y)\|_2$ holds for a constant $c_2 > 0$, any $y \in \mathcal{Y}$ and any $\mathbf{x} \in \mathcal{B}_2(\mathbf{x}, \epsilon)$. Let us consider an attack such that $\left\|\boldsymbol{\tau}_{\Omega}^{\text{adv}}(\mathbf{x})\right\|_2 \leq \epsilon$ with r iterations, and a stepsize α. Then for every margin $\gamma > 0$, the following bound applies:

$$R_{0,\mathcal{D}}^{\text{adv}}(N_\Omega) \leq R_{\gamma,\mathcal{S}}^{\text{adv}}(N_\Omega) + \mathcal{O}\left(\sqrt{\frac{(b+\epsilon)^2 p^2 W \log(pW)\mathcal{R}_{N_\Omega}^{\text{adv}} + p\log\left(\frac{rp|\mathcal{S}|\log(M)}{\delta}\right)}{\gamma^2|\mathcal{S}|}}\right) \tag{2.70}$$

with probability at least $1-\delta$, where the adversarial spectral complexity $\mathcal{R}_{\Omega}^{\text{adv}}$ *is defined as*

$$\mathcal{R}_{N_\Omega}^{\text{adv}} \triangleq \left[\prod_{i=1}^{p}\left\|\mathbf{W}^{(i)}\right\|_2\left(1 + (\alpha/c_2)\frac{1-(2\alpha/c_2)^r\Phi_\Omega^r}{1-(2\alpha/c_2)\Phi_\Omega}\Phi_\Omega\right)\right]^2 \sum_{i=1}^{p}\frac{\left\|\mathbf{W}^{(i)}\right\|_{\text{F}}^2}{\|\mathbf{W}^{(i)}\|_2^2} \tag{2.71}$$

and where Φ_Ω is defined as: $\Phi_\Omega \triangleq \left(\prod_{i=1}^{p}\left\|\mathbf{W}^{(i)}\right\|_2\right)\sum_{i=1}^{p}\prod_{j=1}^{i}\left\|\mathbf{W}^{(j)}\right\|_2$

These generalization bounds for neural networks give us a theoretical justification for *Lipschitz regularization*. However, computing the spectral norm of the weights matrices is a difficult task. In the next chapter, we will review some known techniques and we propose in Chapter 5 new efficient method for regularizing the Lipschitz constant of neural networks.

2.3 Summary of the Chapter

As explained in the Introduction (Chapter 1), our contributions lie at the intersection between neural networks and structured matrices. In this chapter, we have reviewed the necessary concepts to present our contributions and some related works.

First, Section 2.1 introduced circulant and Toeplitz matrices which are the main mathematical objects used in this thesis. Circulant and Toeplitz matrices are structured matrices in which each descending diagonal, from left to right, is constant. These structured matrices are the building blocks of our contribution on compact neural networks (Chapter 4) and enable fast approximation of the Lipschitz constant of convolution layers leading to a new regularization scheme (Chapter 5).

Finally, in Section 2.2.1, we gave a quick overview of the concept of supervised learning, which presents the mathematical tools for optimizing a parameterized function in order to map an input to an output based on a series of input-output pairs. Although the statistical learning framework considers generic hypothesis space, in this work we use a class of functions called neural networks presented in Section 2.2.2. We also presented, in Section 2.2.3, the concept of adversarial attacks and robustness of neural networks. We showed how a neural network can be sensitive to small perturbations to its input and thus vulnerable to adversarial examples. Reducing the sensitivity and therefore increasing the robustness of neural networks is the central theme of our second contribution presented in Chapter 5. Finally, in Section 2.2.4, we have presented some recent results on the theory of neural networks. These results give us insights on how neural networks generalize and a theoretical justification of the regularization scheme that we propose in Chapter 5.

Chapter 3

Related Work

Contents

This chapter, divided into two parts, is intended to provide an overview of the state of the art related to our contributions. First, we present current methods for building compact neural networks. Since the scope of application of these techniques is broad, we have chosen to focus mainly on work that uses linear algebra tools and more particularly structured matrices. We present in the first subsection an overview of general techniques for building compact neural networks. In the next subsection, we present in more detail the current methods for building compact neural networks with structured matrices. Finally, we discuss these techniques with respect to our contribution to compact neural networks.

The second part of this chapter presents current methods for regularizing the Lipschitz constant of neural networks with the aim of improving their robustness. This section is divided into four parts. First, we present techniques that focus on the computation of the Lipschitz constant of neural networks. Although theoretically and empirically interesting, we will see how these techniques do not scale and therefore cannot be applied to current neural network architectures. The following subsection presents the approach of Lipschitz regularization via the Lipschitz constant of individual layers of the networks. We describe the advantages and disadvantages of this approach. Moreover, in the third subsection, we focus our presentation on current techniques that compute the singular values of convolution layers. Finally, we discuss these techniques with respect to our contribution on Lipschitz regularization of convolutional neural networks.

3.1 Related Work on Compact Neural Networks

3.1.1 General Techniques to Build Compact Neural Networks

As seen in the Introduction (Chapter 1), scaling up networks can lead to better accuracy (Tan & Le, 2019; Brown et al. 2020). However, large neural networks lead to difficult and expensive training and after observing that a lot of parameters in large neural networks were redundant (Dai et al. 2018; Frankle & Carbin, 2018), an important question arises: *do neural networks need to be over-parameterized? And if not, how to build accurate and compact neural networks?*

Numerous other directions have been investigated to build compact and cost-effective neural networks without impacting the accuracy. For example Gupta et al. (2015) and Micikevicius et al. (2018) have proposed to represent weights with limited numerical precision to reduce training time and memory requirements. They used half-precision floating-point format instead of single-precision floating-point format which uses 32 bits of computer memory. In the same direction, Courbariaux et al. (2015) have proposed a method to train neural networks with binary weights without an important loss in the accuracy.

An important idea in model compression, proposed by Buciluǎ et al. (2006), is based on the observation that the model used for training is not required to be the same as the one used for inference. Indeed, models compressed after training can be deployed on smartphones or IoT devices. Based on this idea, multiple post-processing techniques have been developed: a quantization procedure which consists in converting the weights into a binary or integer formats *after* the training

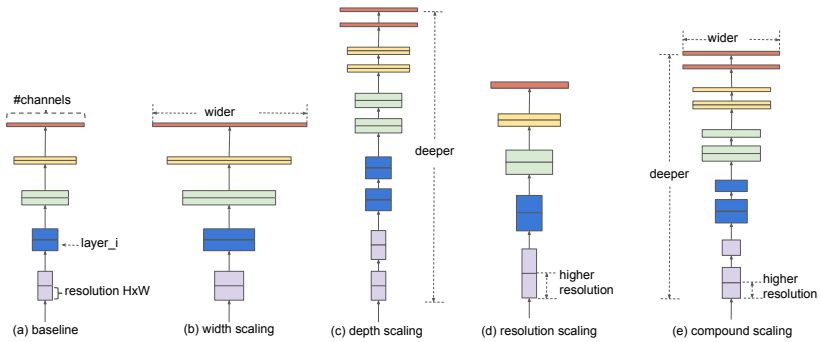


Figure 3.1: Illustration of the scaling of the EfficientNet architecture.

phase (Rastegari et al. 2016; Mellempudi et al. 2017), pruning techniques (Han et al. 2016; Lin et al. 2017; Dai et al. 2018) or sparsity regularizers (Collins & Kohli, 2014; Liu et al. 2015; Dai et al. 2018) which consists in removing redundant weights after training and taking advantage of the sparse structure of the weight matrices.

Sparse neural networks have also been extensively studied since the seminal work of Frankle & Carbin (2018) in which they propose the *Lottery Ticket Hypothesis*. This hypothesis states that there exists a sparse subnetwork of a dense neural network that when trained in isolation can match the test accuracy of the original dense network after training for at most the same number of iterations. This hypothesis led to a series of works on sparse neural networks (Malach et al. 2019; Zhou et al. 2019; Evci et al. 2020).

Moreover, Ba & Caruana (2014) have empirically demonstrated that shallow neural networks can learn the complex functions previously learned by other deep neural networks. This result led Hinton et al. (2015) to propose a technique called *model distillation* which consists in training a large complex model using all the available data and resources to be as accurate as possible, then a smaller and more compact model is trained to approximate the first model. Although interesting for deployment purposes, this approach still requires to train one large network and one shallow, which entails a significant training cost.

More recently, Zoph et al. (2018) and Real et al. (2019) have designed algorithms that automatically tune the width and depth of neural network architectures to obtain the best trade-off between compactness and accuracy. With this approach, Tan & Le (2019) found a new compound scaling method that uniformly scales network width and depth leading to efficient and compact architecture. Figure 3.1 illustrates the different scaling proposed by Tan & Le (2019).

3.1.2 Building Compact Neural Networks with Structured Matrices

An effective method to build compact neural networks is to constrain the hypothesis space by imposing a *structure* on the weight matrices which constitute the different layers of the network.

Structured Neural Networks with Low Rank Approximation

For example, Sainath et al. (2013) were among the first to use low-rank matrices in deep learning contexts followed by the work of Jaderberg et al. (2014) and Yu et al. (2017). Their work consists in replacing the weight matrices of size $n \times m$ by the product of two rectangular matrices of size $n \times r$ and $r \times m$, where r corresponds to the rank of the new matrix. In order to reduce the number of parameters, the rank r is chosen to be small such that $r \ll \min(m, n)$. By representing the weight matrices with a low-rank decomposition, one can reduce the storage from mn parameters to $(mr+nr)$ and accelerate the matrix-vector product from $\mathcal{O}(mn)$ to $\mathcal{O}(mr + rn)$. To enforce the low-rank constraint, reduced storage and computation time during training, the authors trained the coefficients of the two rectangular matrices directly. Formally, let $\mathbf{W} \in \mathbb{R}^{n\times m}$ be a weight matrix and let $\widetilde{\mathbf{W}}$ be the low-rank approximation of rank r of the matrix $\mathbf{W}$. Then, the low-matrix $\widetilde{\mathbf{W}}$ can be decomposed by the product of two rectangular matrices $\mathbf{U} \in \mathbb{R}^{n\times r}$ and $\mathbf{V} \in \mathbb{R}^{r\times m}$ such that $\widetilde{\mathbf{W}} = \mathbf{UV}$. Therefore, a neural network layer with low-rank approximation can be expressed as follows:

$$\phi_{\mathbf{U},\mathbf{V},\mathbf{b}}(\mathbf{x}) = \rho(\mathbf{UVx} + \mathbf{b}). \tag{3.1}$$

The scalar r defining the size of the two rectangular matrices becomes an hyper-parameter and controls the trade-off between the expressivity and compactness of the layer.

In the same vein, Oseledets (2011) have proposed the *Tensor Train decomposition* (TT-decomposition), which is based on the tensor rank decomposition (Tucker decomposition) proposed by Hitchcock (1927) and named after Tucker (1966). The TT-decomposition is defined as follows. Let $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1\times n_2\times\cdots\times n_{d-1}\times n_d}$ be a d-dimensional tensor. The Tensor-Train Decomposition factorizes $\boldsymbol{\mathcal{A}}$ in a product of third-order tensors and it is given by:

$$(\boldsymbol{\mathcal{A}})_{(i_1,\ldots,i_d)} = (\mathbf{G}^{(1)})_{(i_1,:)}(\boldsymbol{\mathcal{G}}^{(2)})_{(:,i_2,:)}(\boldsymbol{\mathcal{G}}^{(3)})_{(:,i_3,:)}\ldots(\mathbf{G}^{(d)})_{(:,i_d)} \tag{3.2}$$

where $\mathbf{G}^{(i)}$ are matrices and $\boldsymbol{\mathcal{G}}^{(i)}$ are third-order tensors of size $r_i \times r_{i+1}$ called *TT-cores*. The sequence $\{r_k\}_{k=0}^{d}$ is referred to as the ranks of the TT-representation. The above equation can be equivalently rewritten as a sum of elements of the TT-cores:

$$(\boldsymbol{\mathcal{A}})_{(i_1,\ldots,i_d)} = \sum_{\alpha_1,\ldots,\alpha_{d-1}} (\mathbf{G}^{(1)})_{(i_1,\alpha_1)} (\boldsymbol{\mathcal{G}}^{(2)})_{(\alpha_1,i_2,\alpha_2)} \ldots (\mathbf{G}^{(d)})_{(\alpha_{d-1},i_d)} \tag{3.3}$$

Oseledets (2011) have shown that for an arbitrary tensor $\boldsymbol{\mathcal{A}}$, several TT-representations exist with different ranks. The TT-decomposition can be very efficient in terms of memory requirement if the ranks are small. Indeed, the tensor $\boldsymbol{\mathcal{A}}$ has $\prod_{k=1}^{d} n_k$ values compared with $\sum_{k=1}^{d} n_k r_{k-1} r_k$ values.

The TT-decomposition has been extensively used in the context of deep learning. Novikov et al. (2015) was one of the first to use this technique to reduce the number of parameters of neural networks by using the decomposition to replace the fully connected layer of the VGG architecture (Simonyan & Zisserman, 2014). They reported a compression factor of the dense weight matrix up to 200000 times leading to the compression factor of the whole network up to 7 times with only 0.3 point drop of TOP-5 accuracy on ImageNet (Deng et al. 2009). With this work, Novikov et al. (2015) have demonstrated that the TT-decomposition allows an important reduction of the number of parameters while preserving the expressive power of the layers. Later, the TT-decomposition was used in other types of architectures. Garipov et al. (2016) used it to compress convolution layers as well as fully connected layers. Yang et al. (2017) used it in the context of video classification, Tjandra et al. (2017) compressed the layers of recurrent neural networks and finally, Ma et al. (2019) developed a compact architecture based on TT-decomposition for Language Modeling.

However, the Tensor-Train decomposition has some limitations. Although it can reduce the number of parameters when the ranks are low, finding the best alignment of the tensor dimensions in order to find the best optimized TT-cores remains a challenging problem, as stated by Pan et al. (2019).

Neural Networks with Diagonal and Circulant Matrices

Cheng et al. (2015) proposed to replace the weight matrix of a fully connected layer by the product of a circulant and a diagonal matrix leading to following structured layer:

$$\phi_{\mathbf{D},\mathbf{C},\mathbf{b}}(\mathbf{x}) = \rho(\mathbf{DCx} + \mathbf{b}) \ , \tag{3.4}$$

where the circulant matrix is learned by a gradient-based optimization algorithm and the diagonal matrix entries are sampled at random in $\{-1, 1\}$. The idea of replacing dense matrices with circulant ones comes from their use in dimensionality reduction with the *fast Johnson-Lindenstrauss transform* (Hinrichs & Vybíral, 2011; Vybíral, 2011), binary embedding (Yu et al. 2014), and kernel approximation (Yu et al. 2015), etc. Circulant matrices exhibit several interesting properties from the perspective of numerical computations. Recall from Theorem 2.1 that circulant matrices can be diagonalized with the Fourier Transform as follows:

$$\mathbf{C} = \frac{1}{n}\mathbf{U}_n^* \mathrm{diag}(\mathbf{U}_n \mathbf{c})\mathbf{U}_n \ . \tag{3.5}$$

where the vector $\mathbf{c}$ corresponds to the first columns of the matrix $\mathbf{C}$. This decomposition allows a compact representation in memory (n values instead of n^2) and efficient matrix-vector product with the FFT algorithm (see Algorithm 2.1). Despite the reduction of expressivity, Cheng et al. (2015) demonstrated good empirical results using only a fraction of the original weights (90% reduction).

Moczulski et al. (2016) built upon the work of Cheng et al. (2015) and Huhtanen & Perämäki (2015) and introduced two *Structured Efficient Linear Layers* (SELL) based on the Fourier and cosine transforms. First, by observing that the DC transform cannot express an arbitrary linear operator they proposed to apply the result of Huhtanen & Perämäki (2015) which states that almost all matrices can be decomposed as a product of DC transforms.

Theorem 3.1 (Reformulation from Huhtanen & Perämäki (2015))**.** *For every matrix $\mathbf{M} \in \mathbb{C}^{n\times n}$, for any $\epsilon > 0$, there exists a sequence of matrices $\{\mathbf{A}^{(i)}\}_{i\in[2n-1]}$ where $\mathbf{A}^{(i)}$ is a circulant matrix if i is odd, and a diagonal matrix otherwise, such that $\left\|\mathbf{A}^{(1)} \dots \mathbf{A}^{(2n-1)} - \mathbf{M}\right\| < \epsilon$.*

Based on this result, they proposed to parameterize the layers of a neural network with k products of diagonal and circulant matrices as follows:

$$\phi_{\mathbf{D},\mathbf{C},\mathbf{b}}(\mathbf{x}) = \rho\left(\left(\prod_{i=1}^{k} \mathbf{D}^{(i)}\mathbf{C}^{(i)}\right)\mathbf{x} + \mathbf{b}\right) \tag{3.6}$$

where $\mathbf{D}$ and $\mathbf{C}$ are sequences of k diagonal and circulant matrices respectively. This structured layer is therefore parameterized by $n(2k+1)$ values and the value k becomes a hyper-parameter controlling the trade-off between compactness and

expressivity. By diagonalizing the circulant matrix, the layer in Equation (3.6) can be expressed as a product of diagonal matrices and the Fourier transform as follows:

$$\phi^{\rho}_{\mathbf{d},\mathbf{c},\mathbf{b}}(\mathbf{x}) = \rho\left(\frac{1}{n^k}\left(\prod_{i=1}^{k} \operatorname{diag}\left(\mathbf{d}^{(i)}\right)\mathbf{U}_n^{*}\operatorname{diag}\left(\mathbf{U}_n\mathbf{c}^{(i)}\right)\mathbf{U}_n\right)\mathbf{x} + \mathbf{b}\right) \tag{3.7}$$

Although interesting and demonstrating good empirical results, the work of Moczulski et al. (2016) suffers from multiple limitations. First, the result from Huhtanen & Perämäki (2015) is expressed with respect to n, the size of the matrices $\mathbf{A}$. Therefore, the theorem does not provide any insights regarding the expressive power of k factors when k is much lower than $2n-1$ as it is the case in most practical scenarios they consider. Finally, in order to stay in the real domain, they replaced the Fourier transform in Equation (3.7) with the cosine transform thus learning a different kind of linear transform (see the work of Sanchez et al. (1995) which characterizes the matrices diagonalizable by the cosine transform). Furthermore, because the cosine transform does not diagonalize circulant matrices, Theorem 3.1 no longer applies.

General Representation of Structured Linear Maps: LDR and K-Matrices

General frameworks for structured matrices that reduce the memory footprint but also accelerate matrix-vector product operations have been used to build compact neural networks. Sindhwani et al. (2015) have used the notion of low displacement rank presented in Section 2.1.4 to learn a broad family of structured matrices. Recall from Theorem 2.4 that all matrices expressed as the following sum of products are called *Toeplitz-like* matrices:

$$\mathbf{M} = \frac{1}{2}\sum_{j=1}^{r} \mathbf{Z}_1(\mathbf{g}^{(j)})\mathbf{Z}_{-1}(\mathbf{J}_n\mathbf{h}^{(j)}) \tag{3.8}$$

where $\mathbf{Z}_f$ is an f-circulant matrix defined in Definition 2.2, $\mathbf{G} = \left(\mathbf{g}^{(1)} \dots \mathbf{g}^{(r)}\right), \mathbf{H} = \left(\mathbf{h}^{(1)} \dots \mathbf{h}^{(r)}\right) \in \mathbb{R}^{n\times r}$ with $r \ll n$ and $\mathbf{J}_n$ is the reflection of the $n \times n$ identity matrix. More precisely, Sindhwani et al. (2015) proposed to learn Toeplitz-like matrices by learning the factors $\mathbf{G}$ and $\mathbf{H}$. Therefore, they proposed the following parameterized layer:

$$\phi_{\mathbf{G},\mathbf{H},\mathbf{b}}(\mathbf{x}) = \rho\left(\left(\sum_{j=1}^{r} \mathbf{Z}_1\left(\mathbf{g}^{(j)}\right)\mathbf{Z}_{-1}\left(\mathbf{J}_n\mathbf{h}^{(j)}\right)\right)\mathbf{x} + \mathbf{b}\right) \tag{3.9}$$

where the rank r is a hyper-parameter and controls the number of parameters of the layer. In addition to offer fast matrix-vector product, they have showed that this class of layers is very rich from a modeling perspective. More precisely, they characterize the expressivity of the layer as follows:

Theorem 3.2 (LDR expressivity Pan (2001) and Sindhwani et al. (2015))**.**
The set of all $n \times n$ matrices that can be written as, $\frac{1}{2}\sum_{i=1}^{r} \mathbf{Z}_1(\mathbf{g}^{(i)})\mathbf{Z}_{-1}(\mathbf{J}_n\mathbf{h}^{(i)})$ for some $\mathbf{G} = \left(\mathbf{g}^{(1)} \dots \mathbf{g}^{(r)}\right), \mathbf{H} = \left(\mathbf{h}^{(1)} \dots \mathbf{h}^{(r)}\right) \in \mathbb{R}^{n\times r}$ contains:

- *All $n \times n$ Circulant and Skew-Circulant matrices for $r \geq 1$.*
- *All $n \times n$ Toeplitz matrices for $r \geq 2$.*
- *Inverses of Toeplitz matrices for $r \geq 2$.*
- *All products of the form $\mathbf{A}^{(1)} \dots \mathbf{A}^{(t)}$ for any $r \geq 2t$.*
- *All linear combinations of the form $\sum_{i=1}^{p} \beta_i \mathbf{A}^{(1,i)} \dots \mathbf{A}^{(t,i)}$ for any $r \geq 2pt$.*
- *All $n \times n$ matrices for $r = n$.*

where each $\mathbf{A}^{(i)}$ above is a Toeplitz matrix or the inverse of a Toeplitz matrix.

In the same line of work, Thomas et al. (2018) have proposed neural network layers directly form the Krylov decomposition presented in Theorem 2.3 which encompasses an even larger family of structured matrices including Toeplitz-like, Vandermonde-like, Cauchy-like ones. Despite being elegant and general, we found that the LDR framework suffers from several limits which are inherent to its generality and makes it difficult to use in the context of deep neural networks. As acknowledged by the authors, the number of parameters required to represent a given structured matrix (a Toeplitz matrix) in practice is unnecessarily high (higher than required in theory) making the training very hard.

More recently, another type of generalization of structured linear maps has been proposed by Dao et al. (2019) and Dao et al. (2020). They introduced a family of matrices called *kaleidoscope matrices* (K-matrices) which are the product of sparse matrices with specific predefined sparsity patterns. They showed that this type of matrices can capture any sparse matrix with near-optimal space (parameter) and time (arithmetic operation) complexity. The authors claim that their structured linear maps can capture more common structures with a few numbers of parameters than the displacement operators presented above. More precisely, their representation is based on products of a particular building block known as a butterfly matrix introduced by Parker (1995). Butterfly matrices have been extensively used in numerical linear algebra (Parker, 1995; Li et al. 2015) and machine learning (Mathieu & LeCun, 2014; Jing et al. 2017; Munkhoeva et al. 2018; Choromanski et al. 2019; Dao et al. 2019).

3.1.3 Discussion

In this section, we have shown current methods and techniques for designing compact neural networks with structured matrices. Our contributions on *Deep Diagonal Circulant Neural Networks* are a direct follow-up to the work of Cheng et al. (2015), Sindhwani et al. (2015), Moczulski et al. (2016) and Thomas et al. (2018) focusing on compact neural networks with *structured matrices*. More precisely, we extend the work of Moczulski et al. (2016) by training *fully structured networks* (*i.e.*, networks with structured layers only) hence demonstrating that diagonal-circulant layers are able to model complex relations between inputs and outputs. Although, this diagonal-circulant layers fit in the low displacement rank framework, we demonstrate much better performances in practice. Indeed, thanks to a solid theoretical analysis and thorough experiments, we were able to train deep (up to 40 layers) circulant neural networks, and apply, for the first time, this structured architecture in the context of large-scale video classification. This contrasts with previous experiments in which only one or a few dense layers were replaced inside a large redundant network such as VGG (Simonyan & Zisserman, 2014).

3.2 Related Work on Lipschitz Regularization

3.2.1 The Global Lipschitz Constant of Neural Networks

The regularization of the Lipschitz constant of neural networks has seen a growing interest in the last few years. Indeed, numerous results have shown that neural networks with a low Lipschitz constant exhibit better generalization (Bartlett et al. 2017) and higher robustness to adversarial attacks (Szegedy et al. 2014; Tsuzuku et al. 2018; Farnia et al. 2019).

The Lipschitz constant, defined in Definition 2.7, is a measure of the stability of the network. If the Lipschitz constant is high, the network will tend to be more sensitive to input perturbations, meaning, if the input changes by ϵ, the output changes by at most $k\epsilon$. The Lipschitz constant of a function can also be expressed using the differential operator as follows:

Theorem 3.3 (Rademacher's Theorem). *If $f : \mathbb{R}^n \to \mathbb{R}^m$ is a Lipschitz continuous function, then f is differentiable almost everywhere. Moreover, if f is Lipschitz continuous, then*

$$\mathrm{Lip}(f) = \sup_{\mathbf{x} \in \mathbb{R}^n} \|\mathrm{D}_{\mathbf{x}} f(\mathbf{x})\|_2 \tag{3.10}$$

where $\mathrm{D}_{\mathbf{x}}$ is the differential operator of f at $\mathbf{x}$.

Tsuzuku et al. (2018) have studied the relationship between the robustness and the Lipschitz constant and the margin of neural networks. By the definition of the Lipschitz constant, we have the following:

$$\|N_\Omega(\mathbf{x}) - N_\Omega(\mathbf{x} + \boldsymbol{\tau})\|_2 \leq \mathrm{Lip}(N_\Omega)\|\boldsymbol{\tau}\|_2 \tag{3.11}$$

Recall the margin operator $M : \mathbb{R}^k \times [k] \to \mathbb{R}$ from Section 2.2.4 defined as:

$$M(\mathbf{v}, j) \triangleq \mathbf{v}_j - \max_{i \neq j} \mathbf{v}_i \tag{3.12}$$

Then, we have the following proposition which characterizes the robustness of a neural network with respect to its margin and Lipschitz constant.

Proposition 3.1 (Tsuzuku et al. (2018)).

$$M\big(N_\Omega(\mathbf{x}), y\big) \geq \sqrt{2}\mathrm{Lip}(N_\Omega)\|\boldsymbol{\tau}\|_2 \quad \Longrightarrow \quad M\big(N_\Omega(\mathbf{x} + \boldsymbol{\tau}), y\big) \geq 0 \tag{3.13}$$

If the inequality on the right-hand side of Equation (3.13) is verified then the adversarial margin is positive, *i.e.*, the network correctly predicts the label. From this proposition, we can conclude that for a given neural network with specific margins, a lower Lipschitz constant allows for an increase in robustness. Note that the margin is already maximized in a multi-class setting with the cross-entropy loss as stated in Hein & Andriushchenko (2017). A multitude of work have tried to reduce the Lipschitz constant in order to improve adversarial robustness. However, Virmaux & Scaman (2018) have shown that computing the exact Lipschitz constant of a neural network is NP-hard. The following theorem shows that, even for shallow neural networks, exact Lipschitz computation is not achievable in polynomial time:

Theorem 3.4 (Virmaux & Scaman (2018)). *Let us define the problem associated with the exact computation of the Lipschitz constant of a 2-layer neural network with* ReLU *activation:*

Input: *Two matrices* $\mathbf{W}^{(1)} \in \mathbb{R}^{l \times n}$ *and* $\mathbf{W}^{(2)} \in \mathbb{R}^{m \times l}$*, and a constant* $c \geq 0$*.*

Question: *Let* $N = \mathbf{W}^{(2)} \circ \rho \circ \mathbf{W}^{(1)}$ *where* ρ *is the* ReLU *activation function.* Is the Lipschitz constant $\mathrm{Lip}(N) \leq c$?

Then, assuming that $\boldsymbol{P} \neq \boldsymbol{NP}$*, the problem above is NP-hard.*

To overcome this difficulty, researchers have relied on devising a tight upper bound of the Lipschitz constant. For example, Virmaux & Scaman (2018) have shown that the Lipschitz constant of a neural network N can be explicitly formulated using Theorem 3.3 and the chain rule:

$$\mathrm{Lip}(N) = \sup_{x \in \mathbb{R}^n} \left\| \mathbf{W}^{(p)} \mathrm{diag}(\rho_p'(\theta_p)) \dots \mathbf{W}^{(2)} \mathrm{diag}(\rho_1'(\theta_1)) \mathbf{W}^{(1)} \right\|_2, \tag{3.14}$$

where $\theta_i = \phi^{\rho_i}_{\mathbf{W}^{(i)}, \mathbf{b}^{(i)}} \circ \dots \circ \phi^{\rho_1}_{\mathbf{W}^{(1)}, \mathbf{b}^{(1)}}(\mathbf{x})$ is the intermediate output after i layers and ρ_i' is the derivative of ρ_i. The Lipschitz of the neural network N can then be upper bounded as follows:

$$\begin{aligned}
\mathrm{Lip}(N) &\leq \max_{\forall i,\ \sigma_i \in [0,1]^{w^{(i+1)}}} \left\| \mathbf{W}^{(p)} \mathrm{diag}(\sigma_{p-1}) \dots \mathrm{diag}(\sigma_1) \mathbf{W}^{(1)} \right\|_2 \\
&\leq \max_{\forall i,\ \sigma_i \in [0,1]^{w^{(i+1)}}} \left\| \mathbf{\Sigma}^{(p)} \mathbf{V}^{(p)\top} \mathrm{diag}(\sigma_{p-1}) \dots \mathrm{diag}(\sigma_1) \mathbf{U}^{(1)} \mathbf{\Sigma}^{(1)} \right\|_2 \\
&\leq \prod_{i=1}^{p-1} \max_{\sigma_i \in [0,1]^{w^{(i+1)}}} \left\| \widetilde{\mathbf{\Sigma}}^{(i+1)} \mathbf{V}^{(i+1)\top} \mathrm{diag}(\sigma_{i+1}) \mathbf{U}^{(i)} \widetilde{\mathbf{\Sigma}}^{(i)} \right\|_2
\end{aligned} \tag{3.15}$$

where $\widetilde{\mathbf{\Sigma}}^{(i)} = \mathbf{\Sigma}^{(i)}$ if $i \in \{1, p\}$ and $\widetilde{\mathbf{\Sigma}}^{(i)} = \mathbf{\Sigma}^{(i)^{1/2}}$ otherwise. The first inequality is due to the fact that the derivatives of the activation functions are bounded, *i.e.*, $\rho_i(\mathbf{x}) \in [0,1]^{w^{(i+1)}}$, the second inequality is obtained by decomposing each weight matrix $\mathbf{W}^{(i)}$ with the *Singular Value Decomposition* such that $\mathbf{W}^{(i)} = \mathbf{U}^{(i)} \mathbf{\Sigma}^{(i)} \mathbf{V}^{(i)\top}$; and finally, the last inequality is due to the submultiplicativity of the operator norm. Although accurate, this bound is still computationally expensive to compute due to the singular value decomposition and the optimization for each layer. In the same line of research, recent work (Fazlyab et al. 2019a,b; Latorre et al. 2020) has proposed a tight bound on the Lipschitz constant of the full network with the use of semi-definite programming. More precisely, Fazlyab et al. (2019) have demonstrated the following result:

Theorem 3.5 (Lipschitz bounds Fazlyab et al. (2019)). *Consider a neural network* $N : \mathbb{R}^n \to \mathbb{R}^m$ *such that* $N(\mathbf{x}) = \mathbf{W}^{(2)}\rho(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}$. *Suppose the activation function* ρ *is* slope-restricted *in the sector* $[\alpha, \beta]$, *i.e.,*

$$\alpha \leq \frac{\rho(y) - \rho(x)}{y - x} \leq \beta \quad \forall x, y \in \mathbb{R}. \tag{3.16}$$

Define the set $\mathcal{T}_n$ *as the following:*

$$\mathcal{T}_n = \{\mathbf{T} \in \mathbb{S}^n \mid \mathbf{T} = \sum_{i=1}^{n} \lambda_{ii}\mathbf{e}^{(i)}\mathbf{e}^{(i)\top} + \sum_{1 \leq i < j \leq n} \lambda_{ij}(\mathbf{e}^{(i)} - \mathbf{e}^{(j)})(\mathbf{e}^{(i)} - \mathbf{e}^{(j)})^\top, \lambda_{ij} \geq 0\}.$$

where $\mathbb{S}^n$ *is the set of all symmetric matrices of size* $n \times n$. *Suppose there exists a constant* $c > 0$ *such that the matrix inequality*

$$\mathbf{M}(c, \mathbf{T}) \triangleq \begin{pmatrix} -2\alpha\beta\mathbf{W}^{(1)\top}\mathbf{T}\mathbf{W}^{(1)} - c\mathbf{I}_n & (\alpha + \beta)\mathbf{W}^{(1)\top}\mathbf{T} \\ (\alpha + \beta)\mathbf{T}\mathbf{W}^{(1)} & -2\mathbf{T} + \mathbf{W}^{(2)\top}\mathbf{W}^{(2)} \end{pmatrix} \leq 0, \tag{3.17}$$

holds for some $\mathbf{T} \in \mathcal{T}_n$. *Then* $\|N(\mathbf{x}) - N(\mathbf{y})\|_2 \leq \sqrt{c}\|\mathbf{x} - \mathbf{y}\|_2$ *for all* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$.

From Theorem 3.5, the constant c is an upper bound on the Lipschitz constant of the network. The authors proposed to find the tightest bound by solving the following optimization problem (Semidefinite Program):

$$\text{minimize} \quad c \quad \text{subject to} \quad \mathbf{M}(c, \mathbf{T}) \leq 0 \quad \text{and} \quad \mathbf{T} \in \mathcal{T}_n, \tag{3.18}$$

where the decision variables are $(c, \mathbf{T}) \in \mathbb{R}_+ \times \mathcal{T}_n$. Note that $\mathbf{M}(c, \mathbf{T})$ is linear in c and $\mathbf{T}$ and the set $\mathcal{T}_n$ is convex. Although, these works on devising a global bound on the Lipschitz constant of a neural network are theoretically interesting, they lack scalability They can only be computed on small networks and cannot be used during the training of large neural networks for regularization purposes.

3.2.2 Lipschitz Constant of Individual Layers

Instead of regularizing the ERM using the global Lipschitz constant, researchers have devised techniques to reduce the Lipschitz constant of *individual layers* instead. The global Lipschitz of a neural network can easily be upper bounded by the product of the spectral norm of each weight matrix as follows:

Proposition 3.2 (Virmaux & Scaman (2018)). *Let N be a neural network of p layers with 1-Lipschitz activation functions (e.g. ReLU, Leaky ReLU, Tanh, Sigmoid, etc.), then, the Lipschitz constant of the neural network can be upper bounded as follows:*

$$\mathrm{Lip}(N) \leq \prod_{i=1}^{p} \left\| \mathbf{W}^{(i)} \right\|_2 , \tag{3.19}$$

where $\mathbf{W}^{(i)}$ are the weights matrices of the neural network.

Remark. *The Lipschitz constant of a layer $\phi^{\rho}_{\mathbf{W},\mathbf{b}}$ (with a 1-Lipschitz activation function) is equal to the spectral norm of the matrix $\mathbf{W}$ (largest singular value). Let $\phi^{\rho}_{\mathbf{W},\mathbf{b}} : \mathbb{R}^n \to \mathbb{R}^m$ such that $\phi^{\rho}_{\mathbf{W},\mathbf{b}} = \rho(\mathbf{W}\mathbf{x} + \mathbf{b})$ then by definition of the Lipschitz constant (see Definition 2.7) and of the operator norm, we have:*

$$\mathrm{Lip}\left(\phi^{\rho}_{\mathbf{W},\mathbf{b}}\right) = \sup_{\substack{\mathbf{x}\in\mathbb{R}^n \\ \mathbf{x}\neq 0}} \frac{\|\mathbf{W}\mathbf{x}\|_2}{\|\mathbf{x}\|_2} = \|\mathbf{W}\|_2 \tag{3.20}$$

The trivial bound given by the product of layer-wise Lipschitz constants in Equation (3.19) is known to be loose and pessimistic. Furthermore, we can show that reducing the Lipschitz constant of each layer independently does not imply that the global Lipschitz constant of the network will be reduced.

Proposition 3.3. *Let N be a neural network, then decreasing the Lipschitz constant of one or more layers does not imply reducing the Lipschitz constant of the network, i.e.,* $\mathrm{Lip}(N)$.

Proof of Proposition 3.3. Let us prove this claim with a counter-example. Let $N_1(\mathbf{x}) = \mathbf{A}^{(2)}\rho(\mathbf{A}^{(1)}\mathbf{x})$ and $N_2(\mathbf{x}) = \mathbf{B}^{(2)}\rho(\mathbf{B}^{(1)}\mathbf{x})$ where ρ is the ReLU activation function. Let

$$\mathbf{A}^{(1)} = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix} \quad \mathbf{A}^{(2)} = \begin{pmatrix} -1 & -1 \\ -1 & 0 \end{pmatrix}$$

$$\mathbf{B}^{(1)} = \begin{pmatrix} 0 & 0 \\ 0 & -1 \end{pmatrix} \quad \mathbf{B}^{(2)} = \begin{pmatrix} -1 & -1 \\ -1 & -1 \end{pmatrix}$$

then: $\left\|\mathbf{A}^{(1)}\right\|_2 = 1,\ \left\|\mathbf{A}^{(2)}\right\|_2 = \sqrt{2}$ and $\left\|\mathbf{B}^{(1)}\right\|_2 = 1,\ \left\|\mathbf{B}^{(2)}\right\|_2 = 2$

From Theorem 3.3 and the chain rule, the Lipschitz constant of the networks N_1 and N_2 can be expressed as follows:

$$\mathrm{Lip}(N_1) = \sup_{\mathbf{x}\in[0,1]^2} \left\|\mathbf{A}^{(2)}\mathrm{diag}(\mathbf{x})\mathbf{A}^{(1)}\right\|_2$$
$$\mathrm{Lip}(N_2) = \sup_{\mathbf{x}\in[0,1]^2} \left\|\mathbf{B}^{(2)}\mathrm{diag}(\mathbf{x})\mathbf{B}^{(1)}\right\|_2$$

It is easy to verify that:

$$\mathrm{Lip}(N_1) = \frac{1+\sqrt{5}}{2} \approx 1.618 \quad \text{and} \quad \mathrm{Lip}(N_2) = \sqrt{2} \approx 1.414$$

which concludes the proof. ■

While we cannot have a guarantee that the global Lipschitz will be reduced, we could still have an idea of the value of the global Lipschitz with the upper bound presented in Equation (3.19).

Huster et al. (2018) have demonstrated several limitations on the expressive power of neural networks where the product of layer-wise Lipschitz constants is constrained. In the same vein, Couellan (2019) empirically showed that Lipschitz Regularization offers a trade-off between adversarial robustness and expressivity of the network. However, the bound in Equation (3.19) appears in multiple generalization bound (Bartlett et al. 2017; Neyshabur, 2017; Golowich et al. 2018) and adversarial generalization (Farnia et al. 2019) (see Chapter 2) which could suggest that reducing the bound would improve the generalization capabilities of neural networks and its robustness.

Based on this theoretical insight, researchers have developed several techniques to constrain the Lipschitz constant of each layer in order to improve the generalization and robustness of neural networks. A technique to enforce 1-Lipschitz layers is to impose or promote an orthogonality constrain of the weight matrices. A square orthogonal matrix $\mathbf{M}$ is a matrix whose columns and rows are orthogonal unit vectors and all eigenvalues are equal to 1. Cisse et al. (2017) and more recently Huang et al. (2020) and Wang et al. (2020) have proposed to minimize the following term:

$$\frac{\beta}{2}\left\|\mathbf{W}^\top\mathbf{W} - \mathbf{I}\right\|_2 , \tag{3.21}$$

to promote the orthogonality constraint, in addition to the usual loss function: In the above equation, the hyper-parameter β controls the constraint. A higher β would

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** $\|\mathbf{W}\mathbf{x}\|_2 / \|\mathbf{x}\|_2$

lead to a better orthogonality constraint and therefore, a Lipschitz constant "almost" equal to 1 for all the layers.

On the other hand, Anil et al. (2019) proposed to enforce the orthogonality of weight matrices by directly optimizing on the Stiefel manifold (*i.e.*, the manifold of orthogonal matrices, see Absil et al. (2009)). To perform this optimization, they made use of an iterative algorithm first introduced by Björck & Bowie (1971). For a given matrix $\mathbf{W} = \mathbf{W}^{(0)}$, the algorithm finds the closest orthonormal matrix by computing the following term:

$$\mathbf{W}^{(k+1)} = \mathbf{W}^{(k)} \left(\mathbf{I} + \frac{1}{2}\mathbf{V}^{(k)} + \cdots + (-1)^r \binom{-\frac{1}{2}}{r} \left(\mathbf{V}^{(k)}\right)^r \right) \tag{3.22}$$

where $\mathbf{V} = \mathbf{I} - \mathbf{W}^{(k)\top}\mathbf{W}^{(k)}$. Although this algorithm works well on dense matrices, it can be difficult to apply it to convolutions. Li et al. (2019) built upon this idea and proposed an algorithm to enforce the orthogonality of convolution layers. They used the orthogonal projection proposed by Kautsky & Turcajová (1994) and Xiao et al. (2018) to build convolutional neural networks with orthogonal convolutions.

All techniques that impose an orthogonality constraint on the weights matrices successfully reduce the Lipschitz constant of the layers of the networks. Moreover, when the Lipschitz constants of all the layers are low, we could have an idea of the value of the global Lipschitz with the upper bound of Equation (3.19) (*i.e.*, if Lipschitz constant of all the layers are equal to 1, then, the network will have a Lipschitz constant below 1). However, enforcing the orthogonality constraint, either by regularizing with the term of Equation (3.21) or by optimizing on the Stiefel manifold, is the costly operation which make it difficult to scale on large neural networks.

Another technique, called *Spectral Normalization*, consists in normalizing each weight matrix by its largest singular value, thus imposing each layer to be 1-Lipschitz.

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 $\mathrm{Lip}(f)$

```
Initialize x with a random vector matching the shape of the convolution input
for i = 1 to N do
    x ← f(x)/‖f(x)‖_2
    x ← g(x)/‖g(x)‖_2
end for
return ‖f(x)‖_2/‖x‖_2
```

As with the orthogonality constraint, this technique leads the network to have a global Lipschitz constant at most 1. Yoshida & Miyato (2017) were the first to propose this method to improve the generalization of neural networks followed by (Gouk et al. 2018; Miyato et al. 2018; Farnia et al. 2019) for improving generalization and robustness against adversarial attacks. In order to perform spectral normalization, they divided the values of each weight matrix by an approximation of its largest singular value. The approximation of the largest singular was computed using the power method (Golub & Van der Vorst, 2000).

The power method is an iterative eigenvalue algorithm (also known as the Von Mises iteration (Mises & Pollaczek-Geiringer, 1929)). Given a matrix $\mathbf{W}$ and a random vector $\mathbf{b}^{(0)}$, the eigenvector associated with the largest eigenvalue of the matrix $\mathbf{W}$ can be computed with the following recurrence relation:

$$\mathbf{b}^{(k+1)} = \frac{\mathbf{W}\mathbf{b}^{(k)}}{\left\|\mathbf{b}^{(k)}\right\|_2} \tag{3.23}$$

Then, the largest eigenvalue (when we talk about "largest eigenvalue" we mean in absolute value) can be obtained with the *Rayleigh quotient*:

$$\sigma_1(\mathbf{W}) = \frac{\mathbf{b}^{(k)\top}\mathbf{W}\mathbf{b}^{(k)}}{\mathbf{b}^{(k)\top}\mathbf{b}^{(k)}} \tag{3.24}$$

With a sufficient number of iterations, the algorithm provably converges to the largest eigenvalue of the matrix. To find the largest singular value, we can leverage the relation between eigenvalues and singular values:

$$\sigma(\mathbf{W}) = \sqrt{\lambda(\mathbf{W}^\top\mathbf{W})} \tag{3.25}$$

The rate of convergence of the algorithm depends on the ratio between the second-largest eigenvalue and the largest eigenvalue. Indeed, a ratio close to one can lead to slow convergence. The pseudocode of the power method is given in Algorithm 3.1. Altough, Algorithm 3.1 needs explicit matrix for computing the largest singular value, Farnia et al. (2019) and Ryu et al. (2019) extended the power method to convolution layers where the matrix $\mathbf{W}$ is not explicitly constructed. The pseudocode of their method is presented in Algorithm 3.2.

In the context of deep learning and spectral normalization, the largest singular value needs to be computed for each layer of the network at each step of the training. Given that current state-of-the-art architecture have between 50 and 100 layers (He et al. 2016; Tan & Le, 2019), using the power method *until convergence* is prohibitive. In Chapter 5, we propose a new regularization scheme for reducing the Lipschitz constant of individual layers. We will shown in Section 5.4.3 that our approach is more efficient that the power method even with a small number of iterations.

3.2.3 Singular Values of Convolutional Layers

The power method is not the only technique available for approximating the largest singular value (Lipschitz constant) of a convolution layer. Several works have devised bounds or approximations on the largest singular value of convolution layers by exploiting the *structure* of the convolution operation (Jia et al. 2017; Sedghi et al. 2018; Bibi et al. 2019; Singla & Feizi, 2019).

To approximate the singular values of a convolution layer, Sedghi et al. (2018) have exploited the properties of doubly-block circulant matrices (*i.e.*, a circulant block matrix where each block is also a circulant matrix). Indeed, a doubly-block circulant matrix is the matrix representation of a convolution with circulant padding. In their work, Sedghi et al. (2018) assume that the properties of doubly-block circulant matrices are 'close' to the properties of a doubly-block Toeplitz matrix.

To compute the singular values of doubly-block circulant matrices, Sedghi et al. (2018) have demonstrated the following result:

Theorem 3.6 (Theorem 5 from Sedghi et al. (2018)). *Let* $\mathbf{A}$ *be a doubly-block circulant matrix such that:*

$$\mathbf{A} = \begin{pmatrix} \mathsf{C}^{(0)} & \mathsf{C}^{(n-1)} & \mathsf{C}^{(n-2)} & \cdots & \cdots & \mathsf{C}^{(1)} \\ \mathsf{C}^{(1)} & \mathsf{C}^{(0)} & \mathsf{C}^{(n-1)} & \ddots & & \vdots \\ \mathsf{C}^{(2)} & \mathsf{C}^{(1)} & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \mathsf{C}^{(n-1)} & \mathsf{C}^{(n-2)} \\ \vdots & & \ddots & \mathsf{C}^{(1)} & \mathsf{C}^{(0)} & \mathsf{C}^{(n-1)} \\ \mathsf{C}^{(n-1)} & \cdots & \cdots & \mathsf{C}^{(2)} & \mathsf{C}^{(1)} & \mathsf{C}^{(0)} \end{pmatrix}$$

where $\mathsf{C}^{(i)} = \mathrm{circ}(\mathbf{c}_i),\ \forall i \in \mathcal{I}_n^+$. *Let* $\mathbf{K} = (\mathbf{c}_0, \mathbf{c}_1, \cdots, \mathbf{c}_{n-1})^\top$ *then, the singular values of the doubly-block circulant matrix* $\mathbf{A}$ *are the modulus of the entries of* $\mathbf{U}_n^\top \mathbf{K} \mathbf{U}_n$.

To prove Theorem 3.6, Sedghi et al. (2018) used the diagonalization of doubly-block circulant matrices (see Chapter 2, Equation (2.31)). The main advantage of this approach is that the singular values of a doubly-block circulant matrix can be computed with the Fast Fourier Transform algorithm (see Section 2.1) which offers a reduced complexity compared to classical approaches for computing the singular values of a matrix. However, this approach exhibits several limitations. First, this method results in a loose approximation of the maximal singular value of a convolution layer which does not use the circulant padding, which is often the case in practical settings. Also, the complexity of their algorithm is dependent on the size of the input which can be high for large datasets. Finally, for multi-channel convolution, their method requires the computation of the spectral norm of n^2 matrices each of size $c_{in} \times c_{out}$ as stated in the following theorem:

Theorem 3.7 (Theorem 6 from Sedghi et al. (2018)). *Let* $\mathbf{M}$ *be the matrix encoding the linear transform computed by a multi-channel convolution layer. Let* $\mathbf{K} \in \mathbb{R}^{c_{in} \times c_{out} \times n \times n}$ *such that* $(\mathbf{K})_{i,j}$ *for all* $i, j \in [n]^2$ *be constructed as in Theorem 3.6, Let* $\widetilde{\mathbf{K}}_{i,j} = \mathbf{U}_n^\top (\mathbf{K})_{i,j} \mathbf{U}_n$ *and define the following operator matrix*

$$\mathbf{P}(i,j) = \begin{pmatrix} \left((\widetilde{\mathbf{K}})_{(0,0)}\right)_{i,j} & \cdots & \left((\widetilde{\mathbf{K}})_{(0,c_{out}-1)}\right)_{i,j} \\ \vdots & & \vdots \\ \left((\widetilde{\mathbf{K}})_{(c_{in}-1,0)}\right)_{i,j} & \cdots & \left((\widetilde{\mathbf{K}})_{(c_{in}-1,c_{out}-1)}\right)_{i,j} \end{pmatrix} \tag{3.26}$$

Then

$$\sigma(\mathbf{M}) = \bigcup_{i,j=0}^{n-1} \sigma(\mathbf{P}(i,j)). \tag{3.27}$$

In the same vein, Singla & Feizi (2019) have used the properties of convolutions to devise several bounds on the singular values of convolution layers. Recall from Section 2.1.3 that a convolution kernel is a 4 dimensional tensor of size $c_{out} \times c_{in} \times k_1 \times k_2$. Singla & Feizi (2019) have demonstrated that the largest singular value of a convolution layer $\phi_{\mathbf{K}}$ parameterized by a kernel $\mathbf{K}$ can be upper-bounded as follows:

Theorem 3.8 (Reformulation of Theorem 1 from Singla & Feizi (2019))**.** *Let* $\mathbf{K} \in \mathbb{R}^{c_{out} \times c_{in} \times k_1 \times k_2}$ *be the kernel of a convolution layer* $\phi_{\mathbf{K}}$*, then,*

$$\mathrm{Lip}(\phi_{\mathbf{K}}) \leq \min\left\{\sqrt{k_1 k_2}\|\mathbf{R}\|_2, \sqrt{k_2 k_2}\|\mathbf{S}\|_2\right\} \tag{3.28}$$

where $\mathbf{R}$ *and* $\mathbf{S}$ *are matrices of size* $k_1 c_{out} \times k_2 c_{in}$ *and* $k_2 c_{out} \times k_1 c_{in}$ *defined as follows:*

$$\mathbf{R} = \begin{pmatrix} (\mathbf{K})_{0,0} & \cdots & (\mathbf{K})_{0,c_{in}-1} \\ (\mathbf{K})_{1,0} & \cdots & (\mathbf{K})_{1,c_{in}-1} \\ \vdots & \ddots & \vdots \\ (\mathbf{K})_{c_{out}-1,0} & \cdots & (\mathbf{K})_{c_{out}-1,c_{in}-1} \end{pmatrix} \tag{3.29}$$

$$\mathbf{S} = \begin{pmatrix} (\mathbf{K})^\top_{0,0} & \cdots & (\mathbf{K})^\top_{0,c_{in}-1} \\ (\mathbf{K})^\top_{1,0} & \cdots & (\mathbf{K})^\top_{1,c_{in}-1} \\ \vdots & \ddots & \vdots \\ (\mathbf{K})^\top_{c_{out}-1,0} & \cdots & (\mathbf{K})^\top_{c_{in}-1,c_{out}-1} \end{pmatrix} \tag{3.30}$$

In order to prove this result, Singla & Feizi (2019) built upon the work of Sedghi et al. (2018) and have also only considered circulant convolutions (performed by doubly-block circulant matrices). Instead of proposing a method to compute *all* singular values of the equivalent doubly-block circulant matrix, their method is an upper-bound on the largest singular value of the Jacobian of the convolution. Because this method is independent of the input dimension, the computational complexity is substantially reduced compared to the approach of Sedghi et al. (2018), however, the reduction in computational complexity is at the expense of accuracy as we will show in Chapter 5.

3.2.4 Discussion

We have presented state-of-the-art methods for regularizing the Lipschitz constant of neural networks with the aim to improve their robustness against adversarial attacks. The power method (Golub & Van der Vorst, 2000) is a popular technique for approximating the maximal singular value of a matrix. Recent works in deep learning use this method in a wide variety of settings, for example, robustness (Tsuzuku et al. 2018; Farnia et al. 2019), generalization (Yoshida & Miyato, 2017; Gouk et al. 2018) or to stabilize the training of Generative Adversarial Networks (GANs) (Miyato et al. 2018). Despite a number of interesting results, using the power method is expensive and results in prohibitive training times. Other approaches to regularize the Lipschitz constant of neural networks have been proposed by Sedghi et al. (2018) and Singla & Feizi (2019). These methods exploit the properties of circulant matrices to approximate the maximal singular value of a convolution layer. Although interesting, theses method results in a loose approximation of the maximal singular value. Our work is positioned at the intersection between these works, we will introduce a new approach for regularizing the Lipschitz constant of neural networks, that is more efficient than the power method and more accurate than methods relying on the structure of convolutions.

Chapter 4

Diagonal and Circulant Matrices for Compact Neural Networks

Contents

4.1 Introduction

As seen in the previous chapters, structured matrices are at the very core of most of the work on compact networks. Despite substantial efforts (*e.g.*, Cheng et al.

(2015) and Moczulski et al. (2016)), the performance of compact models is still far from achieving an acceptable accuracy motivating their use in real-world scenarios. This raises several questions about the effectiveness of such models and about our ability to train them. In particular two main questions call for investigation:

1. *What is the expressive power of structured layers compared to dense layers?*
2. *How to efficiently train deep neural networks with a large number of structured layers?*

We aim at answering these questions by studying deep diagonal-circulant neural networks (*a.k.a.* DCNNs), which are deep neural networks in which weight matrices are the product of diagonal and circulant ones.

To answer the first question, we propose an analysis of the expressivity of DCNNs by extending the results obtained by Huhtanen & Perämäki (2015) which states that any matrix can be decomposed into the product of $2n - 1$ alternating diagonal and circulant matrices. We introduce a new bound on the number of diagonal-circulant products required to approximate a matrix that depends on its rank. Building on this result, we demonstrate that a DCNN with bounded width and small depth can approximate any dense neural networks with ReLU activations.

To answer the second question, we first describe a theoretically sound initialization procedure for DCNN which allows the signal to propagate through the network without vanishing or exploding. Furthermore, we provide a number of empirical insights to explain the behavior of DCNNs and show the impact on the number of nonlinearities in the network on the convergence rate and the accuracy of the network. By combining all these insights, we are able (for the first time) to train large and deep DCNNs and demonstrate the good performance of these networks on a large-scale application (the *YouTube-8M* video classification problem) and obtain very competitive accuracy.

The chapter is organized as follows: Section 4.2 introduces our new result extending the one from Huhtanen & Perämäki (2015). Section 4.3 proposes a theoretical analysis on the expressivity of DCNNs. Section 4.4 describes two efficient techniques for training deep diagonal-circulant neural networks. Section 4.5 presents extensive experiments to compare the performance of deep diagonal-circulant neural networks in different settings with respect to other state-of-the-art approaches. Finally, Section 4.6 provides concluding remarks.

4.2 Diagonal and Circulant Matrices for Matrix Decomposition

As seen in the Background (Chapter 2), circulant matrices exhibit several interesting properties from the perspective of numerical computations. Most importantly, any $n \times n$ circulant matrix $\mathbf{C}$ can be represented using only n coefficients instead of the n^2 coefficients required to represent classical unstructured matrices. In addition, the matrix-vector product is simplified from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log n)$ using the convolution theorem. As we will show in this chapter, circulant matrices combined with diagonal matrices can have a strong expressive power. They can also be used as building blocks to represent any linear transform (Schmid et al. 2000; Huhtanen & Perämäki, 2015) with an arbitrary precision.

We are interested in the relation between the product of diagonal and circulant matrices and the expressivity of low-rank matrices. Huhtanen & Perämäki (2015) were able to bound the number of factors that is required to approximate any matrix $\mathbf{A}$ with arbitrary precision. We recall this result in Theorem 4.1 as it is the starting point of our theoretical analysis.

Theorem 4.1 (Reformulation from Huhtanen & Perämäki (2015))**.** *For every matrix $\mathbf{M} \in \mathbb{C}^{n \times n}$, there exists a sequence of matrices $\mathbf{A}^{(1)} \dots \mathbf{A}^{(2n-1)}$ where $\mathbf{A}^{(i)}$ is a circulant matrix if i is odd, and a diagonal matrix otherwise, such that for any $\epsilon > 0$, we have*

$$\left\| \mathbf{A}^{(1)} \dots \mathbf{A}^{(2n-1)} - \mathbf{M} \right\|_{\mathrm{F}} < \epsilon \ . \tag{4.1}$$

Unfortunately, this theorem is of little use to understand the expressive power of diagonal-circulant matrices when they are used in deep neural networks for two reasons:

1. the bound only depends on the dimension of the matrix $\mathbf{M}$, not on the matrix itself;

2. the theorem does not provide any insights regarding the expressive power of m diagonal-circulant factors when m is much lower than $2n - 1$ as it is the case in most practical scenarios we consider in this chapter.

In the following theorem, we improve the result of Huhtanen & Perämäki (2015) by expressing the number of factors required to approximate $\mathbf{M}$, *as a function of the rank of* $\mathbf{M}$. This is useful when one deals with low-rank matrices, which is common in machine learning problems. Note that in this chapter, our results hold for complex

matrices. This is due to the fact that they are based on Theorem 4.1 which holds for complex matrices.

Theorem 4.2 (Rank-based diagonal-circulant decomposition). *Let $\mathbf{M} \in \mathbb{C}^{n\times n}$ be a matrix of rank at most k. Assume that n can be divided by k. There exists a sequence of $4k+1$ matrices $\mathbf{A}^{(1)} \dots \mathbf{A}^{(4k+1)}$, where $\mathbf{A}^{(i)} \in \mathbb{C}^{n\times n}$ is a circulant matrix if i is odd, and a diagonal matrix otherwise, such that for any $\epsilon > 0$, we have*

$$\left\| \mathbf{A}^{(1)} \dots \mathbf{A}^{(4k+1)} - \mathbf{M} \right\|_{\mathrm{F}} < \epsilon \ . \tag{4.2}$$

Proof of Theorem 4.2. Let $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$ be the SVD decomposition of $\mathbf{M}$ where $\mathbf{U}, \mathbf{V}$ and $\mathbf{\Sigma}$ are $n \times n$ matrices. Because $\mathbf{M}$ is of rank k, the last $n-k$ columns of $\mathbf{U}$ and $\mathbf{V}$ are null. In the following, we will first decompose $\mathbf{U}$ into a product of matrices $\mathbf{WRO}$, where $\mathbf{R}$ and $\mathbf{O}$ are respectively circulant and diagonal matrices, and $\mathbf{W}$ is a matrix which will be further decomposed into a product of diagonal and circulant matrices. Then, we will apply the same decomposition technique to $\mathbf{V}$. Ultimately, we will get a product of $4k+1$ matrices alternatively diagonal and circulant.

Let $\mathbf{R} = \mathrm{circ}(r_1 \dots r_n)$. Let $\mathbf{O}$ be an $n \times n$ diagonal matrix where $(\mathbf{O})_{i,i} = 1$ if $i \leq k$ and 0 otherwise. The k first columns of the product $\mathbf{RO}$ will be equal to that of $\mathbf{R}$, and the $n-k$ last columns of $\mathbf{RO}$ will be zeros. For example, if $k = 2$, we have:

$$\mathbf{RO} = \begin{pmatrix} r_1 & r_n & 0 & \cdots & 0 \\ r_2 & r_1 & & & \\ r_3 & r_2 & \vdots & & \vdots \\ \vdots & \vdots & & & \\ r_n & r_{n-1} & 0 & \cdots & 0 \end{pmatrix} \tag{4.3}$$

Let us define k diagonal matrices $\mathbf{D}^{(i)} = \mathrm{diag}(d_1^{(i)} \dots d_n^{(i)})$ for $i \in [k]$. For now, the values of $d_j^{(i)}$ are unknown, but we will show how to compute them. Let $\mathbf{Z}_{1,k}$ and $\mathbf{Z}_{1,n}$ be 1-unit-circulant matrix respectively of size $k \times k$ and $n \times n$ as defined in Section 2.1 and let $\mathbf{W} = \sum_{i=1}^{k} \mathbf{D}^{(i)} \mathbf{Z}_{1,n}^{i-1}$. Note that the $n-k$ last columns of the product $\mathbf{WRO}$ will be zeros. For example, with $k = 2$, we have:

$$\mathbf{W} = \begin{pmatrix} d_1^{(1)} & & & & d_1^{(2)} \\ d_2^{(2)} & d_2^{(1)} & & & \\ & d_3^{(2)} & \ddots & & \\ & & \ddots & \ddots & \\ & & & d_n^{(2)} & d_n^{(1)} \end{pmatrix} \tag{4.4}$$

$$\mathbf{WRO} = \begin{pmatrix} r_1 d_1^{(1)} + r_n d_1^{(2)} & r_n d_1^{(1)} + r_{n-1} d_1^{(2)} & 0 & \cdots & 0 \\ r_2 d_2^{(1)} + r_1 d_2^{(2)} & r_1 d_2^{(1)} + r_n d_2^{(2)} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ r_n d_n^{(1)} + r_{n-1} d_n^{(2)} & r_{n-1} d_n^{(1)} + r_{n-2} d_n^{(2)} & 0 & \cdots & 0 \end{pmatrix} \tag{4.5}$$

We want to find the values of $d_j^{(i)}$ such that $\mathbf{WRO} = \mathbf{U}$. We can formulate this as a linear equation system. In case $k = 2$, we get:

$$\begin{pmatrix} r_n & r_1 & & & & & & \\ r_{n-1} & r_n & & & & & & \\ & & r_1 & r_2 & & & & \\ & & r_n & r_1 & & & & \\ & & & & r_2 & r_3 & & \\ & & & & r_1 & r_2 & & \\ & & & & & & \ddots & \\ & & & & & & & \ddots \end{pmatrix} \times \begin{pmatrix} d_1^{(2)} \\ d_1^{(1)} \\ d_2^{(2)} \\ d_2^{(1)} \\ d_3^{(2)} \\ d_3^{(1)} \\ \vdots \\ \vdots \end{pmatrix} = \begin{pmatrix} (\mathbf{U})_{0,0} \\ (\mathbf{U})_{0,1} \\ (\mathbf{U})_{1,0} \\ (\mathbf{U})_{1,1} \\ \\ \\ \vdots \\ \\ \end{pmatrix} \tag{4.6}$$

The ith block of this block-diagonal matrix is a Toeplitz matrix induced by a contiguous subsequence of length $k+1$ of $(r_1, \ldots r_n, r_1 \ldots r_n)$. Set $r_j = 1$ for all $j \in \{k, 2k, 3k, \ldots n\}$ and set $r_j = 0$ for all other values of j. Then it is easy to see that each block is equal to $\mathbf{Z}_{1,k}^{\alpha}$ for some α. Note that the matrices $\mathbf{Z}_{1,k}^{\alpha}$ are invertible. This entails that the block diagonal matrix above is also invertible. So by solving this set of linear equations, we could find $d_1^{(1)} \ldots d_n^{(k)}$ such that $\mathbf{WRO} = \mathbf{U}$. We can apply the same idea to factorize $\mathbf{V} = \mathbf{W'RO}$ for some matrix $\mathbf{W'}$. Finally, we get

$$\mathbf{A} = \mathbf{U\Sigma V}^* = \mathbf{WRO\Sigma O}^*\mathbf{R}^*\mathbf{W}'^* \tag{4.7}$$

Note that the matrix $\mathbf{WR}$ can be decomposed as follows:

$$\mathbf{WR} = \left(\sum_{i=1}^{k} \mathbf{D}^{(i)} \mathbf{Z}_{1,n}^{(i-1)}\right) \mathbf{R} \tag{4.8}$$

where the last matrix $\mathbf{Z}_{1,n}^{(k-1)}\mathbf{R}$ is a circulant matrix because both matrices are circulant (see Theorem 2.1). The same reasoning can be applied with the matrix $\mathbf{R}^*\mathbf{W}'^*$. Therefore, by construction, the matrices $\mathbf{WR}$ and $\mathbf{R}^*\mathbf{W}'^*$ can both be factorized by $2k$ circulant and diagonal matrices. Also note that $\mathbf{O\Sigma O}^*$ is a diagonal matrix, because $\mathbf{O}$ and $\mathbf{\Sigma}$ are diagonal matrices. Overall, $\mathbf{A}$ can be represented with a product of $4k + 1$ matrices, alternatively diagonal and circulant. ■

A direct consequence of Theorem 4.2, is that if the number of diagonal-circulant factors is set to a value K, we can represent all linear transforms $\mathbf{M}$ whose rank is $\frac{K-1}{4}$. Compared to Huhtanen & Perämäki (2015), this result shows that structured matrices with fewer than $2n$ diagonal-circulant matrices (as it is the case in practice) can still represent a large class of matrices.

In the following section, we will analyze the expressivity of neural networks based on diagonal and circulant matrices. In order to characterize the expressivity, we will decompose the matrices of a dense neural network with diagonal and circulant matrices based on Theorem 4.2.

4.3 Analysis of Diagonal Circulant Neural Networks

Zhao et al. (2017) have shown that circulant networks with 2 layers and unbounded width are universal approximators. However, results on unbounded networks offer weak guarantees and two important questions have remained open until now:

1. *Can we approximate any function with a bounded-width diagonal-circulant network?*

2. *What function can we approximate with a diagonal-circulant neural network that has a bounded width and a small depth?*

We answer these two questions in this section. First, we present two lemmas that establish a link between the matrix decomposition presented in Theorem 4.2 and DCNNs and allow us to present our answer to the first question (Corollary 4.1). Then,

we analyze the expressive power of small depth diagonal-circulant neural networks by comparing them to dense neural networks. As in the previous section, we still work in the complex domain. Therefore, we need to extend the definition of neural networks to the complex domain. First, let us introduce an extension of the ReLU function.

Definition 4.1 (Complex ReLU function Trabelsi et al. (2018))**.** *Let us define the complex* ReLU *function* $\rho : \mathbb{C}^n \to \mathbb{C}^n$ *by:* $\rho(\mathbf{z}) = \max(0, \Re(\mathbf{z})) + \mathbf{i}\max(0, \Im(\mathbf{z}))$

Definition 4.2 (Dense Neural Network)**.** *Given a depth* $p \in \mathbb{N}$*, let us define* $\Omega = \left\{\left(\mathbf{W}^{(i)}, \mathbf{b}^{(i)}\right)\right\}_{i\in[p]}$ *a set of weights matrices and bias vectors such that* $\mathbf{W}^{(i)} \in \mathbb{C}^{n\times n}$ *and* $\mathbf{b}^{(i)} \in \mathbb{C}^n$*. Let* $\mathcal{X} \subset \mathbb{C}^n$ *and* $\mathcal{Y} \subset \mathbb{C}^n$ *be the input space and output space respectively. A dense neural network is a function* $N_\Omega : \mathcal{X} \to \mathcal{Y}$ *such that*

$$N^\rho_\Omega(\mathbf{x}) \triangleq \phi^\rho_{\mathbf{W}^{(p)},\mathbf{b}^{(p)}} \circ \cdots \circ \phi^\rho_{\mathbf{W}^{(1)},\mathbf{b}^{(1)}}(\mathbf{x}) \tag{4.9}$$

where ρ *is the complex* ReLU *function,* $\phi^\rho_{\mathbf{W}^{(i)},\mathbf{b}^{(i)}} : \mathbb{C}^n \to \mathbb{C}^n$ *is a layer parameterized by the weight matrix* $\mathbf{W}^{(i)}$ *and the bias vector* $\mathbf{b}^{(i)}$*, which can be expressed as follows:*

$$\phi^\rho_{\mathbf{W}^{(i)},\mathbf{b}^{(i)}}(\mathbf{x}) \triangleq \rho\left(\mathbf{W}^{(i)}\mathbf{x} + \mathbf{b}^{(i)}\right) \ , \tag{4.10}$$

Definition 4.3 (Diagonal-Circulant Neural Network)**.** *Given a depth* $p \in \mathbb{N}$*, let us define* $\Pi = \left\{\left(\mathbf{D}^{(i)}, \mathbf{C}^{(i)}, \mathbf{b}^{(i)}\right)\right\}_{i\in[p]}$ *a set of weight matrices and bias vectors such that* $\mathbf{D}^{(i)} \in \mathbb{C}^{n\times n}$ *is diagonal,* $\mathbf{C}^{(i)} \in \mathbb{C}^{n\times n}$ *is circulant and* $\mathbf{b}^{(i)} \in \mathbb{C}^n$*. Let* $\mathcal{X} \subset \mathbb{C}^n$ *and* $\mathcal{Y} \subset \mathbb{C}^n$ *be the input space and output space respectively. Let us denote the product of* $\mathbf{D}^{(i)}$ *and* $\mathbf{C}^{(i)}$ *by* $\mathbf{DC}^{(i)}$*. A diagonal-circulant neural network is a function* $N_\Pi : \mathcal{X} \to \mathcal{Y}$ *such that*

$$N^\rho_\Pi(\mathbf{x}) \triangleq \phi^\rho_{\mathbf{DC}^{(p)},\mathbf{b}^{(p)}} \circ \cdots \circ \phi^\rho_{\mathbf{DC}^{(1)},\mathbf{b}^{(1)}}(\mathbf{x}) \tag{4.11}$$

where $\phi^\rho_{\mathbf{DC}^{(i)},\mathbf{b}^{(i)}} : \mathbb{C}^n \to \mathbb{C}^n$ *is a layer parameterized by the weight matrix* $\mathbf{DC}^{(i)}$*, the bias vector* $\mathbf{b}^{(i)}$ *and can be expressed as follows:*

$$\phi^\rho_{\mathbf{DC}^{(i)},\mathbf{b}^{(i)}}(\mathbf{x}) \triangleq \rho\left(\mathbf{DC}^{(i)}\mathbf{x} + \mathbf{b}^{(i)}\right) \ , \tag{4.12}$$

where ρ *is the complex* ReLU *function.*

Diagonal-circulant neural networks are compact due to the layer being parameterized by diagonal and circulant matrices. Indeed, diagonal and circulant matrices of

size $n \times n$ can be represented with only n values. Therefore, the layer $\phi^{\rho}_{\mathbf{DC}^{(i)},\mathbf{b}^{(i)}}$ is parameterized by $3n$ complex values.

Diagonal-circulant neural networks can have more parameters than a dense neural networks but their depth need to be scaled accordingly. Let p_1 and p_2 be the depth of a dense neural network and a diagonal-circulant neural network respectively, then p_2 needs to be higher than $p_1 \frac{n+1}{3}$ to have more parameters than the dense network.

4.3.1 From Matrix Decomposition to Neural Networks

The purpose of this section is to extend the matrix decomposition presented in Theorem 4.2 to neural networks (Lemma 4.1) and show that bounded-width diagonal-circulant neural networks can approximate any dense neural network (Lemma 4.2).

Lemma 4.1. *Let $\mathbf{W}^{(1)} \dots \mathbf{W}^{(p)} \in \mathbb{C}^{n\times n}$, $\mathbf{b} \in \mathbb{C}^n$ and let $\mathcal{X} \subset \mathbb{C}^n$ be a bounded set. There exists $\mathbf{c}^{(1)} \dots \mathbf{c}^{(p)} \in \mathbb{C}^n$ such that for all $\mathbf{x} \in \mathcal{X}$ we have*

$$\rho\Big(\mathbf{W}^{(p)} \dots \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}\Big) = \phi^{\rho}_{\mathbf{W}^{(p)},\mathbf{c}^{(p)}} \circ \dots \circ \phi^{\rho}_{\mathbf{W}^{(1)},\mathbf{c}^{(1)}}(\mathbf{x}) \tag{4.13}$$

where $\phi^{\rho}_{\mathbf{W}^{(i)},\mathbf{c}^{(i)}} = \rho(\mathbf{W}^{(i)}\mathbf{x} + \mathbf{c}^{(i)})$ and ρ is the complex ReLU *function.*

Proof of Lemma 4.1. Let $\boldsymbol{\mathcal{W}}(j) = \prod_{k=1}^{j} \mathbf{W}^{(k)}$ and let us define the following set:

$$\mathcal{S} = \{(\boldsymbol{\mathcal{W}}(j)\mathbf{x})_t \mid \ \mathbf{x} \in \mathcal{X}, t \in [n], j \in [p]\} \tag{4.14}$$

and let $\Xi = \max\{|\Re(v)| : v \in \mathcal{S}\} + \mathbf{i}\max\{|\Im(v)| : v \in \mathcal{S}\}$. Intuitively, the real and imaginary parts of Ξ are the largest any activation in the network can have. Define $\psi_{\mathbf{W}^{(i)},\mathbf{c}^{(i)}}(\mathbf{x}) = \mathbf{W}^{(i)}\mathbf{x} + \mathbf{c}^{(i)}$. Let $\mathbf{c}^{(1)} = \Xi\mathbf{1}_n$. Clearly, for all $\mathbf{x} \in \mathcal{X}$ we have $\psi_{\mathbf{W}^{(1)},\mathbf{c}^{(1)}}(\mathbf{x}) \geq 0$, so $\rho(\psi_{\mathbf{W}^{(1)},\mathbf{c}^{(1)}}(\mathbf{x})) = \psi_{\mathbf{W}^{(1)},\mathbf{c}^{(1)}}(\mathbf{x})$ where ρ is the complex ReLU function. More generally, for all $j < p-1$ define $\mathbf{c}^{(j+1)} = \mathbf{1}_n\Xi - \mathbf{W}^{(j+1)}\mathbf{c}^{(j)}$. It is easy to see that for all $j < p$ we have

$$\psi_{\mathbf{W}^{(j)},\mathbf{c}^{(j)}} \circ \dots \circ \psi_{\mathbf{W}^{(1)},\mathbf{c}^{(1)}}(\mathbf{x}) = \boldsymbol{\mathcal{W}}(j)\mathbf{x} + \mathbf{1}_n\Xi \ . \tag{4.15}$$

This guarantees that for all $j < p$,

$$\psi_{\mathbf{W}^{(j)},\mathbf{c}^{(j)}} \circ \dots \circ \psi_{\mathbf{W}^{(1)},\mathbf{c}^{(1)}}(\mathbf{x}) = \rho \circ \psi_{\mathbf{W}^{(j)},\mathbf{c}^{(j)}} \circ \dots \circ \rho \circ \psi_{\mathbf{W}^{(1)},\mathbf{c}^{(1)}}(\mathbf{x}) \ . \tag{4.16}$$

Finally, define $\mathbf{c}^{(p)} = \mathbf{b} - \mathbf{W}^{(p)}\mathbf{c}^{(p-1)}$. We have,

$$\rho \circ \psi_{\mathbf{W}^{(p)},\mathbf{c}^{(p)}} \circ \ldots \circ \rho \circ \psi_{\mathbf{W}^{(1)},\mathbf{c}^{(1)}}(\mathbf{x}) = \rho(\boldsymbol{\mathcal{W}}(p)\mathbf{x} + \mathbf{b}) \ , \tag{4.17}$$

which concludes the proof. ■

The following lemma is our first result on the expressivity of diagonal-circulant neural networks. It states that a diagonal-circulant neural network with bounded width and depth can approximate any dense neural network. To prove this result, we use the matrix decomposition from Theorem 4.1 and Lemma 4.1 to decompose the dense matrices of the layers of a dense network and unfold it.

Lemma 4.2. *Let N_Ω be a dense neural network of width n and depth p, and let $\mathcal{X} \subset \mathbb{C}^n$ be a bounded set. There exists a diagonal-circulant neural network N_Π of width n and of depth $(2n-1)p$ such that for any $\epsilon > 0$, we have*

$$\|N_\Omega(\mathbf{x}) - N_\Pi(\mathbf{x})\|_2 < \epsilon, \quad \forall \mathbf{x} \in \mathcal{X} \ . \tag{4.18}$$

Proof of Lemma 4.2. Let us assume $N_\Omega = \phi_{\mathbf{W}^{(p)},\mathbf{b}^{(p)}} \circ \ldots \circ \phi_{\mathbf{W}^{(1)},\mathbf{b}^{(1)}}$. By Theorem 4.1, for any $\epsilon' > 0$, any matrix $\mathbf{W}^{(i)}$, there exists a sequence of $2n-1$ diagonal, $\{\mathbf{D}^{(i,j)}\}_{i\in[p],j\in[2n-1]}$, and circulant matrices, $\{\mathbf{C}^{(i,j)}\}_{i,\in[p],j\in[2n-1]}$, such that for all $i \in [p]$,

$$\left\| \prod_{j=1}^{2n-1} \mathbf{D}^{(i,2n-j)}\mathbf{C}^{(i,2n-j)} - \mathbf{W}^{(i)} \right\|_{\mathrm{F}} < \epsilon' \ . \tag{4.19}$$

For simplicity, let us denote the product of the two matrices $\mathbf{D}^{(i,j)}\mathbf{C}^{(i,j)}$ by $\mathbf{DC}^{(i,j)}$. By Lemma 4.1, we know that there exists a sequence of bias vectors $\left\{\mathbf{c}^{(i,j)}\right\}_{i\in[p],j\in[2n-1]}$ such that for all $i \in [p]$,

$$\phi^{\rho}_{\mathbf{DC}^{(i,2n-1)},\mathbf{c}^{(i,2n-1)}} \circ \ldots \circ \phi^{\rho}_{\mathbf{DC}^{(i,1)},\mathbf{c}^{(i,1)}}(\mathbf{x}) = \rho\Big(\mathbf{DC}^{(i,2n-1)} \ldots \mathbf{DC}^{(i,1)}\mathbf{x} + \mathbf{b}^{(i)}\Big). \tag{4.20}$$

Now if ϵ' tends to zero,

$$\left\| \phi^{\rho}_{\mathbf{DC}^{(i,2n-1)},\mathbf{c}^{(i,2n-1)}} \circ \ldots \circ \phi^{\rho}_{\mathbf{DC}^{(i,1)},\mathbf{c}^{(i,1)}} - \rho\Big(\mathbf{W}^{(i)}\mathbf{x} + \mathbf{b}^{(i)}\Big) \right\|_2 \tag{4.21}$$

will also tend to zero for any $\mathbf{x} \in \mathcal{X}$, because the ReLU function is continuous and $\mathcal{X}$ is bounded. Let $N_\Pi = \phi^\rho_{\mathbf{DC}^{(p,2n-1)},\mathbf{c}^{(p,2n-1)}} \circ \ldots \circ \phi^\rho_{\mathbf{DC}^{(1,1)},\mathbf{c}^{(1,1)}}$, because all functions are continuous, for all $\mathbf{x} \in \mathcal{X}$, $\|N_\Omega(\mathbf{x}) - N_\Pi(\mathbf{x})\|_2$ tends to zero as ϵ' tends to zero which concludes the proof. ■

Now that we know that diagonal-circulant neural networks can approximate any dense neural networks with arbitrary precision, we can extend is result to any function, thus demonstrating that they are universal approximators. First, let us present universal approximation results for neural networks. Cybenko (1989) and Hornik et al. (1989) have shown that neural networks with a single hidden layer and sigmoid activation can approximate any function if the hidden layer is allowed to be arbitrary large. However, arbitrary large neural networks lack practical applications.

More recently, the universal approximation results have been extended to bounded width neural networks with arbitrary depth (Hanin, 2017; Lu et al. 2017). More formally, we have the following result for neural networks with ReLU activations:

Theorem 4.3 (Universal Approximation Theorem for Neural Network Hanin (2017))**.** *For any continuous function $f : [0,1]^n \to \mathbb{R}_+$ of bounded supremum norm, for any $\epsilon > 0$, there exists a neural network N_Ω parameterized by Ω with an input layer of width n, an output layer of width 1, hidden layers of width $n+3$ and ReLU activations such that*

$$\forall x \in [0,1]^n, \quad |f(\mathbf{x}) - N_\Omega(\mathbf{x})| < \epsilon \ . \tag{4.22}$$

From Lemmas 4.1 and 4.2 and Theorem 4.3 by Hanin (2017) which states that dense neural networks are universal approximators, we can prove that bounded-width diagonal-circulant neural networks are also universal approximators.

Corollary 4.1. *Diagonal circulant neural networks with bounded width are universal approximators in the following sense: for any continuous function $f : [0,1]^n \to \mathbb{R}_+$ of bounded supremum norm, for any $\epsilon > 0$, there exists a complex-valued diagonal-circulant neural network N_Π of width $n+3$ such that $\forall \mathbf{x} \in [0,1]^{n+3}$, $|f(\mathbf{x}_1 \ldots \mathbf{x}_n) - (N_\Pi(\mathbf{x}))_0| < \epsilon$ where $|\cdot|$ refer to the complex modulus.*

Proof of Corollary 4.1. From Theorem 4.3, we know that there exists a dense neural network N_Ω with an input layer of width n, an output layer of width 1, hidden layers of width $n+3$ and ReLU activations such that $\forall \mathbf{x} \in [0,1]^n, |f(\mathbf{x}) - N_\Omega(\mathbf{x})| < \epsilon$. From N_Ω, we can easily build a dense neural networks $\widetilde{N}_\Omega$ of width exactly $n+3$, such that $\forall \mathbf{x} \in [0,1]^{n+3}$, $\left| f(\mathbf{x}_1 \dots \mathbf{x}_n) - \left(\widetilde{N}_\Omega(\mathbf{x})\right)_0 \right| < \epsilon$. Thanks to Lemma 4.2, this last network can be approximated arbitrarily well by a diagonal-circulant neural network of width $n+3$. Note that the matrices in the diagonal-circulant neural network are complex, even though we are approximating a real-valued function. ■

The previous result shows that diagonal-circulant neural networks are universal approximators of real-valued functions. However the depth needed is in $\mathcal{O}(n)$ where n is the width of the network (size of the input). The depth needed to reach universal approximation is not small, in our experiments, n can be over 3000. Nonetheless, Cheng et al. (2015) have provided empirical evidence that diagonal-circulant neural networks with small depth can offer good performance. In the following subsection, we study the theoretical expressivity of diagonal-circulant neural networks with bounded-width and small depth. This study allows us to better understand why DCNNs show good empirical performances with limited depth.

4.3.2 The Expressive Power of Diagonal-Circulant Neural Networks

In this subsection, we study the expressive power of diagonal-circulant neural networks with small depth. To assess the expressivity of DCNNs, we compare the depth needed to approximate dense neural networks with low *total rank* which we define as the sum of ranks of each weights matrix. With the concept of total rank, we present in the following, our result on the expressive power of DCNNs with respect to the total rank of dense neural networks.

Definition 4.4 (Total Rank)**.** *The total rank $\kappa(N_\Omega)$ of the neural network N_Ω corresponds to the sum of the ranks of the matrices $\mathbf{W}^{(1)} \dots \mathbf{W}^{(p)}$ as follows*

$$\kappa(N_\Omega) \triangleq \sum_{i \in [p]} \operatorname{rank}(\mathbf{W}^{(i)}) \ . \tag{4.23}$$

Theorem 4.4 (Rank-based expressive power of DCNNs). *Let N_Ω be a dense neural network of width n, depth p and a total rank K, and assume n is a power of 2. Let $\mathcal{X} \subset \mathbb{C}^n$ be a bounded set. Then, for any $\epsilon > 0$, there exists a diagonal-circulant neural network N_Π of width n such that $\|N_\Omega(\mathbf{x}) - N_\Pi(\mathbf{x})\|_2 < \epsilon$ for all $\mathbf{x} \in \mathcal{X}$ and the depth of N_Π is bounded by $5K$.*

Proof of Theorem 4.4. Let N_Ω be a dense neural networks parameterized by $\Omega = \{(\mathbf{W}^{(i)}, \mathbf{b}^{(i)}\}_{i\in[p]}$ of width n, depth p. Let $k^{(1)} \dots k^{(p)}$ be the ranks of matrices $\mathbf{W}^{(1)} \dots \mathbf{W}^{(p)}$, which are $n \times n$ matrices. Assume that $\forall i$, n can be divided by k^i. By Theorem 4.2, for any $\epsilon > 0$, any matrix $\mathbf{W}^{(i)}$ of rank $k^{(i)}$, there exists a sequence of diagonal matrices $\{\mathbf{D}^{(i,j)}\}_{i\in[p],j\in[4k^i+1]}$ and circulant matrices, $\{\mathbf{C}^{(i,j)}\}_{i,\in[p],j\in[4k^i+1]}$, such that for all $i \in [p]$,

$$\left\| \prod_{j=1}^{4k^i+1} \mathbf{D}^{(i,4k^i+2-j)} \mathbf{C}^{(i,4k^i+2-j)} - \mathbf{W}^{(i)} \right\|_{\mathrm{F}} < \epsilon' \ . \tag{4.24}$$

Using the exact same technique as in Lemma 4.2, we can build a diagonal-circulant neural network N_Π, such that

$$\|N_\Omega(\mathbf{x}) - N_\Pi(\mathbf{x})\|_2 < \epsilon, \quad \forall \mathbf{x} \in \mathcal{X}, \tag{4.25}$$

for which the total number of layers is bounded as follows:

$$\sum_{i\in[p]} \left(4k^{(i)} + 1\right) \leq p + 4 \sum_{i\in[p]} k^{(i)} \leq p + 4\kappa(N_\Omega) \leq 5\kappa(N_\Omega) \ . \tag{4.26}$$

where $\kappa(N_\Omega)$ is the total rank of the dense neural network N_Ω. ■

Remark that in the theorem, we require that n is a power of 2. We conjecture that the result still holds even without this condition. This result refines Lemma 4.2 and answers our second question: does DCNN of bounded width and small depth can approximate a dense neural network of low total rank? Note that the converse is not true because an $n \times n$ circulant matrix can be of full rank, therefore approximating a DCNN of depth 1 can require a dense network of total rank equal to n.

Finally, what if we choose to use diagonal-circulant networks with a small depth to approximate a dense neural network whose matrices are not of lower rank? To

answer this question, we present three results. First, we characterize the negative impact of replacing matrices by their low rank approximation. Then, we extend this result to neural networks and bound the error between a dense neural network with full total rank and one with low total rank. Finally, Corollary 4.2 presents our result which bounds the error between a dense neural network with full total rank and a diagonal-circulant neural network.

Lemma 4.3. *Let $\mathbf{W} \in \mathbb{C}^{n\times n}$ with singular values $\sigma_1 \dots \sigma_n$, and let $\mathbf{b}, \mathbf{x}, \mathbf{y} \in \mathbb{C}^n$. Let $\widetilde{\mathbf{W}}$ be the matrix obtained by an SVD approximation of rank k of matrix $\mathbf{W}$. Then we have:*

$$\left\|\rho(\mathbf{W}\mathbf{x}+\mathbf{b}) - \rho(\widetilde{\mathbf{W}}\mathbf{y}+\mathbf{b})\right\|_2 \leq \sigma_1\|\mathbf{x}-\mathbf{y}\|_2 + \sigma_{k+1}\|\mathbf{x}\|_2 \tag{4.27}$$

Proof of Lemma 4.3. Let us denote σ_j be the jth singular value of $\mathbf{W}$ and recall that $\sigma_1(\mathbf{W}) = \|\mathbf{W}\|_2$ by the definition of the spectral norm. Furthermore, we have $\sigma_1(\mathbf{W}) = \sigma_1(\widetilde{\mathbf{W}})$ because the greatest singular values are equal for both $\mathbf{W}$ and $\widetilde{\mathbf{W}}$. Also, note that $\left\|\mathbf{W} - \widetilde{\mathbf{W}}\right\|_2 = \sigma_{k+1}$. First, let us bound the formula without ReLUs:

$$\begin{aligned}
\left\|(\mathbf{W}\mathbf{x}+\mathbf{b}) - (\widetilde{\mathbf{W}}\mathbf{y}+\mathbf{b})\right\|_2 &= \left\|\mathbf{W}\mathbf{x} - \widetilde{\mathbf{W}}\mathbf{x} - \widetilde{\mathbf{W}}(\mathbf{y}-\mathbf{x})\right\|_2 && (4.28)\\
&\leq \left\|(\mathbf{W} - \widetilde{\mathbf{W}})\mathbf{x}\right\|_2 + \left\|\widetilde{\mathbf{W}}\right\|_2\|\mathbf{x}-\mathbf{y}\|_2 && (4.29)\\
&\leq \|\mathbf{x}\|_2\sigma_{k+1} + \sigma_1\|\mathbf{x}-\mathbf{y}\|_2 && (4.30)
\end{aligned}$$

Finally, it is easy to see that for any pair of vectors $\mathbf{x}, \mathbf{y} \in \mathbb{C}^n$, we have

$$\|\rho(\mathbf{x}) - \rho(\mathbf{y})\|_2 \leq \|\mathbf{x}-\mathbf{y}\|_2 \ , \tag{4.31}$$

because the complex ReLU function is 1-Lipschitz. This concludes the proof. ■

The lemma above bound the error between a linear transform and its equivalent low rank approximation. In the following, we extend this result to neural networks.

Proposition 4.1. *Let $N_\Omega : \mathbb{C}^n \to \mathbb{C}^n$ be a dense neural network, with* ReLU *activation, parameterized by $\Omega = \left\{(\mathbf{W}^{(i)}, \mathbf{b}^{(i)})\right\}_{i\in[p]}$ with $\mathbf{W}^{(i)} \in \mathbb{C}^{n\times n}, \mathbf{b}^{(i)} \in \mathbb{C}^n$ for all $i \in [p]$ and $N_\Omega = \phi_{\mathbf{W}^{(p)},\mathbf{b}^{(p)}} \circ \dots \circ \phi_{\mathbf{W}^{(1)},\mathbf{b}^{(1)}}$ of depth p and width n. Let*

$\widetilde{\Omega} = \left\{ (\widetilde{\mathbf{W}}^{(i)}, \mathbf{b}^{(i)}) \right\}_{i \in [p]}$ where $\widetilde{\mathbf{W}}^{(i)}$ is the matrix obtained by the SVD approximation of rank k of matrix $\mathbf{W}^{(i)}$. Define the network $N_{\widetilde{\Omega}}$ and let $\sigma_j^{(i)}$ be the jth singular value of $\mathbf{W}^{(i)}$ and denote $\sigma_j^{(\max)} = \max_i \sigma_j^{(i)}$, the largest j^{th} singular value across layers. Then, for any $\mathbf{x} \in \mathbb{C}^n$, we have:

- *if $\sigma_1^{(\max)} = 1$:*

$$\left\| N_\Omega(\mathbf{x}) - N_{\widetilde{\Omega}}(\mathbf{x}) \right\|_2 \leq p\left(R\sigma_{k+1}^{(\max)} \right) \ . \tag{4.32}$$

- *if $\sigma_1^{(\max)} \neq 1$:*

$$\left\| N_\Omega(\mathbf{x}) - N_{\widetilde{\Omega}}(\mathbf{x}) \right\|_2 \leq \frac{\left((\sigma_1^{(\max)})^p - 1 \right) R\sigma_{k+1}^{(\max)}}{\sigma_1^{(\max)} - 1} \tag{4.33}$$

where R is an upper bound on the norm of the output of any layer in N_Ω.

Proof of Proposition 4.1. Let $\mathbf{x}^{(0)} \in \mathbb{C}^n$ and $\mathbf{y}^{(0)} = \mathbf{x}^{(0)}$. For all $i \in [p]$, define $\mathbf{x}^{(i)} = \rho\left(\mathbf{W}^{(i)}\mathbf{x}^{(i-1)} + \mathbf{b}^{(i)} \right)$ and $\mathbf{y}^{(i)} = \rho\left(\widetilde{\mathbf{W}}^{(i)}\mathbf{y}^{(i-1)} + \mathbf{b}^{(i)} \right)$. We aim to upper bound the difference in norm of $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(i)}$. First, let us consider the linear transform within $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(i)}$: The difference in norm between $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(i)}$ can be upper bounded as follows:

$$\begin{aligned} \left\| \mathbf{x}^{(i)} - \mathbf{y}^{(i)} \right\|_2 &\leq \sigma_1^{(i)} \left\| \mathbf{x}^{(i-1)} - \mathbf{y}^{(i-1)} \right\|_2 + \sigma_{k+1}^{(i)} \left\| \mathbf{x}^{(i-1)} \right\|_2 && (4.34) \\ &\leq \sigma_1^{(\max)} \left\| \mathbf{x}^{(i-1)} - \mathbf{y}^{(i-1)} \right\|_2 + \sigma_{k+1}^{(\max)} R && (4.35) \end{aligned}$$

where the first inequality stems from Lemma 4.3 and the second by setting $\sigma_j^{(\max)} = \max_i \sigma_j^{(i)}$ where $\sigma_j^{(\max)}$ is the largest j^{th} singular value across layers and $R = \max_i \left\| \mathbf{x}^{(i)} \right\|_2$. From there, we need to consider two cases:

- If $\sigma_1^{(\max)} = 1$: we have a recurrence relation of the form $a_n = a_{n-1} + s$ with $a_0 = 0$ which can unfold as follows: $a_n = ns$. We can apply this formula to bound our error as follows:

$$\left\| \mathbf{x}^{(p)} - \mathbf{y}^{(p)} \right\|_2 \leq p\left(R\sigma_{k+1}^{(\max)} \right) \ . \tag{4.36}$$

- If $\sigma_1^{(\max)} \neq 1$: we have a recurrence relation of the form $a_n = ra_{n-1} + s$ with $a_0 = 0$ which can unfold as follows: $a_n = \frac{s(r^n-1)}{r-1}$. We can apply this formula to bound our error as follows:

$$\left\|\mathbf{x}^{(p)} - \mathbf{y}^{(p)}\right\|_2 \leq \frac{\left((\sigma_1^{(\max)})^p - 1\right) R\sigma_{k+1}^{(\max)}}{\sigma_1^{(\max)} - 1} \quad , \tag{4.37}$$

which concludes the proof. ■

Proposition 4.1 bounds the error between a dense neural network and a neural network whose matrices are the low rank approximations of the first ones. Two cases are presented. If the largest singular value across the network is equal to 1, then the error is polynomial with the depth of the network. In the case where the largest singular value across the network is different from 1, the error is exponential with respect to the depth of the network.

Now, we can easily extend Proposition 4.1 to diagonal-circulant neural networks. By Theorem 4.4, we can replace the layers with low rank approximation by the product of diagonal and circulant matrices leading to a diagonal-circulant neural network with a higher depth.

Corollary 4.2. *Let N_Ω be a dense neural network of depth p and width n and parameterized by $\Omega = \left\{(\mathbf{W}^{(i)}, \mathbf{b}^{(i)})\right\}_{i\in[p]}$. Let $\sigma_1^{(i)}$ be the largest singular value of $\mathbf{W}^{(i)}$. Let $\mathcal{X} \subset \mathbb{C}^n$ be a bounded set. Let k be an integer dividing n. There exists a diagonal-circulant neural network N_Π of width n and of depth $m = (4k+1)p$, parameterized by $\Pi = \left\{(\mathbf{D}^{(i)}\mathbf{C}^{(i)}, \mathbf{c}^{(i)})\right\}_{i\in[m]}$ such that, for any $\mathbf{x} \in \mathbb{C}^n$, we have:*

- *if $\sigma_1^{(\max)} = 1$:*

$$\|N_\Omega(\mathbf{x}) - N_\Pi(\mathbf{x})\|_2 \leq p\left(R\sigma_{k+1}^{(\max)}\right) \quad . \tag{4.38}$$

- *if $\sigma_1^{(\max)} \neq 1$:*

$$\|N_\Omega(\mathbf{x}) - N_\Pi(\mathbf{x})\|_2 \leq \frac{\left((\sigma_1^{(\max)})^p - 1\right) R\sigma_{k+1}^{(\max)}}{\sigma_1^{(\max)} - 1} \quad , \tag{4.39}$$

where R is an upper bound on the norm of the output of any layer in N_Ω.

Proof of Corollary 4.2. Let $N_{\widetilde{\Omega}}$ be a dense neural network of depth p and width n and let $\widetilde{\Omega} = \left\{(\widetilde{\mathbf{W}}^{(i)}, \mathbf{b}^{(i)})\right\}_{i\in[p]}$ such that $\widetilde{\mathbf{W}}^{(i)}$ is the matrix obtained by the SVD approximation of rank k of matrix $\mathbf{W}^{(i)}$. With Proposition 4.1, we have an error bound on $\left\|N_\Omega(\mathbf{x}) - N_{\widetilde{\Omega}}(\mathbf{x})\right\|_2$. Now each matrix $\widetilde{\mathbf{W}}^{(i)}$ can be replaced by a product of $4k+1$ diagonal-circulant matrices. By Theorem 4.4, this product yields a diagonal-circulant neural network of depth $m = (4k+1)p$, strictly equivalent to $N_{\widetilde{\Omega}}$ on $\mathcal{X}$. This concludes the proof. ■

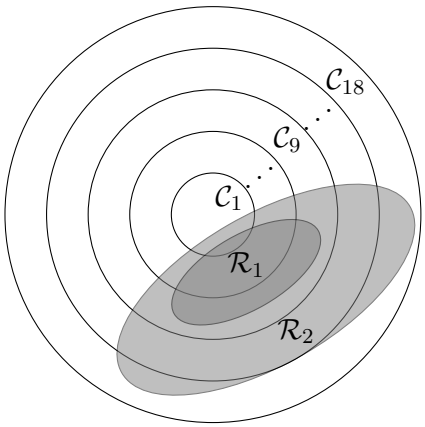


Figure 4.1: Illustration of the expressivity of diagonal-circulant neural networks.

We highlight the significance of these results with the two following properties.

Properties. *Let $\mathcal{R}_K$ be the set of all functions $N_\Omega : \mathbb{C}^n \to \mathbb{C}^n$ for all n, representable by a dense neural network with complex* ReLU *activation of total rank at most K and let $\mathcal{C}_p$ be the set of all functions $N_\Pi : \mathbb{C}^n \to \mathbb{C}^n$ for all n, representable by deep diagonal-circulant networks of depth at most p, then:*

$$\forall K, \exists p \quad \mathcal{R}_K \subsetneq \mathcal{C}_p \tag{4.40}$$

$$\forall p, \nexists K \quad \mathcal{C}_p \subseteq \mathcal{R}_K \tag{4.41}$$

We illustrate the meaning of these properties using Figure 4.1. As we can see, the set $\mathcal{R}_K$ of all the functions representable by a dense neural network of total rank K is strictly included in the set $\mathcal{C}_{9K}$ of all diagonal-circulant neural networks of depth $9K$ (as by Theorem 4.4). These properties are interesting for many reasons. First,

Equation (4.41) shows that diagonal-circulant networks are *strictly more expressive* than networks with low total rank. Second and most importantly, in standard deep neural networks, it is known that the most of the singular values are close to zero. This phenomenon is called *rank collapse* and it has been first observed by Saxe et al. (2013) and confirmed later by Sedghi et al. (2018) and Arora et al. (2019). Equation (4.40) shows that these networks can efficiently be approximated by diagonal-circulant networks. Finally, several publications have shown that neural networks can be trained explicitly to have low-rank weight matrices (Li & Shi, 2018; Goyal et al. 2019). This opens the possibility of learning compact and accurate diagonal-circulant networks.

4.4 How to Train Deep Diagonal Circulant Neural Networks?

Training diagonal-circulant neural networks has revealed to be a challenging problem. Indeed, as discussed earlier, the expressivity of diagonal-circulant neural networks scale with depth. In the following, we devise two techniques to facilitate the training of deep diagonal-circulant neural networks. First, we propose an initialization procedure which guarantees the signal is propagated across the network without vanishing nor exploding. Secondly, we study the behavior of DCNNs with different nonlinearity functions and determine the best parameters for different settings. Note that we choose to train diagonal-circulant neural networks with real matrices instead of complex ones. Indeed, the complex version of diagonal-circulant neural networks would have twice the number of parameters.

4.4.1 Initialization Scheme of Diagonal-Circulant Neural Networks

In order to facilitate the training of deep diagonal-circulant neural networks, we extend the Xavier initialization (Glorot & Bengio, 2010) which is an initialization scheme proposed for dense and convolutional neural networks. First, for each circulant matrix $\mathbf{C} = \text{circ}(\mathbf{c})$ with $\mathbf{c} \in \mathbb{R}^n$, each $\mathbf{c}_i$ is randomly drawn from $\mathcal{N}(0, \alpha^2)$, with $\alpha = \sqrt{\frac{2}{n}}$. Next, for each diagonal matrix $\mathbf{D} = \text{diag}(\mathbf{d})$ with $\mathbf{d} \in \mathbb{R}^n$, each $\mathbf{d}_i$ is drawn randomly and uniformly from $\{-1, 1\}$ for all i. Finally, all biases in the network are randomly drawn from $\mathcal{N}(0, \alpha'^2)$, for some small value of α'.

Lemma 4.4. *Let* $\mathbf{c}, \mathbf{d}, \mathbf{b}$ *be random variables in* $\mathbb{R}^n$ *such that* $\mathbf{c} \sim \mathcal{N}(0, \mathbf{I}_n\alpha^2)$, $\mathbf{b} \sim \mathcal{N}(0, \mathbf{I}_n\alpha'^2)$ *and* $\mathbf{d}_i \sim \{-1, 1\}, \forall i$ *uniformly. Define* $\mathbf{C} = \text{circ}(\mathbf{c})$ *and* $\mathbf{D} = \text{diag}(\mathbf{d})$

Define $\mathbf{y} = \mathbf{DCu} + \mathbf{b}$ *for some vector* $\mathbf{u}$ *in* $\mathbb{R}^n$*. Then, for all* i*, the probability density function (p.d.f.) of* $\mathbf{y}_i$ *is symmetric. Also:*

- *Assume* $\mathbf{u}_0, \ldots, \mathbf{u}_{n-1}$ *are fixed. Then, we have for* $i \neq i'$*:*

$$\mathrm{Cov}(\mathbf{y}_i, \mathbf{y}_{i'}) = 0 \quad \textit{and} \quad \mathrm{Var}(\mathbf{y}_i) = \alpha'^2 + \sum_j \mathbf{u}_j^2 \alpha^2 \tag{4.42}$$

- *Let* $\mathbf{x}$ *be random variables in* $\mathbb{R}^n$ *such that the p.d.f. of* $\mathbf{x}_i$ *is symmetric for all* i*, and let* $\mathbf{u}_i = \rho(\mathbf{x}_i)$*. We have for* $i \neq i'$ *:*

$$\mathrm{Cov}(\mathbf{y}_i, \mathbf{y}_{i'}) = 0 \quad \textit{and} \quad \mathrm{Var}(\mathbf{y}_i) = \alpha'^2 + \frac{1}{2} \sum_j \mathrm{Var}(\mathbf{x}_i) \alpha^2 \tag{4.43}$$

Proof of Lemma 4.4. By an abuse of notation, we write $\mathbf{c}_0 = \mathbf{c}_n, \mathbf{c}_{-1} = \mathbf{c}_{n-1}$ and so on. First, note that: $\mathbf{y}_i = \sum_{j=0}^{n-1} \mathbf{c}_{j-i} \mathbf{u}_j \mathbf{d}_j + \mathbf{b}_i$. Observe that the term $\mathbf{c}_{j-i} \mathbf{u}_j \mathbf{d}_j$ has symmetric p.d.f. because of $\mathbf{d}_j$. Thus, $\mathbf{y}_i$ has a symmetric p.d.f.. Now let us compute the covariance.

$$\mathrm{Cov}(\mathbf{y}_i, \mathbf{y}_{i'}) = \left(\sum_{j,j'=0}^{n-1} \mathrm{Cov}(\mathbf{c}_{j-i} \mathbf{u}_j \mathbf{d}_j, \mathbf{c}_{j'-i'} \mathbf{u}_{j'} \mathbf{d}_{j'}) \right) + \mathrm{Cov}(\mathbf{b}_i, \mathbf{b}_{i'}) \tag{4.44}$$

$$= \sum_{j,j'=0}^{n-1} \mathbb{E}[\mathbf{c}_{j-i} \mathbf{u}_j \mathbf{d}_j \mathbf{c}_{j'-i'} \mathbf{u}_{j'} \mathbf{d}_{j'}] - \mathbb{E}[\mathbf{c}_{j-i} \mathbf{u}_j \mathbf{d}_j] \mathbb{E}[\mathbf{c}_{j'-i'} \mathbf{u}_{j'} \mathbf{d}_{j'}] \tag{4.45}$$

Observe that $\mathbb{E}[\mathbf{c}_{j-i} \mathbf{u}_j \mathbf{d}_j] = \mathbb{E}[\mathbf{c}_{j-i} \mathbf{u}_j] \mathbb{E}[\mathbf{d}_j] = 0$ because $\mathbf{d}_j$ is independent from $\mathbf{c}_{j-i} \mathbf{u}_j$. Also, observe that if $j \neq j'$ then $\mathbb{E}[\mathbf{d}_j \mathbf{d}_{j'}] = 0$ and

$$\mathbb{E}[\mathbf{c}_{j-i} \mathbf{u}_j \mathbf{d}_j \mathbf{c}_{j'-i'} \mathbf{u}_{j'} \mathbf{d}_{j'}] = \mathbb{E}[\mathbf{d}_j \mathbf{d}_{j'}] \mathbb{E}[\mathbf{c}_{j-i} \mathbf{u}_j \mathbf{c}_{j'-i'} \mathbf{u}_{j'}] = 0 \ . \tag{4.46}$$

Therefore, the only non null terms are those for which $j = j'$. We get:

$$\mathrm{Cov}(\mathbf{y}_i, \mathbf{y}_{i'}) = \sum_{j=0}^{n-1} \mathbb{E}[\mathbf{c}_{j-i} \mathbf{u}_j \mathbf{d}_j \mathbf{c}_{j-i'} \mathbf{u}_j \mathbf{d}_j] \tag{4.47}$$

$$= \sum_{j=0}^{n-1} \mathbb{E}\left[\mathbf{c}_{j-i} \mathbf{c}_{j-i'} \mathbf{u}_j^2\right] \tag{4.48}$$

Assume $\mathbf{u}$ is a fixed vector. Then, $\mathrm{Var}(\mathbf{y}_i) = \sum_{j=0}^{n-1} \mathbf{u}_j^2 \alpha^2$ and $\mathrm{Cov}(\mathbf{y}_i, \mathbf{y}_{i'}) = 0$ for $i \neq i'$ because $\mathbf{c}_{j-i}$ is independent from $\mathbf{c}_{j-i'}$. Now assume that $\mathbf{u}_j = \rho(\mathbf{x}_j)$ where $\mathbf{x}_j$ is a random variables in $\mathbb{R}^n$. Clearly, $\mathbf{u}_j^2$ is independent from $\mathbf{c}_{j-i}$ and $\mathbf{c}_{j-i'}$, thus, we have:

$$\mathrm{Cov}(\mathbf{y}_i, \mathbf{y}_{i'}) = \sum_{j=0}^{n-1} \mathbb{E}\big[\mathbf{c}_{j-i}\mathbf{c}_{j-i'}\big]\mathbb{E}\Big[\mathbf{u}_j^2\Big] \ . \tag{4.49}$$

For $i \neq i'$, then $\mathbf{c}_{j-i}$ and $\mathbf{c}_{j-i'}$ are independent, we have $\mathbb{E}\big[\mathbf{c}_{j-i}c_{j-i'}\big] = \mathbb{E}[\mathbf{c}_{j-i}]\mathbb{E}\big[\mathbf{c}_{j-i'}\big] = 0$ and $\mathrm{Cov}(\mathbf{y}_i, \mathbf{y}_{i'}) = 0$ if $i \neq i'$. Let us compute the variance. We get $\mathrm{Var}(\mathbf{y}_i) = \sum_{j=0}^{n-1} \mathrm{Var}(\mathbf{c}_{j-i})\mathbb{E}\Big[\mathbf{u}_j^2\Big]$. Because the p.d.f. of $\mathbf{x}_j$ is symmetric, $\mathbb{E}\Big[\mathbf{x}_j^2\Big] = 2\mathbb{E}\Big[\mathbf{u}_j^2\Big]$ and $\mathbb{E}[\mathbf{x}_j] = 0$. Thus,

$$\mathrm{Var}(\mathbf{y}_i) = \frac{1}{2}\sum_{j=0}^{n-1} \mathrm{Var}(\mathbf{c}_{j-i})\mathbb{E}\Big[\mathbf{x}_j^2\Big] = \frac{1}{2}\sum_{j=0}^{n-1} \mathrm{Var}(\mathbf{c}_{j-i})\mathrm{Var}(\mathbf{x}_j) \tag{4.50}$$

which concludes the proof. ■

Now we can state our result on the initialization of diagonal-circulant neural networks. The following proposition states that the covariance matrix at the output of any layer in a diagonal-circulant neural network is constant. Moreover, note that the result of this proposition is independent of the depth of the network.

Proposition 4.2 (Initialization of Diagonal-Circulant Neural Networks)**.** *Let N_Π be a diagonal-circulant neural network of depth p initialized according to our procedure, with $\alpha' = 0$. Assume that all layers 1 to $p-1$ have* ReLU *activation functions, and that the last layer has the identity activation function. Then, for any $\mathbf{x} \in \mathbb{R}^n$, the covariance matrix of $N_\Pi(\mathbf{x})$ is $\frac{2}{n}\mathbf{I}_n\|\mathbf{x}\|_2^2$.*

Proof of Proposition 4.2. Let $N_\Pi \triangleq \phi_{\mathbf{D}^{(p)},\mathbf{C}^{(p)}} \circ \ldots \circ \phi_{\mathbf{D}^{(1)},\mathbf{C}^{(1)}}$ be a p layer diagonal-circulant neural network. All matrices are initialized as described in the statement of the proposition. Let $\mathbf{y} = \mathbf{D}^{(1)}\mathbf{C}^{(1)}\mathbf{x}$. Lemma 4.4 shows that $\mathrm{Cov}(\mathbf{y}_i, \mathbf{y}_{i'}) = 0$ for $i \neq i'$ and $\mathrm{Var}(\mathbf{y}_i) = \frac{2}{n}\|x\|_2^2$. For any $j \leq p$, define

$$\mathbf{z}^{(j)} = \phi_{\mathbf{D}^{(j)},\mathbf{C}^{(j)}} \circ \ldots \circ \phi_{\mathbf{D}^{(1)},\mathbf{C}^{(1)}}(\mathbf{x}) \ . \tag{4.51}$$

By a recursive application of Lemma 4.4, we get

$$\mathrm{Cov}(\mathbf{z}_i^{(j)}, \mathbf{z}_{i'}^{(j)}) = 0 \quad \text{and} \quad \mathrm{Var}(\mathbf{z}_i^{(j)}) = \frac{2}{n}\|\mathbf{x}\|_2^2 \tag{4.52}$$

which concludes the proof. ■

The effect of Proposition 4.2 is that the signal and the gradient will not vanish during the training, facilitating the convergence. The fact that the result of Proposition 4.2 is independent of the depth of the network allows us to train very deep diagonal-circulant neural networks.

4.4.2 Analysis of the Use of Nonlinearities

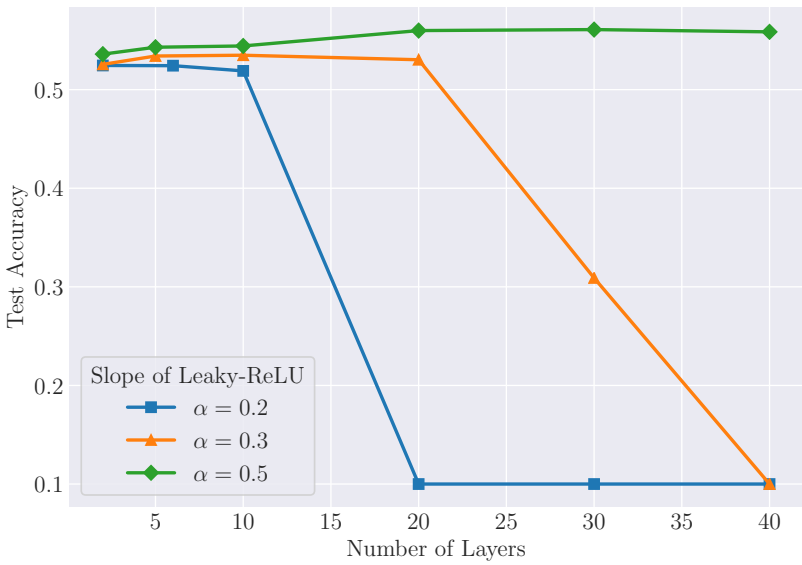


Figure 4.2: Impact of increasing the slope of a Leaky-ReLU in DCNNs.

We empirically found that the ReLU activations had an impact on the training of deep diagonal-circulant neural networks on CIFAR10 dataset. Indeed, the deeper the network, the more nonlinear it is, which makes convergence difficult. In an experiment, we replace the ReLU activations with Leaky-ReLU activations (*cf.* Section 2.2.2) and vary the slope of the Leaky-ReLU (a higher slope means an activation function that is closer to a linear function). The results of this experiment are presented in Figure 4.2. In this experiment, we try different slopes for the Leaky-ReLU activation and train

diagonal-circulant neural networks with different depth. We can observe that a higher slope (making the network more linear) facilitates convergence, allowing us to train deeper networks. This is an interesting result, since we can use this technique to adjust the number of parameters in the network (increasing depth), without facing training difficulties. We hence rely on this setting in the experimental section.

4.5 Experiments

This experimental section aims at answering the following questions:

Q1 How do DCNNs compare to other approaches such as ACDC, LDR or other structured approaches?

Q2 How do DCNNs compare to other compression based techniques?

Q3 How do DCNNs perform in the context of large-scale real-world machine learning applications?

4.5.1 Comparison with Other Structured Approaches (Q1)

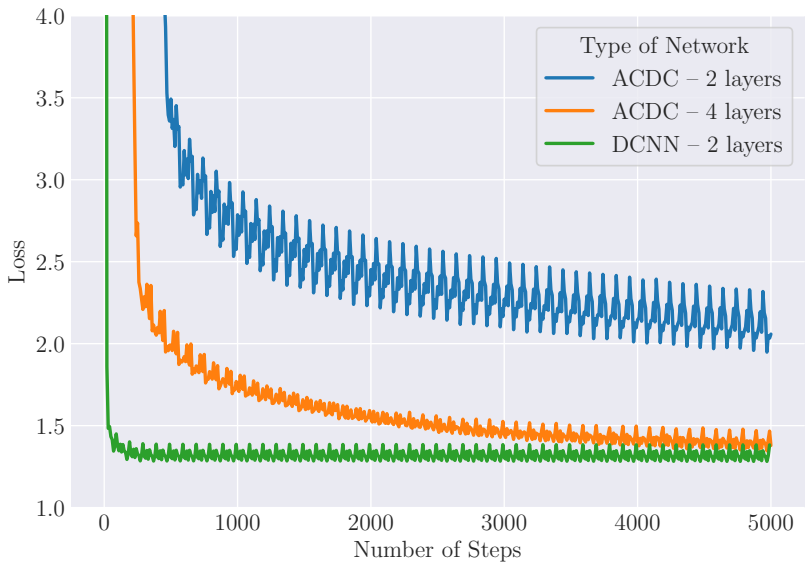


Figure 4.3: Comparison of the training loss of DCNNs and ACDC networks on a regression task with synthetic data.

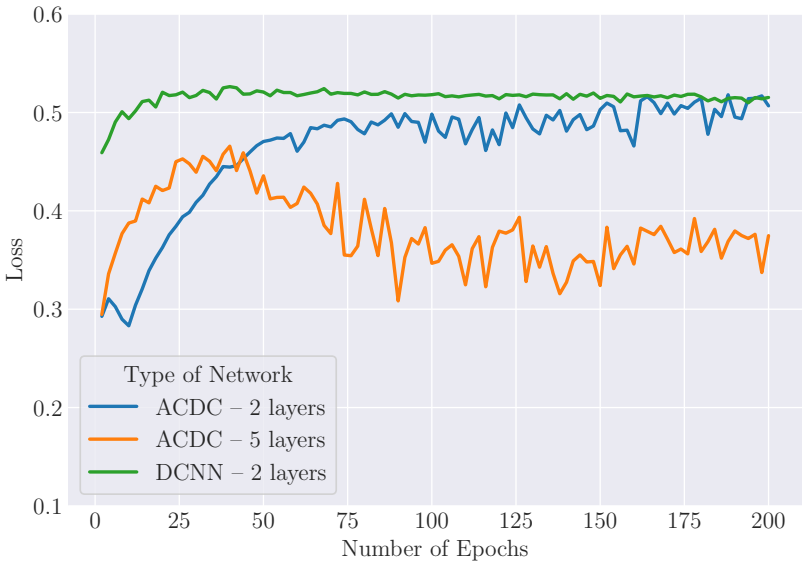


Figure 4.4: Comparison of the training loss of DCNNs and ACDC networks on a CIFAR-10 dataset.

Comparison with ACDC

In Chapter 3, we have discussed the differences between the ACDC framework and our approach from a theoretical perspective. In this section, we conduct experiments to compare the performance of DCNNs with neural networks based on ACDC layers. We first reproduce the experimental setting from Moczulski et al. (2016), and compare both approaches using only linear networks (*i.e.*, networks without any nonlinear activation). The synthetic dataset has been created in order to reproduce the experiment on the regression linear problem proposed by Moczulski et al. (2016). We draw $\mathbf{X}$ and $\mathbf{W}$ from a uniform distribution between [-1, +1] and ϵ from a normal distribution with mean 0 and variance 0.01. The relationship between $\mathbf{X}$ and $\mathbf{Y}$ is defined by $\mathbf{Y} = \mathbf{X}\mathbf{W} + \epsilon$. The results are presented in Figure 4.3. In this simple setting, while both architectures demonstrate good performance, we can observe that DCNNs offer a better convergence rate. In Figure 4.4, we compare neural networks with ReLU activations on CIFAR-10.

We found that networks which are based only on ACDC layers are difficult to train and offer poor accuracy on CIFAR-10 (we have tried different initialization schemes including the one from the original paper, and the one we introduce in this chapter). Moczulski et al. (2016) managed to train a large VGG network however these

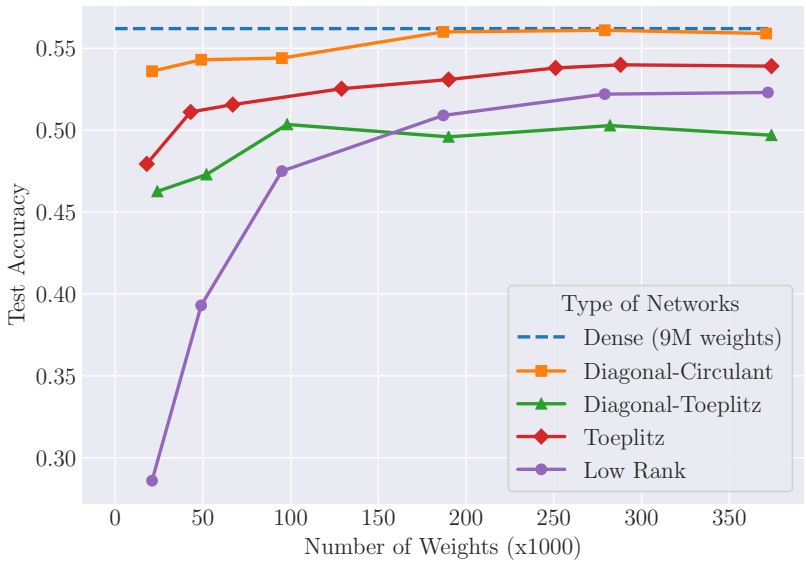


Figure 4.5: Network size *vs.* Accuracy compared on Dense networks, DCNNs, DTNNs, neural networks based on Toeplitz matrices and neural networks based on Low Rank-based matrices.

networks are generally highly redundant and the contribution of the structured layer is difficult to quantify. We also observe that adding a single dense layer improves the convergence rate of ACDC in the linear case, which explains the good results of Moczulski et al. (2016). However, it is difficult to characterize the true contribution of the ACDC layers when the network has a large number of expressive layers. In contrast, deep DCNNs can be trained and offer good performance without additional dense layers (these results are in line with our experiments on the *YouTube-8M* dataset).

Comparison with Dense Networks, Toeplitz Networks and Low Rank Networks

We now compare DCNNs with other state-of-the-art structured networks by measuring the accuracy on a flattened version of the CIFAR-10 dataset. Our baseline is a dense feed-forward network with a fixed number of weights (9 million weights). We compare with DCNNs and with DTNNs (see below), Toeplitz networks, and Low-Rank networks (Yu et al. 2017). We first consider Toeplitz networks which are stacked Toeplitz matrices interleaved with ReLU activations since Toeplitz matrices are closely related to circulant matrices. However, Toeplitz networks have a different

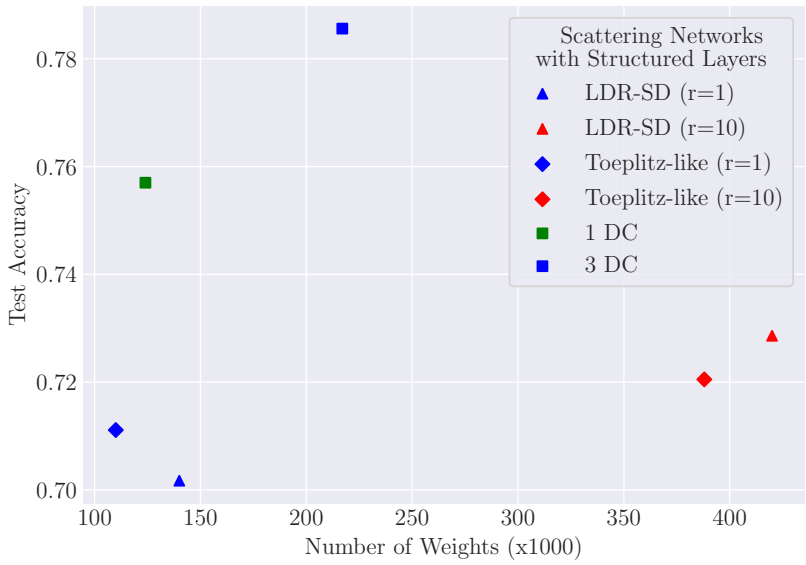


Figure 4.6: Accuracy of different structured architecture given the number of trainable parameters.

structure than DCNNs (they do not include diagonal matrices), therefore, we also experiment using DTNNs, a variant of DCNNs where all the circulant matrices have been replaced by Toeplitz matrices. Finally we conduct experiments using networks based on low-rank matrices as they are also closely related to our work. For each approach, we report the accuracy of several networks with a varying depth ranging from 1 to 40 (DCNNs, Toeplitz networks) and from 1 to 30 (from DTNNs). For low-rank networks, we used a fixed depth network and increased the rank of each matrix from 7 to 40. We also tried to increase the depth of low rank matrices, but we found that deep low-rank networks are difficult to train so we do not report the results here. We compare all the networks based on the number of weights from 21K (0.2% of the dense network) to 370K weights (4% of the dense network) and we report the results in Figure 4.5. First we can see that the size of the networks correlates positively with their accuracy which demonstrated successful training in all cases. We can also see that the DCNNs achieves the maximum accuracy of 56% with 20 layers ($\sim$ 200K weights) which is as good as the dense networks with only 2% of the number of weights. Other approaches also offer good trade-offs but they are not able to reach the accuracy of a dense network.

Architectures	#Parameters	Accuracy
Dense	*9.4M*	*0.562*
DCNN (5 *layers*)	**49K**	**0.543**
DCNN (2 *layers*)	**21K**	**0.536**
LDR–TD ($r = 2$)	64K	0.511
LDR–TD ($r = 3$)	70K	0.473
Toeplitz-like ($r = 2$)	46K	0.483
Toeplitz-like ($r = 3$)	52K	0.496

Table 4.1: Comparison of LDR and Diagonal-Circulant neural networks on a flattened version of CIFAR-10.

Comparison with LDR networks

We now compare DCNNs with the LDR framework using the network configuration proposed by Thomas et al. (2018): a single LDR structured layer followed by a dense layer. In the LDR framework, we can change the size of a network by adjusting the rank of the residual matrix, effectively capturing matrices with a structure that is close to a known structure but not exactly (in the LDR framework, Toeplitz matrices can be encoded with a residual matrix with rank=2, so a matrix that can be encoded with a residual of rank=3 can be seen as Toeplitz-like.). The results are presented in Table 4.1 and demonstrate that DCNNs outperform all LDR networks both in terms of size and accuracy.

4.5.2 Comparison with Other Compression Based Approaches (Q2)

We provide a comparison with other compression based approaches such as HashNet (Chen et al. 2015), Dark Knowledge (Hinton et al. 2015). Table 4.2 shows the test error of DCNN against other known compression techniques on the MNIST datasets. We can observe that DCNN outperforms HashNet (Chen et al. 2015) and Dark Knowledge (Hinton et al. 2015) with fewer number of parameters.

4.5.3 Large-scale Video Classification on the *YouTube-8M* Dataset (Q3)

To understand the performance of deep DCNNs on large-scale applications, we conducted experiments on the *YouTube-8M* video classification with 3.8 training examples introduced by Abu-El-Haija et al. (2016). This section provides a summary of the results obtained on the *YouTube-8M* dataset, the full experimental analysis

Architecture	#Params	Error (%)
LeNet (LeCun et al. 1998)	4.2M	0.61
HashNet (Chen et al. 2015)	46K	2.79
Dark Knowledge (Hinton et al. 2015)	46K	6.32
DCNN	**25K**	**1.74**

Table 4.2: Comparison with compression based approaches.

Architecture	#Weights	GAP@20
Baseline	5.7M	0.773
4 DC	25K	0.599
32 DC	122K	0.685
4 DC + 1 FC	4.46M	**0.747**

Table 4.3: GAP score on the *YouTube-8M* dataset with DCNNs.

is reported in Appendix B. Notice that we favor this experiment over ImageNet applications because modern image classification architectures involve a large number of convolution layers, and compressing convolution layers is out of our scope. Also, as mentioned earlier, testing the performance of DCNN architectures mixed with a large number of expressive layers makes little sense. The *YouTube-8M* includes two datasets describing 8 million labeled videos. Both datasets contain audio and video features for each video. In the first dataset (*aggregated*) all audio and video features have been aggregated every 300 frames. The second dataset (*full*) contains the descriptors for all the frames. To compare the models we use the GAP score (Global Average Precision) proposed by Abu-El-Haija et al. (2016). On the simpler *aggregated* dataset we compared off-the-shelf DCNNs with a dense baseline with 5.7 million weights. On the full dataset, we designed three new compact architectures based on the state-of-the-art architecture introduced by Abu-El-Haija et al. (2016).

Experiments on the aggregated dataset with DCNNs We compared DCNNs with a dense baseline with 5.7 million weights. The goal of this experiment is to discover a good trade-off between depth and model accuracy. To compare the models we use the GAP score (Global Average Precision) following the experimental protocol proposed by Abu-El-Haija et al. (2016), to compare our experiments. Table 4.3 shows the results of our experiments on the *aggregated YouTube-8M* dataset in terms of the number of weights and GAP score. These results suggest that we can compress the baseline at the cost of a little decrease of GAP score.

Architecture	#Weights	GAP@20
original	*45M*	*0.846*
DBoF with DC	36M (*80*)	0.838
FC with DC	41M (*91*)	**0.845**
MoE with DC	12M (***26***)	0.805

Table 4.4: GAP score on the *YouTube-8M* dataset with different layers represented with diagonal-circulant decomposition.

Experiments with DCNNs Deep Bag-of-Frames Architecture: The Deep Bag-of-Frames architecture can be decomposed into three blocks of layers, as illustrated in Figure 4.7. The first block of layers, composed of the Deep Bag-of-Frames embedding (DBoF), is meant to model an embedding of these frames in order to make a simple representation of each video. A second block of fully connected layers (FC) reduces the dimensionality of the output of the embedding and merges the resulting output with a concatenation operation. Finally, the classification block uses a combination of Mixtures-of-Experts (MoE) (Jordan & Jacobs, 1993; Abu-El-Haija et al. 2016) and Context Gating (Miech et al. 2017) to calculate the final class probabilities. Table 4.4 shows the results in terms of the number of weights, size of the model (MB) and GAP on the full dataset, replacing the DBoF block reduces the size of the network without impacting the accuracy. We obtain the best compression ratio by replacing the MoE block with DCNNs (26%) of the size of the original dataset with a GAP score of 0.805 (95% of the score obtained with the original architecture). We conclude that DCNN are both theoretically sound and of practical interest in real, large-scale applications.

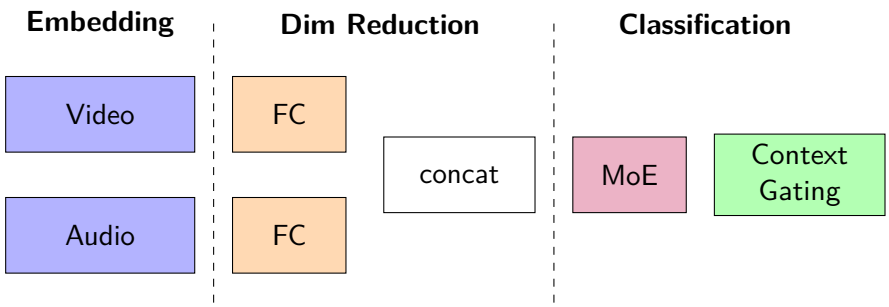


Figure 4.7: Diagram of the state-of-the-art neural network architecture, initially proposed by Abu-El-Haija et al. (2016) and later improved by Miech et al. (2017).

Architectures & Hyper-Parameters: For the first set of our experiments (*experiments on CIFAR-10*), we train all networks for 200 epochs, a batch size of 200, Leaky ReLU activation with a different slope. We minimize the Cross Entropy Loss with Adam optimizer and use a piecewise constant learning rate of 5×10^{-5}, 2.5×10^{-5}, 5×10^{-6} and 1×10^{-6} after respectively 40K, 60K and 80K steps. For the *YouTube-8M* dataset experiments, we built a neural network based on state-of-the-art architecture initially proposed by Abu-El-Haija et al. (2016) and later improved by Miech et al. (2017). Remark that no convolution layer is involved in this application since the input vectors are embeddings of video frames processed using state-of-the-art convolutional neural networks trained on ImageNet. We trained our models with the CrossEntropy loss and used Adam optimizer with a 0.0002 learning rate and a 0.8 exponential decay every 4 million examples. All fully connected layers are composed of 512 units. DBoF, NetVLAD and NetFV are respectively 8192, 64 and 64 of cluster size for video frames and 4096, 32, 32 for audio frames. We used 4 mixtures for the MoE Layer. We used all the available 300 frames for the DBoF embedding. In order to stabilize and accelerate the training, we used batch normalization before each nonlinear activation and gradient clipping.

4.5.4 Exploiting Image Features

Architectures	#Parameters	Accuracy
DC (1 *layers*)	**124K**	**0.757**
DC (3 *layers*)	**217K**	**0.785**
LDR-SD ($r = 1$)	140K	0.701
LDR-SD ($r = 10$)	420K	0.728
Toeplitz-like ($r = 1$)	110K	0.711
Toeplitz-like ($r = 10$)	388K	0.720

Table 4.5: Accuracy of scattering models followed by LDR or DC layer on CIFAR-10 dataset.

Dense layers and DCNNs are not designed to capture task-specific features such as the translation invariance inherently useful in image classification. We can further improve the accuracy of such general-purpose architectures on image classification without dramatically increasing the number of trained parameters by stacking them on top of fixed (ie non-trained) transforms such as the scattering transform (Mallat, 2010). In this section we compare the accuracy of various structured networks, enhanced with the scattering transform, on an image classification task, and run comparative experiments on CIFAR-10.

Our test architecture consists of 2 depth scattering on the RGB images followed by a batch norm and LDR or DC layer. To vary the number of parameters of Scattering+LDR architecture, we increase the rank of the matrix (stacking several LDR matrices quickly exhausted the memory). The Figure 4.6 and Table 4.5 show the accuracy of these architectures given the number of trainable parameters.

First, we can see that the DCNN architecture very much benefits from the scattering transform and is able to reach a competitive accuracy over 78%. We can also see that scattering followed by a DC layer systematically outperforms scattering + LDR or scattering + Toeplitz-like with fewer parameters.

4.6 Concluding Remarks

This chapter dealt with the training and understanding of diagonal-circulant neural networks. To the best of our knowledge, training such networks with a large number of layers had not been done before. We also endowed this kind of architecture with theoretical guarantees, hence enriching and refining previous theoretical work from the literature. More importantly, we showed that DCNNs outperform their competing structured alternatives, including the very recent general approach based on LDR networks. Our results suggest that stacking diagonal-circulant layers with nonlinearities improves the convergence rate and the final accuracy of the network. Formally proving these statements constitutes the future directions of this work. We would like to generalize the good results of DCNNs to convolutional neural networks. We also believe that circulant matrices deserve particular attention in deep learning because of their strong ties with convolutions: a circulant matrix operator is equivalent to the convolution operator with circular padding. This fact makes any contribution to the area of circulant matrices particularly relevant to the field of deep learning with impacts beyond the problem of designing compact models.

Chapter 5

Bound on the Lipschitz Constant of Convolution Layers

Contents

5.1 Introduction

In this chapter we introduce a new upper bound on the largest singular value of convolution layers that is both tight and easy to compute. Instead of using the power method to iteratively approximate this value, we study the properties of *doubly-block Toeplitz matrices* and its links with Fourier analysis. Our work is based

on the result of Gray (2006) which states that an upper bound on the singular value of Toeplitz matrices can be computed from the inverse Fourier transform of the characteristic sequence of these matrices. We first extend this result to doubly-block Toeplitz matrices (*i.e.*, block Toeplitz matrices where each block is Toeplitz) and then to convolutional operators, which can be represented as stacked sequences of doubly-block Toeplitz matrices. From our analysis immediately follows an algorithm for bounding the Lipschitz constant of a convolution layer, and by extension the Lipschitz constant of the whole network. We theoretically study the approximation of this algorithm and show experimentally that it is more efficient and accurate than competing approaches.

Finally, we illustrate our approach on adversarial robustness. Recent work has shown that empirical methods such as adversarial training (AT) offer poor generalization (Schmidt et al. 2018), and can be improved by applying Lipschitz regularization (Farnia et al. 2019). To illustrate the benefit of our new method, we train a large Wide ResNet with Lipschitz regularization and show that it offers a significant improvement over adversarial training alone, and over other methods for Lipschitz regularization. In summary, we make the three following contributions:

1. We devise an upper bound on the singular values of the operator matrix of convolution layers by leveraging Toeplitz matrix theory and its links with Fourier analysis.

2. We propose an efficient algorithm to compute this upper bound which enables its use in the context of Convolutional Neural Networks.

3. We use our method to regularize the Lipschitz constant of neural networks for adversarial robustness and show that it offers a significant improvement over AT alone.

5.2 Results on the Spectrum of Matrices from the Toeplitz Family

5.2.1 Upper-Bounds on the Largest Singular Value of Toeplitz and Block Toeplitz Matrices

Doubly-block Toeplitz matrices inherit the properties of Toeplitz and block Toeplitz matrices. Recall that for Toeplitz and block Toeplitz matrices, there exist no closed-form expression to compute their eigenvalues. However, we can represent

Toeplitz and block Toeplitz matrices with a 2π-periodic function which can describe very precisely the spectrum of the matrices. Let $\{a_h\}_{h\in\mathcal{I}_n}$ be the characteristic sequence of a Toeplitz matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ and let $\{\mathsf{B}^{(h)}\}_{h\in\mathcal{I}_n}$ be the characteristic sequence of $m \times m$ blocks of a block Toeplitz matrix $\mathbf{B} \in \mathbb{R}^{nm\times nm}$ such that $\mathbf{A} = (a_{j-i})_{i,j\in\mathcal{I}_n}$ and $\mathbf{B} = (\mathsf{B}^{(j-i)})_{i,j\in\mathcal{I}_n}$ with $\mathcal{I}_n = \{-n+1, \cdots, n-1\}$. Building on the results presented in the Background (Chapter 2), we can define two trigonometric polynomials $f : \mathbb{R} \to \mathbb{C}$ and $F : \mathbb{R} \to \mathbb{C}^{m\times m}$ as follows:

$$f(\omega) \triangleq \sum_{h\in\mathcal{I}_n} a_h e^{\mathbf{i}h\omega} \qquad F(\omega) \triangleq \sum_{h\in\mathcal{I}_n} \mathsf{B}^{(h)} e^{\mathbf{i}h\omega} \ . \tag{5.1}$$

$f(\omega)$ and $F(\omega)$ are the *inverse Fourier transforms* of the sequences $\{a_h\}_{h\in\mathcal{I}_n}$ and $\{\mathsf{B}^{(h)}\}_{h\in\mathcal{I}_n}$ respectively. From there, inspired by the work done by Grenander et al. (1958), we recall from Section 2.1.2 the operator $\mathbf{T}$ mapping integrable functions to Toeplitz matrices:

$$\mathbf{T}(f) \triangleq \left(\frac{1}{2\pi}\int_0^{2\pi} e^{-\mathbf{i}(i-j)\omega} f(\omega)\ \mathrm{d}\omega\right)_{i,j\in\mathcal{I}_n^+} \ , \tag{5.2}$$

with this operator, we have $\mathbf{T}(f) = \mathbf{A}$ and $\mathbf{T}(F) = \mathbf{B}$.

Now, we can state two known theorems which upper bound the largest singular value of Toeplitz and block Toeplitz matrices with respect to their generating functions.

Theorem 5.1 (Bound on the singular values of Toeplitz matrices)**.** *Let $f : \mathbb{R} \to \mathbb{C}$, be a continuous and 2π-periodic function, then $\mathbf{T}(f) \in \mathbb{R}^{n\times n}$ is a Toeplitz matrix generated by the function f We can bound the largest singular value of the Toeplitz matrix $\mathbf{T}(f)$ as follows:*

$$\sigma_1(\mathbf{T}(f)) \leq \sup_{\omega\in[0,2\pi]} |f(\omega)|. \tag{5.3}$$

Theorem 5.1 is a direct application of Lemma 4.1 in Gray (2006) for real Toeplitz matrices.

Theorem 5.2 (Bound on the singular values of Block Toeplitz matrices Gutiérrez Gutiérrez & Crespo (2012))**.** *Let $F : \mathbb{R} \to \mathbb{C}^{m\times m}$ be a continuous and 2π-periodic matrix-valued function, then, $\mathbf{T}(F) \in \mathbb{R}^{mn\times mn}$ is a block Toeplitz matrix generated by*

the function F. We can bound the largest singular value of the block Toeplitz matrix $\mathbf{T}(F)$ as follows:

$$\sigma_1(\mathbf{T}(F)) \leq \sup_{\omega \in [0,2\pi]} \sigma_1(F(\omega)). \tag{5.4}$$

5.2.2 Upper-Bound on the Largest Singular Value of Doubly-Block Toeplitz Matrices

We extend the reasoning from Toeplitz and block Toeplitz matrices to doubly-block Toeplitz matrices (*i.e.* block Toeplitz matrices where each block is also a Toeplitz matrix). A doubly-block Toeplitz matrix can be generated by a function $f : \mathbb{R}^2 \to \mathbb{C}$ using the 2-dimensional inverse Fourier transform. For this purpose, we define an operator $\mathbf{D}$ which maps a function $f : \mathbb{R}^2 \to \mathbb{C}$ to a doubly-block Toeplitz matrix of size $nm \times nm$. For the sake of clarity, the dependence of $\mathbf{D}(f)$ on m and n is omitted. Let $\mathbf{D}(f) \triangleq (\mathbf{D}_{i,j}(f))_{i,j \in \mathcal{I}_n^+}$ where $\mathbf{D}_{i,j}(f)$ is a $m \times m$ matrix defined as:

$$\mathbf{D}_{i,j}(f) \triangleq \left(\frac{1}{4\pi^2} \int_0^{2\pi} \int_0^{2\pi} e^{-\mathbf{i}((i-j)\omega_1 + (k-l)\omega_2)} f(\omega_1, \omega_2) \, \mathrm{d}\omega_1 \, \mathrm{d}\omega_2 \right)_{k,l \in \mathcal{I}_m^+}. \tag{5.5}$$

We are now able to combine Theorems 5.1 and 5.2 to bound the largest singular value of doubly-block Toeplitz matrices with respect to their generating functions. Note that in the following, we only consider generating functions as trigonometric polynomials with real coefficients therefore the matrices generated by $\mathbf{D}(f)$ are real.

Theorem 5.3 (Bound on the largest singular value of a Doubly-Block Toeplitz Matrix)**.** *Let $f : \mathbb{R}^2 \to \mathbb{C}$ be a multivariate trigonometric polynomial of the form:*

$$f(\omega_1, \omega_2) \triangleq \sum_{h_1 \in \mathcal{I}_n} \sum_{h_2 \in \mathcal{I}_m} d_{h_1,h_2} e^{\mathbf{i}(h_1\omega_1 + h_2\omega_2)}. \tag{5.6}$$

Then, $\mathbf{D}(f) \in \mathbb{R}^{nm \times nm}$ is a doubly-block Toeplitz matrix where d_{h_1,h_2} is the ${h_2}^{th}$ scalar of the ${h_1}^{th}$ block of the matrix. We can bound the largest singular value of the matrix $\mathbf{D}(f)$ as follows:

$$\sigma_1(\mathbf{D}(f)) \leq \sup_{\omega_1,\omega_2 \in [0,2\pi]^2} |f(\omega_1, \omega_2)| \tag{5.7}$$

Proof of Theorem 5.3. By definition, a doubly-block Toeplitz matrix is a block matrix where each block is a Toeplitz matrix. Let $\mathbf{A}$ be a $mn \times mn$ doubly-block Toeplitz matrices determined by the sequence of blocks $\{\mathsf{A}^{(-n+1)}, \ldots, \mathsf{A}^{(n-1)}\}$ of size $m \times m$ where the blocks A are Toeplitz matrices such that $\mathsf{A}^{(i)}$ is determined by the sequence of scalars $\{d_{i,-m+1}, \ldots, d_{i,m-1}\}$. Therefore the matrix $\mathbf{A}$ can be expressed with the operator $\mathbf{T}$ with the matrix-valued generating function $F : \mathbb{R} \to \mathbb{C}^{n\times n}$ such that:

$$F(\omega_1) = \sum_{h_1 \in \mathcal{I}_n} \mathsf{A}^{(h_1)} e^{\mathbf{i} h_1 \omega_1} \tag{5.8}$$

From Theorem 5.2 we have:

$$\sigma_1(\mathbf{T}(F)) \leq \sup_{\omega_1 \in [0,2\pi]} \sigma_1(F(\omega_1)) \tag{5.9}$$

Note that because the function F is a linear combination of the Toeplitz matrices A and that Toeplitz matrices are closed under addition and scalar product, $F(\omega_1)$ is also a Toeplitz matrix of size $m \times m$. Therefore, we can define a function $f : \mathbb{R}^2 \to \mathbb{C}$ such that:

$$f(\omega_1, \omega_2) = \sum_{h_2 \in \mathcal{I}_m} [F(\omega_1)]_{h_2} e^{\mathbf{i} h_2 \omega_2} \tag{5.10}$$

$$f(\omega_1, \omega_2) = \sum_{h_2 \in \mathcal{I}_m} \left[\sum_{h_1 \in \mathcal{I}_n} \mathsf{A}^{(h_1)} e^{\mathbf{i} h_1 \omega_1} \right]_{h_2} e^{\mathbf{i} h_2 \omega_2} \tag{5.11}$$

$$f(\omega_1, \omega_2) = \sum_{h_2 \in \mathcal{I}_m} \left[\sum_{h_1 \in \mathcal{I}_n} \mathsf{A}^{(h_1)} \right]_{h_2} e^{\mathbf{i}(h_1 \omega_1 + h_2 \omega_2)} \tag{5.12}$$

$$f(\omega_1, \omega_2) = \sum_{h_1 \in \mathcal{I}_n} \sum_{h_2 \in \mathcal{I}_m} d_{h_1,h_2} e^{\mathbf{i}(h_1 \omega_1 + h_2 \omega_2)} \quad , \tag{5.13}$$

From Theorem 5.1, we can write:

$$\sigma_1(F(\omega_1)) \leq \sup_{\omega_2 \in [0,2\pi]} |f(\omega_1, \omega_2)| \tag{5.14}$$

$$\Rightarrow \sup_{\omega_1 \in [0,2\pi]} \sigma_1(F(\omega_1)) \leq \sup_{\omega_1, \omega_2 \in [0,2\pi]^2} |f(\omega_1, \omega_2)| \tag{5.15}$$

$$\Rightarrow \sigma_1(\mathbf{T}(F)) \leq \sup_{\omega_1, \omega_2 \in [0,2\pi]^2} |f(\omega_1, \omega_2)| \tag{5.16}$$

> Because the function $f(\omega_1, \cdot)$ is the generating function of $F(\omega_1)$ it is easy to show that the function f is also the generating function of the matrix $\mathbf{T}(F)$. Therefore, $\mathbf{T}(F) = \mathbf{D}(f)$ which concludes the proof. ■

5.3 Extending the Bound to Convolutional Layers

From now on, without loss of generality, we will assume that $n = m$ to simplify notations. A discrete convolution operation with a 2-dimensional kernel applied on a 2-dimensional signal (*e.g.*, an image) is equivalent to a matrix multiplication with a doubly-block Toeplitz matrix (Jain, 1989). In practice, the input signal often has 3 or more dimensions called *channels* (for example, RGB images have 3 channels, one for each color). If we denote c_{in}, the number of channels of the input signal, then, the input signal is a tensor of size $c_{in} \times n \times n$. Moreover, when we perform multiple convolutions on the same signal the output signal will have multiple channels denoted c_{out}. Therefore, the kernel is defined as a 4-dimensional tensor of size: $c_{out} \times c_{in} \times s \times s$. The operation performed by a 4-dimensional kernel on a 3-dimensional signal can be formulated as the concatenation (horizontally and vertically) of doubly-block Toeplitz matrices. Hereafter, we bound the singular value of multiple vertically stacked doubly-block Toeplitz matrices which corresponds to the operation performed by a 3-dimensional kernel with $c_{out} = 1$ on a 3-dimensional signal.

Theorem 5.4 (Bound on the largest singular value of stacked Doubly-block Toeplitz matrices)**.** *Consider doubly-block Toeplitz matrices* $\mathbf{D}(f_1), \ldots, \mathbf{D}(f_{c_{in}})$ *where each* $f_i : \mathbb{R}^2 \to \mathbb{C}$ *is a multivariate polynomial of the same form as Equation* (5.6)*. Construct a matrix* $\mathbf{M}$ *with* $c_{in} \times n^2$ *rows and* n^2 *columns, as follows:*

$$\mathbf{M} \triangleq \left(\mathbf{D}^\top(f_1), \ldots, \mathbf{D}^\top(f_{c_{in}})\right)^\top. \tag{5.17}$$

Then, we can bound the largest singular value of the matrix $\mathbf{M}$ *as follows:*

$$\sigma_1(\mathbf{M}) \leq \sup_{\omega_1, \omega_2 \in [0, 2\pi]^2} \sqrt{\sum_{i=1}^{c_{in}} |f_i(\omega_1, \omega_2)|^2} \ . \tag{5.18}$$

In order to prove Theorem 5.4, we will need the following lemmas:

Lemma 5.1 (Gutiérrez Gutiérrez & Crespo (2012)). *Let $f : \mathbb{R}^2 \to \mathbb{C}$ and $g : \mathbb{R}^2 \to \mathbb{C}$ be two continuous and 2π-periodic functions. Let $\mathbf{D}(f)$ and $\mathbf{D}(g)$ be doubly-block Toeplitz matrices generated by the functions f and g respectively. Then:*

- $\mathbf{D}^\top(f) = \mathbf{D}(f^*)$
- $\mathbf{D}(f) + \mathbf{D}(g) = \mathbf{D}(f + g)$

Lemma 5.2 (Serra (1994)). *If the doubly-block Toeplitz matrix $\mathbf{D}(f)$ is generated by a function $f : \mathbb{R}^2 \to \mathbb{R}$, then the matrix $\mathbf{D}(f)$ is Hermitian.*

Lemma 5.3 (Serra (1994)). *If the doubly-block Toeplitz matrix $\mathbf{D}(f)$ is generated by a non-negative function f not identically zero, then the matrix $\mathbf{D}(f)$ is positive definite.*

Lemma 5.4 (Zhang (2011)). *Let $\mathbf{A}$ and $\mathbf{B}$ be Hermitian positive semi-definite matrices. If $\mathbf{A} - \mathbf{B}$ is positive semi-definite, then:*

$$\lambda_1(\mathbf{B}) \leq \lambda_1(\mathbf{A}) \tag{5.19}$$

We need now to extend the well known Widom identity (Widom, 1976) which expresses the relation between Toeplitz and Hankel matrices to doubly-block Toeplitz and Hankel matrices. Let us first generalize the doubly-block Toeplitz operator presented in Section 5.2.2.

Given a function $f : \mathbb{R}^2 \to \mathbb{C}$, let $\mathbf{G}^{\alpha_p}(f)$ be a matrix such that $\mathbf{G}^{\alpha_p}(f) = \left(\mathbf{G}^{\alpha_p}_{i,j}(f)\right)_{i,j \in \mathcal{I}_n^+}$ where $\mathbf{G}^{\alpha_p}_{i,j}$ is defined as:

$$\mathbf{G}^{\alpha_p}_{i,j}(f) = \left(\frac{1}{4\pi^2}\int_0^{2\pi}\int_0^{2\pi} e^{-\mathbf{i}\alpha_p(i,j,k,l,\omega_1,\omega_2)} f(\omega_1,\omega_2)\ \mathrm{d}\omega_1\ \mathrm{d}\omega_2\right)_{k,l \in \mathcal{I}_n^+}. \tag{5.20}$$

Note that as with the operator $\mathbf{D}(f)$ we only consider generating functions as trigonometric polynomials with real coefficients therefore the matrices generated by $\mathbf{G}(f)$ are real. And as with the operator $\mathbf{D}(f)$, the matrices generated by the operator $\mathbf{G}^{\alpha_p}$ are of size $n^2 \times n^2$.
We will use the following α functions:

$$\alpha_0(i,j,k,l,\omega_1,\omega_2) = (-j-i-1)\omega_1 + (k-l)\omega_2$$

$$\alpha_1(i,j,k,l,\omega_1,\omega_2) = (i-j)\omega_1 + (-l-k-1)\omega_2$$

$$\alpha_2(i,j,k,l,\omega_1,\omega_2) = (-j-i-1)\omega_1 + (-l-k-1)\omega_2$$

$$\alpha_3(i,j,k,l,\omega_1,\omega_2) = (-j-i+n)\omega_1 + (-l-k-1)\omega_2$$

We now present the generalization of the Widom identity for Doubly-Block Toeplitz matrices below:

Lemma 5.5 (Extension of Widom Identity to doubly-block operators)**.** *Let $f : \mathbb{R}^2 \to \mathbb{C}$ and $g : \mathbb{R}^2 \to \mathbb{C}$ be two continuous and 2π-periodic functions. Let fg be the product of the functions f and g such that $(fg)(\omega_1,\omega_2) = f(\omega_1,\omega_2)g(\omega_1,\omega_2)$. We can decompose the Doubly-Block Toeplitz matrix $\mathbf{D}(fg)$ as follows:*

$$\mathbf{D}(fg) = \mathbf{D}(f)\mathbf{D}(g) + \sum_{p=0}^{3} \mathbf{G}^{\alpha_p\top}(f^*)\mathbf{G}^{\alpha_p}(g) + \mathbf{J}_{n^2}\left(\sum_{p=0}^{3} \mathbf{G}^{\alpha_p\top}(f)\mathbf{G}^{\alpha_p}(g^*)\right)\mathbf{J}_{n^2}. \tag{5.21}$$

where $\mathbf{J}_{n^2}$ is the reflection of the identity matrix of size $n^2 \times n^2$.

The proof of this Lemma is delayed to Appendix A. Now we have all the elements to prove Theorem 5.4 which bounds the largest singular value of vertically stacked doubly-block Toeplitz matrices with their generating functions.

Proof of Theorem 5.4. Consider doubly-block Toeplitz matrices $\mathbf{D}(f_1),\ldots,\mathbf{D}(f_{c_{in}})$ where each $f_i : \mathbb{R}^2 \to \mathbb{C}$ is a multivariate polynomial of the same form as Equation (5.6). Construct a matrix $\mathbf{M}$ with $c_{in} \times n^2$ rows and n^2 columns, such that:

$$\mathbf{M} \triangleq \left(\mathbf{D}^\top(f_1),\ldots,\mathbf{D}^\top(f_{c_{in}})\right)^\top. \tag{5.22}$$

First, let us observe the following equality which relates the largest singular value of the matrix $\mathbf{M}$ and the largest eigenvalue of the sum of the doubly-block Toeplitz matrices composing $\mathbf{M}$:

$$\sigma_1^2(\mathbf{M}) = \lambda_1\left(\mathbf{M}^\top\mathbf{M}\right) = \lambda_1\left(\sum_{i=1}^{c_{in}} \mathbf{D}^\top(f_i)\mathbf{D}(f_i)\right). \tag{5.23}$$

Secondly, let us bound the largest eigenvalue of the sum of doubly-block Toeplitz generated by $|f_i|^2$:

$$\lambda_1\left(\sum_{i=1}^{c_{in}} \mathbf{D}\left(|f_i|^2\right)\right) = \lambda_1\left(\mathbf{D}\left(\sum_{i=1}^{c_{in}} |f_i|^2\right)\right) \tag{5.24}$$

$$= \sigma_1\left(\mathbf{D}\left(\sum_{i=1}^{c_{in}} |f_i|^2\right)\right) \tag{5.25}$$

$$\leq \sup_{\omega_1,\omega_2\in[0,2\pi]^2} \sum_{i=1}^{c_{in}} |f_i(\omega_1,\omega_2)|^2. \tag{5.26}$$

where the first equality is due to Lemma 5.1, the second equality is due to Lemma 5.2 and the last inequality is due to Theorem 5.3. To finalize the proof, we need to demonstrate that the following inequality holds:

$$\lambda_1\left(\sum_{i=1}^{c_{in}} \mathbf{D}^\top(f_i)\mathbf{D}(f_i)\right) \leq \lambda_1\left(\mathbf{D}\left(\sum_{i=1}^{c_{in}} |f_i|^2\right)\right). \tag{5.27}$$

In order to prove the inequality above, let us study the positive definiteness of the following matrix:

$$\mathbf{D}\left(\sum_{i=1}^{c_{in}} |f_i|^2\right) - \sum_{i=1}^{c_{in}} \mathbf{D}^\top(f_i)\mathbf{D}(f_i), \tag{5.28}$$

One can observe that the term $\mathbf{D}(\sum_{i=1}^{c_{in}} |f_i|^2)$ of Equation (5.28) is a real symmetric positive definite matrix by Lemmas 5.2 and 5.3. Furthermore, the term $\sum_{i=1}^{c_{in}} \mathbf{D}^\top(f_i)\mathbf{D}(f_i)$ of Equation (5.28) is a sum of positive semi-definite matrices. Therefore, if the subtraction of the two is positive semi-definite, one could apply Lemma 5.4 to prove the Equation (5.27). We know from Lemma 5.5 that

$$\mathbf{D}(fg) - \mathbf{D}(f)\mathbf{D}(g) = \sum_{p=0}^{3} \mathbf{G}^{\alpha_p\top}(f^*)\mathbf{G}^{\alpha_p}(g) + \mathbf{J}\left(\sum_{p=0}^{3} \mathbf{G}^{\alpha_p\top}(f)\mathbf{G}^{\alpha_p}(g^*)\right)\mathbf{J}. \tag{5.29}$$

By choosing $f = f^*$, $g = f$ and with the use of Lemma 5.1, we obtain:

$$\mathbf{D}(f^* f) - \mathbf{D}(f^*)\mathbf{D}(f) = \mathbf{D}(|f|^2) - \mathbf{D}^\top(f)\mathbf{D}(f) \tag{5.30}$$

$$= \sum_{p=0}^{3} \mathbf{G}^{\alpha_p\top}(f)\mathbf{G}^{\alpha_p}(f) + \mathbf{J}\left(\sum_{p=0}^{3} \mathbf{G}^{\alpha_p\top}(f^*)\mathbf{G}^{\alpha_p}(f^*)\right)\mathbf{J}. \tag{5.31}$$

From Equation (5.30), we can see that the matrix $\mathbf{D}(|f|^2) - \mathbf{D}^\top(f)\mathbf{D}(f)$ is positive semi-definite because it can be decomposed into a sum of positive semi-definite matrices and because positive semi-definiteness is closed under addition, we have:

$$\sum_{i=1}^{c_{in}} \Big(\mathbf{D}\Big(|f_i|^2\Big) - \mathbf{D}^\top(f_i)\mathbf{D}(f_i)\Big) \geq 0 \tag{5.32}$$

By re-arranging and with the use Lemma 5.1, we obtain:

$$\sum_{i=1}^{c_{in}} \mathbf{D}\Big(|f_i|^2\Big) - \sum_{i=1}^{c_{in}} \Big(\mathbf{D}^\top(f_i)\mathbf{D}(f_i)\Big) \geq 0 \tag{5.33}$$

$$\mathbf{D}\left(\sum_{i=1}^{c_{in}} |f_i|^2\right) - \sum_{i=1}^{c_{in}} \Big(\mathbf{D}^\top(f_i)\mathbf{D}(f_i)\Big) \geq 0 \tag{5.34}$$

We can conclude that the Equation (5.27) is true and therefore by Lemma 5.4 we have:

$$\lambda_1\left(\sum_{i=1}^{c_{in}} \mathbf{D}^\top(f_i)\mathbf{D}(f_i)\right) \leq \lambda_1\left(\mathbf{D}\left(\sum_{i=1}^{c_{in}} |f_i|^2\right)\right) \tag{5.35}$$

$$\sigma_1^2(\mathbf{M}) \leq \sup_{\omega_1,\omega_2\in[0,2\pi]^2} \sum_{i=1}^{c_{in}} |f_i(\omega_1,\omega_2)|^2 \tag{5.36}$$

$$\sigma_1(\mathbf{M}) \leq \sup_{\omega_1,\omega_2\in[0,2\pi]^2} \sqrt{\sum_{i=1}^{c_{in}} |f_i(\omega_1,\omega_2)|^2} \tag{5.37}$$

which concludes the proof. ■

To have a bound on the full convolution operation, we extend Theorem 5.4 to take into account the number of output channels. The matrix of a full convolution operation is a block matrix where each block is a doubly-block Toeplitz matrix. Below, we present our main result:

Theorem 5.5 (Bound on the largest singular value on the discrete convolution operation)**.** *Let us define doubly-block Toeplitz matrices* $\mathbf{D}(f_{1,1}), \ldots, \mathbf{D}(f_{c_{in},c_{out}})$ *where each* $f_{i,j} : \mathbb{R}^2 \to \mathbb{C}$ *is a multivariate polynomial of the same form as Equation* (5.6)*. Construct a matrix* $\mathbf{M}$ *with* $c_{in} \times n^2$ *rows and* $c_{out} \times n^2$ *columns. We can bound the largest singular value of the matrix* $\mathbf{M}$ *as follows:*

$$\sigma_1(\mathbf{M}) \leq \sqrt{\sum_{i=1}^{c_{out}} \sup_{\omega_1,\omega_2 \in [0,2\pi]^2} \sum_{j=1}^{c_{in}} |f_{ij}(\omega_1, \omega_2)|^2}. \tag{5.38}$$

First, in order to prove Theorem 5.5, we will need the following lemma which bounds the singular values of a matrix constructed from the concatenation of multiple matrices.

Lemma 5.6 (Bound on the singular values of concatenation of matrices)**.** *Let us define matrices* $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(p)}$ *with* $\mathbf{A}^{(i)} \in \mathbb{R}^{n \times n}$*. Let us construct the matrix* $\mathbf{M} \in \mathbb{R}^{n \times pn}$ *as follows:*

$$\mathbf{M} \triangleq \left(\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(p)}\right) \tag{5.39}$$

where $(\,\cdot\,)$ *define the concatenation operation. Then, we can bound the singular values of the matrix* $\mathbf{M}$ *as follows:*

$$\sigma_1(\mathbf{M}) \leq \sqrt{\sum_{i=1}^{p} \sigma_1(\mathbf{A}^{(i)})^2} \tag{5.40}$$

Proof of Lemma 5.6.

$$\sigma_1(\mathbf{M})^2 = \lambda_1\left(\mathbf{M}\mathbf{M}^\top\right) \tag{5.41}$$

$$= \lambda_1\left(\sum_{i=1}^{p} \mathbf{A}^{(i)}\mathbf{A}^{(i)\top}\right) \tag{5.42}$$

$$\leq \sum_{i=1}^{p} \lambda_1 \left(\mathbf{A}^{(i)} \mathbf{A}^{(i)\top} \right) \tag{5.43}$$

$$\leq \sum_{i=1}^{p} \sigma_1 \left(\mathbf{A}^{(i)} \right)^2 \tag{5.44}$$

$$\Leftrightarrow \sigma_1(\mathbf{M}) \leq \sqrt{\sum_{i=1}^{p} \sigma_1(\mathbf{A}^{(i)})^2} \tag{5.45}$$

which concludes the proof. ■

Now, the proof of Theorem 5.5 is a combination of Lemma 5.6 and Theorem 5.4.

Proof of Theorem 5.5. Let us define the matrix $\mathbf{M}^{(i)}$ as follows:

$$\mathbf{M}^{(i)} = \left(\mathbf{D}(f_{1,i})^\top, \ldots, \mathbf{D}(f_{c_{in},i})^\top \right)^\top. \tag{5.46}$$

We can express the matrix $\mathbf{M}$ as the concatenation of multiple $\mathbf{M}^{(i)}$ matrices:

$$\mathbf{M} = \left(\mathbf{M}^{(1)}, \ldots, \mathbf{M}^{(c_{out})} \right) \tag{5.47}$$

Then, we can bound the singular values of the matrix $\mathbf{M}$ as follows:

$$\sigma_1(\mathbf{M}) \leq \sqrt{\sum_{i=1}^{c_{out}} \sigma_1(\mathbf{M}^{(i)})^2} \tag{5.48}$$

$$\sigma_1(\mathbf{M}) \leq \sqrt{\sum_{j=1}^{c_{out}} \sup_{\omega_1,\omega_2 \in [0,2\pi]^2} \sum_{i=1}^{c_{in}} |f_{i,j}(\omega_1, \omega_2)|^2} \tag{5.49}$$

where the first inequality is due to Lemma 5.6 and the second one is due to Theorem 5.4. This concludes the proof. ■

Theorem 5.5 depends on the convolution matrix $\mathbf{M}$, however, we can easily formulate the bound with the values of a 4-dimensional kernel. Let us define a kernel

$\mathbf{K} \in \mathbb{R}^{c_{out} \times c_{in} \times s \times s}$, a padding $p \in \mathbb{N}$ and $d = \lfloor s/2 \rfloor$ the degree of the trigonometric polynomial, then:

$$f_{ij}(\omega_1, \omega_2) = \sum_{h_1=-d}^{d} \sum_{h_2=-d}^{d} k_{i,j,h_1,h_2} e^{\mathbf{i}(h_1\omega_1 + h_2\omega_2)}. \tag{5.50}$$

where $k_{i,j,h_1,h_2} = (\mathbf{K})_{i,j,a,b}$ with $a = s - p - 1 + h_1$ and $b = s - p - 1 + h_2$.

In the rest of the chapter, we will refer to the bound in Theorem 5.5 applied to a kernel as LipBound and we denote LipBound($\mathbf{K}$) the Lipschitz upper bound of the convolution performed by the kernel $\mathbf{K}$.

5.4 Computation and Performance Analysis of LipBound

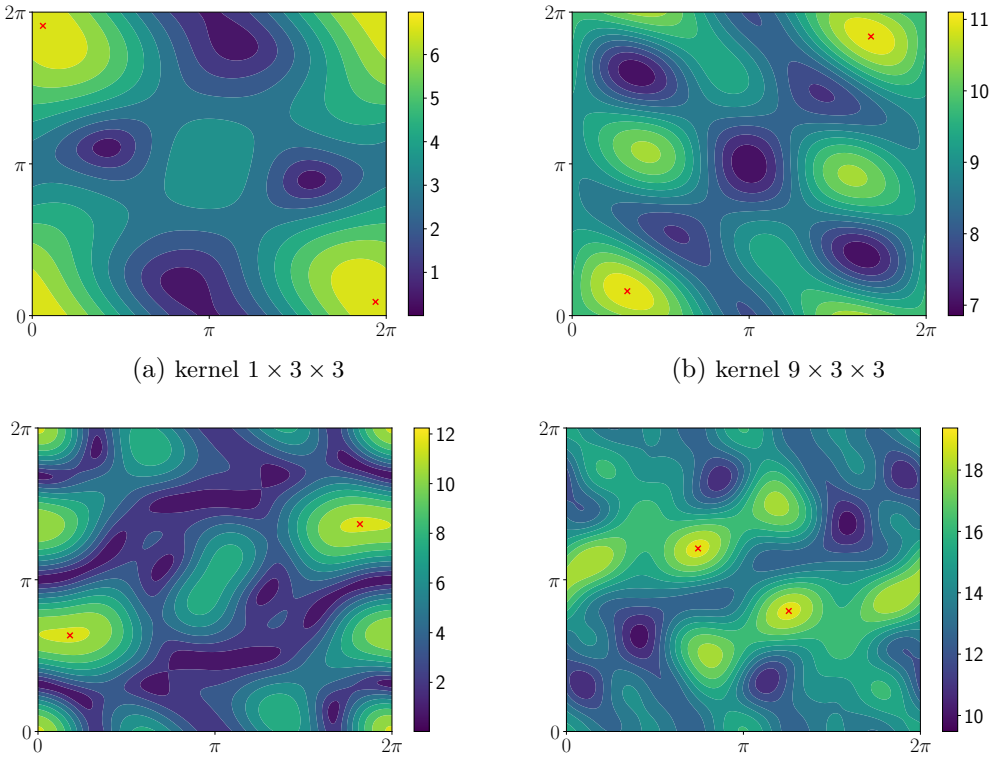
This section aims at analyzing the bound introduced in Theorem 5.5. First, we present an algorithm to efficiently compute the bound, we analyze its tightness by comparing it against the true largest singular value. Finally, we compare the efficiency and the accuracy of our bound against the state-of-the-art methods.

5.4.1 The Maximum Modulus of a Trigonometric Polynomial

In order to compute LipBound from Theorem 5.5, we have to compute the maximum modulus of several trigonometric polynomials. However, finding the maximum modulus of a trigonometric polynomial has been known to be NP-Hard (Pfister & Bresler, 2018), and in practice they exhibit low convexity (see Figure 5.1). We found that for 2-dimensional kernels, a simple grid search algorithm such as PolyGrid (see Algorithm 5.1), works better than more sophisticated approximation algorithms (*e.g.* Green (1999) and De La Chevrotiere (2009)). This is because the complexity of the computation depends on the degree of the polynomial which is equal to $\lfloor s/2 \rfloor$ where s is the size of the kernel and is usually small in most practical settings (*e.g.* $s = 3$). Furthermore, the grid search algorithm can be parallelized effectively on CPUs or GPUs and runs within less time than alternatives with lower asymptotic complexity.

To fix the number of samples S in the grid search, we rely on the work of (Pfister & Bresler, 2018), who has analyzed the quality of the approximation depending on S. Following this work we first define Θ_S, the set of S equidistant sampling points as follows:

$$\Theta_S \triangleq \left\{ \omega \mid \omega = k \cdot \frac{2\pi}{S} \text{ with } k = 0, \ldots, S-1 \right\}. \tag{5.51}$$



(a) kernel $1 \times 3 \times 3$

(b) kernel $9 \times 3 \times 3$

(c) kernel $1 \times 5 \times 5$

(d) kernel $9 \times 5 \times 5$

Figure 5.1: Contour plots of multivariate trigonometric polynomials where the values of the coefficient are the values of a random convolutional kernel. The red dots in the figures represent the maximum modulus of the trigonometric polynomials.

Then, for a trigonometric polynomial $f : [0, 2\pi]^2 \to \mathbb{C}$, we have:

$$\max_{\omega_1,\omega_2 \in [0,2\pi]^2} |f(\omega_1, \omega_2)| \leq (1-\alpha)^{-1} \max_{\omega_1',\omega_2' \in \Theta_S^2} |f(\omega_1', \omega_2')|, \tag{5.52}$$

where d is the degree of the polynomial and $\alpha = 2d/S$. For a 3×3 kernel which gives a trigonometric polynomial of degree 1, we use $S = 10$ which gives $\alpha = 0.2$. Using this result, we can now compute LipBound for a convolution operator with c_{out} output channels as per Theorem 5.4.

Algorithm 5.1 PolyGrid Algorithm

```
procedure PolyGrid(f, S)                    ▷ polynomial f, number of samples S
    σ ← 0, ω1 ← 0, ϵ ← 2π/S
    for i = 0 to S − 1 do
        ω2 ← 0
        for j = 0 to S − 1 do
            ω2 ← ω2 + ϵ
            σ ← max(σ, |f(ω1, ω2)|)
        end for
        ω1 ← ω1 + ϵ
    end for
    return σ                                ▷ approximated maximum modulus of f
end procedure
```

5.4.2 Analysis of the Tightness of the Bound

In this section, we study the tightness of the bound with respect to the dimensions of the doubly-block Toeplitz matrices. For each $n \in \mathbb{N}$, we define the matrix $\mathbf{M}^{(n)}$ of size $kn^2 \times n^2$ as follows:

$$\mathbf{M}^{(n)} \triangleq \left(\mathbf{D}^{(n)\top}(f_1), \ldots, \mathbf{D}^{(n)\top}(f_k)\right)^\top \tag{5.53}$$

where the matrices $\mathbf{D}^{(n)}(f_i)$ are of size $n^2 \times n^2$. To analyze the tightness of the bound, we define the function Γ, which computes the difference between LipBound and the largest singular value of the function $\mathbf{M}^{(n)}$:

$$\Gamma(n) = \text{LipBound}(\mathbf{K}_{\mathbf{M}^{(n)}}) - \sigma_1(\mathbf{M}^{(n)}) \tag{5.54}$$

where $\mathbf{K}_{\mathbf{M}^{(n)}}$ is the convolution kernel associated with the matrix $\mathbf{M}^{(n)}$.

To compute a very close approximation of the exact largest singular value of $\mathbf{M}^{(n)}$ for a specific n, we use the Implicitly Restarted Arnoldi Method (IRAM) (Lehoucq & Sorensen, 1996) available in SciPy. The results of this experiment are presented in Figure 5.2. We observe that the difference between the bound and the actual value (approximation gap) quickly decreases as the input size increases. For an input size of 50, the approximation gap is as low as 0.012 using a standard $6 \times 3 \times 3$ convolution kernel. For a larger input size such as ImageNet (224), the gap is lower than 4.10^{-4}. Therefore LipBound gives an almost exact value of the largest singular value of the operator matrix for most realistic settings.

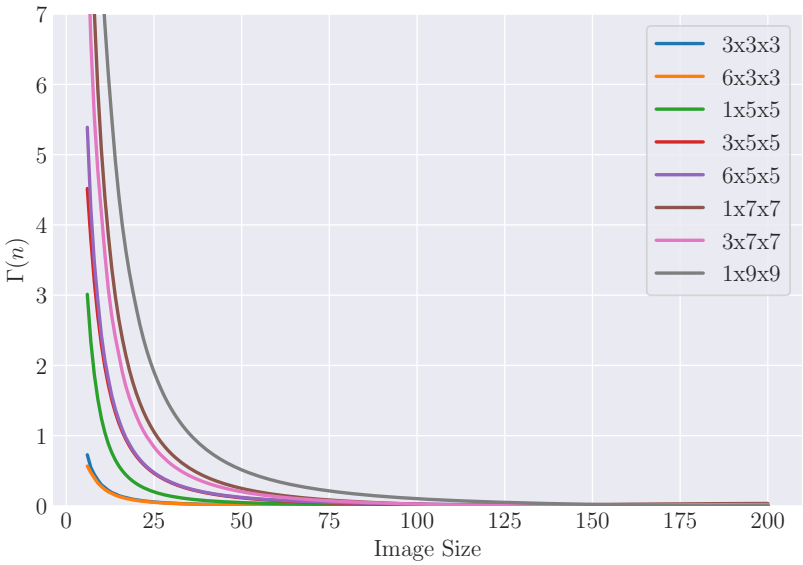


Figure 5.2: Representation of the function $\Gamma(n)$ defined for different kernel size.

5.4.3 Comparison of LipBound with State-of-the-Art Approaches

	1x3x3		32x3x3	
	Ratio	**Time (ms)**	**Ratio**	**Time (ms)**
Sedghi et al.	0.431 ± 0.042	1088 ± 251	0.666 ± 0.123	1729 ± 399
Singla & Feizi	1.293 ± 0.126	1.90 ± 0.48	1.441 ± 0.188	1.90 ± 0.46
Farnia et al.	0.973 ± 0.006	4.30 ± 0.64	0.972 ± 0.004	4.93 ± 0.67
LipBound	0.992 ± 0.012	0.49 ± 0.05	0.984 ± 0.021	0.63 ± 0.46

Table 5.1: Comparison of the accuracy of approximation methods for computing an approximation of the largest singular value of a convolution layer.

In this section we compare our PolyGrid algorithm with the values obtained using alternative approaches. We consider the 3 alternative techniques by Sedghi et al. (2018), Farnia et al. (2019) and Singla & Feizi (2019) which have been described in Chapter 3, Section 3.2.

To compare the different approaches, we extracted 20 kernels from a trained model. For each kernel we construct the corresponding doubly-block Toeplitz matrix and compute its largest singular value. Then, we compute the ratio between the approximation obtained with the considered approach and the approximated singular value obtained by IRAM, and average the ratios over the 20 kernels. Thus good

Network	LipBound (ms)	Power Method (ms)	Ratio
AlexNet	4.75 ± 1.10	38.75 ± 2.52	8.14
ResNet 18	29.88 ± 1.73	148.35 ± 14.92	4.96
ResNet 34	54.73 ± 3.62	266.85 ± 25.35	4.87
ResNet 50	60.77 ± 4.62	467.61 ± 36.52	7.69
ResNet 101	102.72 ± 11.53	817.06 ± 102.87	7.95
ResNet 152	158.80 ± 20.84	1373.57 ± 164.37	8.64
DenseNet 121	125.55 ± 14.59	937.35 ± 11.52	7.46
DenseNet 161	176.11 ± 19.13	1292.61 ± 30.50	7.33
DenseNet 169	188.03 ± 19.74	1372.62 ± 21.16	7.29
DenseNet 201	281.13 ± 23.41	1930.19 ± 170.79	6.86
VGG 11	13.73 ± 1.19	81.78 ± 4.45	5.95
VGG 13	14.96 ± 1.99	102.04 ± 4.20	6.82
VGG 16	21.92 ± 1.94	132.29 ± 5.99	6.03
VGG 19	29.05 ± 0.66	162.28 ± 4.87	5.58
WideResNet 50-2	113.28 ± 45.44	468.74 ± 6.54	4.13
SqueezeNet 1-0	18.44 ± 5.93	222.40 ± 25.49	12.05
SqueezeNet 1-1	18.26 ± 6.65	209.80 ± 3.59	11.48

Table 5.2: Efficiency of LipBound computation *vs.* the Power Method with 10 iterations on full networks.

approximations result in approximation ratios that are close to 1. The results of this experiment are presented in Table 5.1. The comparison has been made on a Tesla V100 GPU. The time was computed with the PyTorch CUDA profiler and we "warmed" up the GPU before starting the timer for caching purposes.

The method introduced by Sedghi et al. (2018) and presented in Section 3.2.3 computes the largest singular value of convolution layers based on doubly-block circulant matrices. Doubly-block circulant matrices perform a convolution with a "wrapping around" operation which do not correspond to the more general setting. We can see in Table 5.1 that the values differ by an important margin. This technique is also computationally expensive as it requires computing the SVD of n^2 small matrices where n is the size of inputs. Singla & Feizi (2019) have shown that the singular value of the reshape kernel is a bound on the largest singular value of the convolution layer. Their approach is very efficient but the approximation is loose and overestimate the real value. As said previously, the power method provides a good approximation at the expense of the efficiency. We also compare our approach

to the power method with 10 iterations from Farnia et al. (2019) (see Algorithm 3.2). The results show that our proposed technique: PolyGrid algorithm can get the best of both worlds. It achieves a near perfect accuracy while being very efficient to compute.

The results of Table 5.1 shows the performance for the computation for only one convolution layer. However, during the training Lipbound or the power method need to be computed for every layer of the network and the computation time is dependent on the architecture of the network, for example, the size of the activations or the size of the kernels. In Table 5.2, we compare our approach method against the power method on the full architecture, *i.e.*, the time needed for the computation on all the layers of the networks. We measure on the following convolutional architectures: AlexNet (Krizhevsky et al. 2012), ResNet (He et al. 2016), DenseNet (Huang et al. 2017), VGG (Simonyan & Zisserman, 2014), WideResNet (Zagoruyko & Komodakis, 2016), SqueezeNet (Iandola et al. 2016). Table 5.2 shows that our approach is systematically faster than the power method by a factor up to 12 when considering all the layers of the networks. This demonstrates the scalability of our method.

5.5 Lipschitz Regularization for Adversarial Robustness

One promising application of Lipschitz regularization is in the area of adversarial robustness. Empirical techniques to improve robustness against adversarial examples such as Adversarial Training only impact the training data, and often show poor generalization capabilities (Schmidt et al. 2018). Farnia et al. (2019) have shown that the adversarial generalization error depends on the Lipschitz constant of the network, which suggests that the adversarial test error can be improved by applying Lipschitz regularization in addition to adversarial training.

In this section, we illustrate the usefulness of LipBound by training a Wide ResNet (Zagoruyko & Komodakis, 2016) with Lipschitz regularization and adversarial training. Our regularization scheme is inspired by the one used by Yoshida & Miyato (2017) but instead of using the power method, we use our **PolyGrid** algorithm presented in Section 5.4.1 which efficiently computes an upper bound on the largest singular value of convolution layers.

We introduce the **AT+LipReg** loss to combine Adversarial Training and our Lipschitz regularization scheme in which layers with a large Lipschitz constant are penalized. We consider a neural network $N_\Omega : \mathcal{X} \to \mathcal{Y}$ with p layers $\phi^{(1)}_{\mathbf{W}^{(1)},\mathbf{b}^{(1)}}, \dots, \phi^{(p)}_{\mathbf{W}^{(p)},\mathbf{b}^{(p)}}$ where $\mathbf{W}^{(1)}, \dots, \mathbf{W}^{(p)}$ are the weight matrices and Ω is the union of all the parameters

as defined in Definition 2.5. Given a distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, we can train the parameters of the network by minimizing the AT+LipReg loss as follows:

$$\min_{\Omega} \mathbb{E}_{\mathbf{x},y\sim\mathcal{D}}\Big[L(N_\Omega(\mathbf{x} + \boldsymbol{\tau}_\Omega^{\text{adv}}(\mathbf{x})), y) + r(\Omega)\Big] \tag{5.55}$$

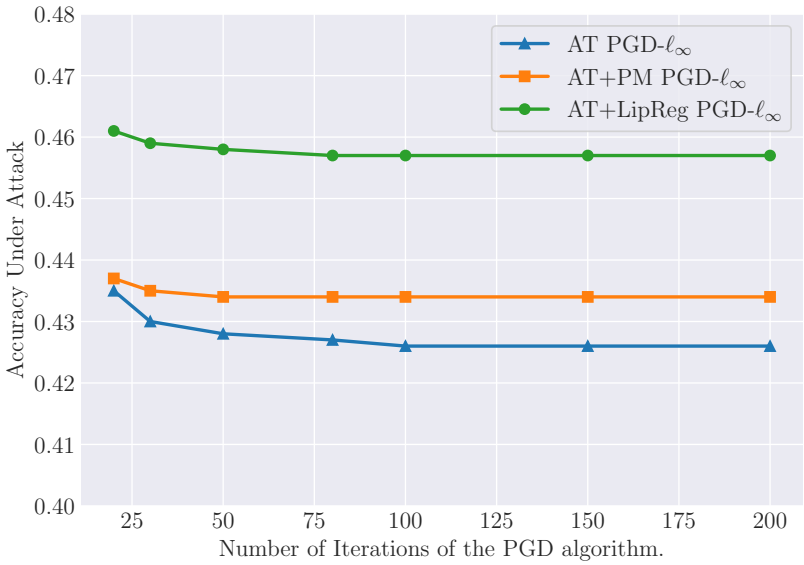
where L is the cross-entropy loss function, $\boldsymbol{\tau}_\Omega^{\text{adv}}(\mathbf{x})$ is an adversarial perturbation following the loss maximization strategy presented in Section 2.2.3 and the regularization function r is defined as follows:

$$r(\Omega) = C_1 \underbrace{\sum_{(\mathbf{W},\mathbf{b})\in\Omega} (\|\mathbf{W}\|_{\text{F}} + \|\mathbf{b}\|_2)}_{\ell_2 \text{ regularization}} + C_2 \underbrace{\sum_{i=1}^{p-1} \log(\text{LipBound}(\mathbf{K}_{\mathbf{W}^{(i)}}))}_{\text{Lipschitz regularization}} \tag{5.56}$$

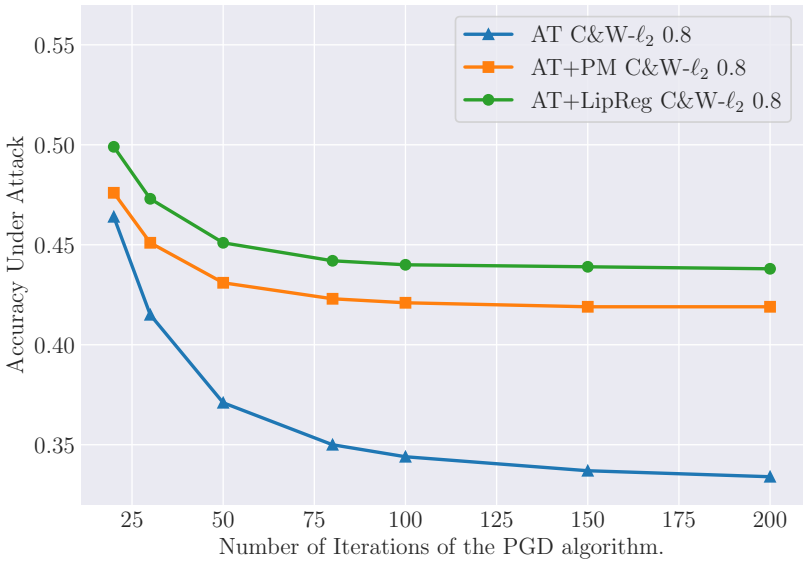
where C_1, C_2 are two user-defined hyper-parameters. Note that regularizing the sum of logs is equivalent to regularizing the product of all the LipBound which is an upper bound on the global Lipschitz constant. In practice, we also include the upper bound on the Lipschitz of the batch normalization since we can compute it very efficiently (see Appendix C.4.1 of Tsuzuku et al. (2018)) but we omit the last fully connected layer.

In this section, we compare the robustness of Adversarial Training (Goodfellow et al. 2015; Madry et al. 2018) against the combination of Adversarial Training and Lipschitz regularization. To regularize the Lipschitz constant of the network, we use the objective function defined in Equation 5.55. We train Lipschitz regularized neural networks with LipBound (see Theorem 5.5) implemented with PolyGrid (see Algorithm 5.1) (AT+LipBound) with $S = 10$ or with the specific power method for convolutions introduced by Farnia et al. (2019) with 10 iterations (AT+PM).

Table 5.3 shows the gain in robustness against strong adversarial attacks across different datasets. We can observe that both AT+LipBound and AT+PM offer a better defense against adversarial attacks and that AT+LipBound offers a further improvement over the Power Method. Figures 5.3a and 5.3b show the Accuracy under attack with different numbers of iterations of the PGD algorithm. Table 5.4 presents our results on the ImageNet Dataset. First, we can observe that the AT+LipReg trained networks offer a better generalization than with standalone Adversarial Training. Secondly, we can observe the gain in robustness against strong adversarial attacks. Network trained with Lipschitz regularization and Adversarial Training offer a consistent increase in robustness across ℓ_∞ and ℓ_2 attacks with different ϵ

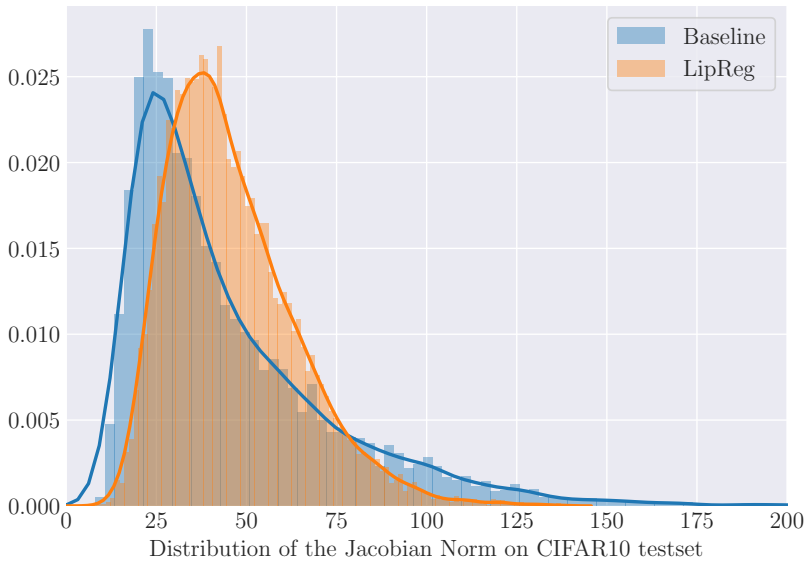


(a) Robustness against ℓ_∞ attacks for different classifiers trained with Adversarial Training given the number of iterations.

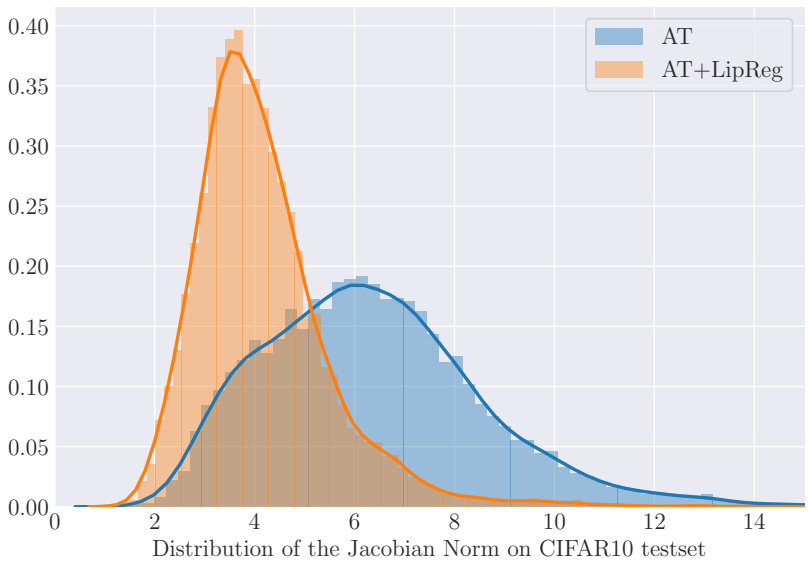


(b) Robustness against ℓ_∞ attacks for different classifiers trained with Adversarial Training given the number of iterations.

Figure 5.3: Accuracy under attack on CIFAR10 test set with ℓ_∞ and ℓ_2 attacks for several classifiers trained with Adversarial Training given the number of iterations.



(a) Comparison of the distribution of the norm of the Jacobian of the baseline model against the model trained with Lipschitz regularization.



(b) Comparison of the distribution of the norm of the Jacobian of the model trained with Adversarial training against the model trained with Adversarial training and Lipschitz regularization.

Figure 5.4: Distribution of the norm of the Jacobian matrix with respect to the CIFAR10 test set from a Wide ResNet trained with different schemes.

Model	Accuracy	PGD-ℓ_∞	C&W-ℓ_2 0.6	C&W-ℓ_2 0.8
Baseline	**0.953** $\pm$ 0.001	0.000 $\pm$ 0.000	0.002 $\pm$ 0.000	0.000 $\pm$ 0.000
AT	0.864 $\pm$ 0.001	0.426 $\pm$ 0.000	0.477 $\pm$ 0.000	0.334 $\pm$ 0.000
AT+PM	0.788 $\pm$ 0.010	0.434 $\pm$ 0.007	0.521 $\pm$ 0.005	0.419 $\pm$ 0.003
AT+LipReg	0.808 $\pm$ 0.022	**0.457** $\pm$ 0.002	**0.547** $\pm$ 0.022	**0.438** $\pm$ 0.020

(a) Results on CIFAR10 dataset

Model	Accuracy	PGD-ℓ_∞	C&W-ℓ_2 0.6	C&W-ℓ_2 0.8
Baseline	**0.792** $\pm$ 0.000	0.000 $\pm$ 0.000	0.001 $\pm$ 0.000	0.000 $\pm$ 0.000
AT	0.591 $\pm$ 0.000	0.199 $\pm$ 0.000	0.263 $\pm$ 0.000	0.183 $\pm$ 0.000
AT+LipReg	0.552 $\pm$ 0.019	**0.215** $\pm$ 0.004	**0.294** $\pm$ 0.010	**0.226** $\pm$ 0.008

(b) Results on CIFAR100 dataset

Table 5.3: Accuracy under ℓ_2 and ℓ_∞ attacks of different training schemes on CIFAR10/100 datasets.

value. We can also note that increasing the regularization leads to an increase in generalization and robustness.

Finally, we also conducted an experiment to study the impact of the regularization on the gradients of the whole network by measuring the distributions of the norm of the Jacobian matrix with respect to the inputs from the test set. The results of this experiment are presented in Figure 5.4a and show more concentrated gradients with Lipschitz regularization. Indeed, we can observe that while the median is higher, the regularization decreases the number of points with very high Lipschitz constant. Although Lipschitz regularization is not a Jacobian regularization, we can observe a clear shift in the distribution. This suggests that our method does not only work layer-wise, but also at the level of the entire network. A second experiment, using Adversarial Training, presented in Figure 5.4b demonstrates that the effect is even stronger when the two techniques are combined together. It also demonstrates that Lipschitz regularization and Adversarial Training (or other Jacobian regularization techniques) are complementary. Hence they offer an increased robustness to adversarial attacks as demonstrated above.

Experimental Settings CIFAR10/100 Dataset. For all our experiments, we use the Wide ResNet architecture introduced by Zagoruyko & Komodakis (2016) to train our classifiers. We use Wide ResNet networks with 28 layers and a width factor of 10. We train our networks for 200 epochs with a batch size of 200. We

Model	Natural	PGD-ℓ_∞		C&W-ℓ_2		
		0.02	0.031	1.00	2.00	3.00
Baseline (He et al. 2016)	**0.782**	0.000	0.000	0.000	0.000	0.000
AT	0.509	0.251	0.118	0.307	0.168	0.099
AT+LipReg ($C_2 = 0.0006$)	**0.515**	**0.255**	**0.121**	**0.316**	**0.177**	**0.105**
AT+LipReg ($C_2 = 0.0010$)	**0.519**	**0.259**	**0.123**	**0.338**	**0.204**	**0.129**

Table 5.4: Natural accuracy and accuracy under ℓ_2 and ℓ_∞ attacks of different training schemes on the ImageNet dataset.

use Stochastic Gradient Descent with a momentum of 0.9, an initial learning rate of 0.1 with exponential decay of 0.1 (MultiStepLR gamma = 0.1) after the epochs 60, 120 and 160. For Adversarial Training (Madry et al. 2018), we use Projected Gradient Descent with an $\epsilon = 8/255(\approx 0.031)$, a step size of $\epsilon/5(\approx 0.0062)$ and 10 iterations, we use a random initialization but run the attack only once. To evaluate the robustness of our classifiers, we rigorously followed the experimental protocol proposed by Carlini et al. (2019) and Tramer et al. (2020). More precisely, as an ℓ_∞ attack, we use PGD with the same parameters ($\epsilon = 8/255$, a step size of $\epsilon/5$) but we increase the number of iterations up to 200 with 10 restarts. For each image, we select the perturbation that maximizes the loss among all the iterations and the 10 restarts. As ℓ_2 attacks, we use a bounded version of the Carlini & Wagner (2017) attack. We choose 0.6 and 0.8 as bounds for the ℓ_2 perturbation. Note that the ℓ_2 ball with a radius of 0.8 has approximately the same volume as the ℓ_∞ ball with a radius of 0.031 for the dimensionality of CIFAR10/100.

Experimental Settings for ImageNet Dataset. For all our experiments, we use the ResNet-101 architecture (He et al. 2016). We have used Stochastic Gradient Descent with a momentum of 0.9, a weight decay of 0.0001, label smoothing of 0.1, an initial learning rate of 0.1 with exponential decay of 0.1 (MultiStepLR gamma = 0.1) after the epochs 30 and 60. We have used Exponential Moving Average over the weights with a decay of 0.999. We have trained our networks for 80 epochs with a batch size of 4096. For Adversarial Training, we have used PGD with 5 iterations, $\epsilon = 8/255(\approx 0.031)$ and a step size of $\epsilon/5(\approx 0.0062)$. To evaluate the robustness of our classifiers on ImageNet Dataset, we have used an ℓ_∞ and an ℓ_2 attacks. More precisely, as an ℓ_∞ attack, we use PGD with an epsilon of 0.02 and 0.031, a step size of $\epsilon/5$) with a number of iterations to 30 with 5 restarts. For each image, we select the perturbation that maximizes the loss among all the iterations and the 10 restarts.

As ℓ_2 attacks, we use a bounded version of the Carlini & Wagner (2017) attack. We have used 1, 2 and 3 as bounds for the ℓ_2 perturbation.

5.6 Concluding Remarks

In this chapter, we introduced a new bound on the Lipschitz constant of convolution layers that is both accurate and efficient to compute. We used this bound to regularize the Lipschitz constant of neural networks and demonstrated its computational efficiency in training large neural networks with a regularized Lipschitz constant. As an illustrative example, we combined our bound with adversarial training, and showed that this increases the robustness of the trained networks to adversarial attacks. The scope of our results goes beyond this application and can be used in a wide variety of settings, for example, to stabilize the training of Generative Adversarial Networks (GANs) and invertible networks, or to improve generalization capabilities of classifiers.

Chapter 6

Conclusion

Contents

6.1 Summary of the Contributions

State-of-the-art in a variety of domains, deep neural networks exhibit important limitations. Indeed, current neural networks tend to be very large in terms of their number of parameters which make them difficult to train and to deploy in real-world applications. Furthermore, they exhibit instability to small perturbations of their inputs which lead to adversarial attacks.

In this thesis, we have used structured matrices from the Toeplitz family to make contributions to the field of deep learning. Our contributions are twofold. First, we studied deep diagonal-circulant neural networks, which are deep neural networks in which weight matrices are the product of diagonal and circulant ones. Using diagonal and circulant matrices instead of dense ones allows for an important reduction in the number of parameters which make them more efficient and cost-effective. In addition to being more compact than fully connected neural networks, diagonal-circulant neural

networks have a high expressivity that makes them useful for numerous use cases. In order to characterize the expressive power of diagonal-circulant neural networks, we build upon the work of Huhtanen & Perämäki (2015) which states that any matrix can be decomposed into a product of alternating diagonal and circulant matrices. Based on this result, we have successfully demonstrated that neural networks with diagonal and circulant matrices are *universal approximators* and characterized their expressive power with respect to their depth. We also demonstrated the effectiveness of this class of compact neural networks to video classification with a real-world dataset.

Secondly, we studied the properties of doubly-block Toeplitz matrices which are equivalent to the convolution operation. Using the properties of this type of structured matrix and a Fourier representation introduced by Grenander et al. (1958), we devised an upper-bound on the singular values of convolution layers leading to a new regularization scheme that improves the robustness of neural networks against adversarial attacks. In order to use this upper-bound in a large-scale setting, we introduced the PolyGrid algorithm (see Algorithm 5.1) which efficiently and accurately computes an approximation of this upper-bound.

6.2 Perspectives and Future Works

6.2.1 Designing Compact Transformers for Natural Language Processing

In order to improve upon our work on compact neural networks, one idea follows naturally. The race towards larger convolutional neural networks seemed to have slowed down following the work of Tan & Le (2019) which devised compact state-of-the-art neural networks for image recognition. However, other types of architecture, *e.g.*, *Transformers* which rely heavily on dense matrices, have seen their number of parameters exploding in recent years. The latest model which was designed by Fedus et al. (2021) has 1 trillion parameters, 5.7 times than the second largest, proposed by Brown et al. (2020), which had 175 billion parameters.

In Chapter 4 and Appendix B, we have used the diagonal-circulant decomposition for compressing embedded layers in the context of video classification. This decomposition could also be used to compress attention layers of Transformers networks (Vaswani et al. 2017) where the attention layer is described as follows:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V} \ , \tag{6.1}$$

where $\mathbf{Q}, \mathbf{K}$ and $\mathbf{V}$ are dense matrices. Taking this layer as a building block leads to large neural networks as demonstrated by the GPT-3 architecture with 96 attention layers and 175 billion parameters. Although, the diagonal-circulant decomposition could successfully reduce the number of parameters of attention layers, it may have limited impact on *multi-head attention layers* which are a concatenation of small attention layers due to the reduced dimension of each matrix.

6.2.2 Regularization on the Condition Number of Convolution Layers

In Chapter 5, we have proposed an upper-bound on the largest singular value of convolution layers which allow us to regularize the Lipschitz constant of the network thus improving the robustness. However, an important reduction of the Lipschitz constant seems to prevent the network from learning correctly. Indeed, as demonstrated by the work of (Zhang et al. 2019), accuracy and robustness are actually at odds, meaning that improving the robustness (*i.e.* in our case, reducing the expressivity) hurts the training and the natural accuracy of the network. In our experiments, the phenomenon called *rank collapse* (Saxe et al. 2014) where the rank of the weights matrices tend to decrease during training combined with a strong Lipschitz regularization would prevent convergence. An interesting solution would be to regularize the largest singular value and promoting the smallest in order to enforce orthogonality. The following bound on the condition number of general matrices Guggenheimer et al. (1995) could be studied:

$$\kappa(\mathbf{W}) \leq \frac{2}{|\det(\mathbf{W})|}\left(\frac{\|\mathbf{W}\|_{\mathrm{F}}}{\sqrt{r}}\right)^{r} \tag{6.2}$$

where r is the rank of $\mathbf{W}$. As such, using the bound as a regularizer will enforce the orthogonality, just like a layer normalization. Therefore, we could design some heuristic regularizers to encourage a smaller $\left(\frac{\|\mathbf{W}\|_{\mathrm{F}}}{\sqrt{r}}\right)^{r}$ and larger $|\det(\mathbf{W})|$ separately, as given by the following objective function:

$$\min_{\Omega} \mathbb{E}_{\mathbf{x},y\sim\mathcal{D}}\left[L(N_{\Omega}(\mathbf{x}), y) + C_1 \sum_{i=1}^{p} \left\|\mathbf{W}^{(i)}\right\|_{\mathrm{F}} + C_2 \sum_{i=1}^{p} \log\left|\det\left(\mathbf{W}^{(i)}\right)\right|\right] \tag{6.3}$$

where the determinant of doubly-block Toeplitz matrices under some assumption could be expressed with the Szegö Theorem (Szegö, 1915) and can be approximated with the help of *Random Matrix Theory* (Basor, 2017).

6.2.3 Going Beyond the Lipschitz Constant

Finally, in order to better understand the behavior of neural networks and the transformation they perform, it would be interesting to go beyond the Lipschitz constant and consider their full spectrum. Indeed, the spectrum of a linear map is a set that contains the eigenvalues and can be seen as a description of the properties and behavior of the operator. For example, for a linear operator $\mathbf{L} : \mathcal{X} \to \mathcal{X}$, the spectrum gives precise information on the solvability of the following linear equation

$$\lambda \mathbf{x} - \mathbf{L}\mathbf{x} = \mathbf{y} \tag{6.4}$$

It is natural to ask if we could define a spectrum that equivalently gives information on the following nonlinear equation

$$\lambda \mathbf{x} - \mathbf{F}(\mathbf{x}) = \mathbf{y} \ . \tag{6.5}$$

In this line of research, Kachurovskii (1969) have defined a spectrum for nonlinear continuous Lipschitz operators which share important properties with the spectrum of linear operators. More precisely, let $\mathbf{F} : \mathcal{X} \to \mathcal{X}$ be a nonlinear continuous Lipschitz map, the *Kachurovskij spectrum* of $\mathbf{F}$ is given by

$$\sigma(\mathbf{F}) \triangleq \{\lambda \in \mathbb{C} \mid \lambda \mathbf{I} - \mathbf{F} \text{ is not a lipeomorphism}\} \tag{6.6}$$

where a nonlinear Lipschitz continuous operator is a *lipeomorphism* if its inverse is also nonlinear Lipschitz continuous. We can also define the complement of the spectrum, *i.e.*, the *Kachurovskij resolvent set* as follows:

$$\mu(\mathbf{F}) \triangleq \mathbb{C} \setminus \sigma(\mathbf{F}) \tag{6.7}$$

The resolvent set can be seen as the set of complex numbers for which the operator is *well behaved*. The Kachurovskij spectrum is a compact subset of the complex plane but may be empty. Kachurovskij have also shown that the emptiness of this spectrum can be prevented if we restrict ourselves to nonlinear continuous Lipschitz operators that admit a Fréchet-derivative $\mathbf{F}'(\mathbf{x}_0)$ at some point $\mathbf{x}_0 \in \mathcal{X}$. In this case, the Kachurovskij spectrum share all the properties of a linear operator which are: closed, compact, bounded and non-empty.

Neural networks with differentiable nonlinearities are differentiable nonlinear Lipschitz continuous functions, therefore the study of the Kachurovskij spectrum could

give important insight on their stability, invertibility and robustness to adversarial examples.

6.3 Discussion

Although our contributions offer concrete techniques for building compact and reliable neural networks, they also highlight some important difficulties in them. First, if we discard techniques such as pruning or quantization for building compact neural networks due to the necessity of training a large neural network prior to compression, designing parameters-efficient neural networks that are compact *by design* requires rethinking the whole architecture. For computer vision tasks, the convolution operation is a compact and powerful transform, however, we still haven't found such equivalent transforms for other use cases. Although the multi-head attention layer is more efficient than the attention layer for NLP tasks, using this type of transform in a neural network is still very parameter-hungry as demonstrated by the recent state-of-the-art for language models (Brown et al. 2020).

Secondly, defense techniques against adversarial attacks have shown great improvements in the last few years. However, with current state-of-the-art techniques, it is still difficult to reach an accuracy higher than 60% on CIFAR10 (which is considered a small dataset) and the accuracy decreases further on datasets with a larger dimensionality. Consequently, building robust neural networks still remain very much an open question. We believe that further breakthroughs in this area will come as a by-product on research on *understanding neural networks*. Accordingly, we hope that our contribution to the understanding of diagonal-circulant and convolution neural networks is a small step in this direction.

Appendix

Appendices

Appendix A

Generalization of Widom Identity

This appendix aims at proving a generalization of Widom Identity for doubly-block Toeplitz operators. The Widom identity, which states the relation between Toeplitz and Hankel operators, was introduced by Harold Widom in a 1976 seminal paper (Widom, 1976). Let us define the semi-infinite Toeplitz and Hankel operators:

$$\mathbf{T}_\infty(f) \triangleq \left(\frac{1}{2\pi}\int_0^{2\pi} e^{-\mathbf{i}(i-j)\omega} f(\omega)\ \mathrm{d}\omega\right)_{i,j\in\{0,\ldots,\infty\}} \tag{A.1}$$

$$\mathbf{H}_\infty(f) \triangleq \left(\frac{1}{2\pi}\int_0^{2\pi} e^{-\mathbf{i}(i+j+1)\omega} f(\omega)\ \mathrm{d}\omega\right)_{i,j\in\{0,\ldots,\infty\}} \tag{A.2}$$

Then, for f and g integrable functions, the Widom identity can be written as follows:

$$\mathbf{T}_\infty(fg) - \mathbf{T}_\infty(f)\mathbf{T}_\infty(g) = \mathbf{H}_\infty(f)\mathbf{H}_\infty(g^*) \tag{A.3}$$

Note that Widom extends this identity from finite Toeplitz matrices:

$$\mathbf{T}_n(fg) - \mathbf{T}_n(f)\mathbf{T}_n(g) = \mathbf{H}_n(f)\mathbf{H}_n(g^*) - \mathbf{J}_n\mathbf{H}_n(f^*)\mathbf{H}_n(g^*)\mathbf{J}_n \tag{A.4}$$

where $\mathbf{J}_n$ is the anti-identity matrix, *i.e.*, the reflexion matrix.

We would like to expend the identity presented in Equation (A.4) to finite doubly-block Toeplitz operator. We will need to generalize the doubly-block Toeplitz operator presented in Section 5.2.2. Let $\mathbf{G}^{\alpha_p}(f) = \left(\mathbf{G}^{\alpha_p}_{i,j}(f)\right)_{i,j\in\mathcal{I}_n^+}$ where $\mathbf{G}^{\alpha_p}_{i,j}$ is defined as:

$$\mathbf{G}^{\alpha_p}_{i,j}(f) = \left(\frac{1}{4\pi^2}\int_0^{2\pi}\int_0^{2\pi} e^{-\mathbf{i}\alpha_p(i,j,k,l,\omega_1,\omega_2)} f(\omega_1,\omega_2)\ \mathrm{d}\omega_1\ \mathrm{d}\omega_2)\right)_{k,l\in\mathcal{I}_n^+}. \tag{A.5}$$

Note that as with the operator $\mathbf{D}(f)$ we only consider generating functions as trigonometric polynomials with real coefficients therefore the matrices generated

by $\mathbf{G}(f)$ are real. And as with the operator $\mathbf{D}(f)$, the matrices generated by the operator $\mathbf{G}^{\alpha_p}$ are of size $n^2 \times n^2$.
We will use the following α functions:

$$\alpha_0(i,j,k,l,\omega_1,\omega_2) = (-j-i-1)\omega_1 + (k-l)\omega_2$$

$$\alpha_1(i,j,k,l,\omega_1,\omega_2) = (i-j)\omega_1 + (-l-k-1)\omega_2$$

$$\alpha_2(i,j,k,l,\omega_1,\omega_2) = (-j-i-1)\omega_1 + (-l-k-1)\omega_2$$

$$\alpha_3(i,j,k,l,\omega_1,\omega_2) = (-j-i+n)\omega_1 + (-l-k-1)\omega_2$$

We now present the generalization of the Widom identity for Doubly-Block Toeplitz matrices below:

Lemma A.1 (Generalization of Widom Identity)**.** *Let $f : \mathbb{R}^2 \to \mathbb{C}$ and $g : \mathbb{R}^2 \to \mathbb{C}$ be two continuous and 2π-periodic functions. We can decompose the Doubly-Block Toeplitz matrix $\mathbf{D}(fg)$ as follows:*

$$\mathbf{D}(fg) = \mathbf{D}(f)\mathbf{D}(g) + \sum_{p=0}^{3} \mathbf{G}^{\alpha_p\top}(f^*)\mathbf{G}^{\alpha_p}(g) + \mathbf{J}_{n^2}\left(\sum_{p=0}^{3} \mathbf{G}^{\alpha_p\top}(f)\mathbf{G}^{\alpha_p}(g^*)\right)\mathbf{J}_{n^2}. \quad \text{(A.6)}$$

where $\mathbf{J}$ is the reflection of the identity matrix of size $n^2 \times n^2$.

Proof of Lemma A.1. Let (i,j) be matrix indexes such $(\,\cdot\,)_{i,j}$ correspond to the value at the ith row and j^{th} column, let us define the following notation:

$$i_1 = \lfloor i/n \rfloor \qquad j_1 = \lfloor j/n \rfloor$$
$$i_2 = i \bmod n \qquad j_2 = j \bmod n$$

Let us define $\hat{f}$ as the 2 dimensional Fourier transform of the function f. We refer to $\hat{f}_{h_1,h_2}$ as the Fourier coefficient indexed by (h_1,h_2) where h_1 correspond to the

index of the block of the doubly-block Toeplitz and h_2 correspond to the index of the value inside the block. More precisely, we have

$$(\mathbf{D}(f))_{i,j} = \hat{f}_{(\lfloor j/n \rfloor - \lfloor i/n \rfloor),((j \bmod n)-(i \bmod n)))} \tag{A.7}$$
$$(\mathbf{G}^{\alpha_0}(f))_{i,j} = \hat{f}_{(\lfloor j/n \rfloor + \lfloor i/n \rfloor + 1),((j \bmod n)-(i \bmod n)))} \tag{A.8}$$
$$(\mathbf{G}^{\alpha_1}(f))_{i,j} = \hat{f}_{(\lfloor j/n \rfloor - \lfloor i/n \rfloor),((j \bmod n)+(i \bmod n)+1))} \tag{A.9}$$
$$(\mathbf{G}^{\alpha_2}(f))_{i,j} = \hat{f}_{(\lfloor j/n \rfloor - \lfloor i/n \rfloor),((j \bmod n)-(i \bmod n)))} \tag{A.10}$$
$$(\mathbf{G}^{\alpha_3}(f))_{i,j} = \hat{f}_{(\lfloor j/n \rfloor + \lfloor i/n \rfloor + n),((j \bmod n)+(i \bmod n)+1))} \tag{A.11}$$

We simplify the notation of the expressions above as follow:

$$(\mathbf{D}(f))_{i,j} = \hat{f}_{(j_1-i_1),(j_2-i_2)} \tag{A.12}$$
$$(\mathbf{G}^{\alpha_0}(f))_{i,j} = \hat{f}_{(j_1+i_1+1),(j_2-i_2)} \tag{A.13}$$
$$(\mathbf{G}^{\alpha_1}(f))_{i,j} = \hat{f}_{(j_1-i_1),(j_2+i_2+1)} \tag{A.14}$$
$$(\mathbf{G}^{\alpha_2}(f))_{i,j} = \hat{f}_{(j_1-i_1),(j_2-i_2)} \tag{A.15}$$
$$(\mathbf{G}^{\alpha_3}(f))_{i,j} = \hat{f}_{(j_1+i_1+n),(j_2+i_2+1)} \tag{A.16}$$

The convolution theorem states that the Fourier transform of a product of two functions is the convolution of their Fourier coefficients. Therefore, one can observe that the entry (i,j) of the matrix $\mathbf{D}(fg)$ can be express as follows:

$$(\mathbf{D}(fg))_{i,j} = \sum_{k_1=-2n+1}^{2n-1} \sum_{k_2=-2n+1}^{2n-1} \hat{f}_{(k_1-i_1),(k_2-i_2)} \hat{g}_{(j_1-k_1),(j_2-k_2)}.$$

By splitting the double sums and simplifying, we obtain:

$$\begin{aligned}(\mathbf{D}(fg))_{i,j} = \sum_{k_1,k_2 \in P} \Big(& \hat{f}_{(k_1-i_1),(k_2-i_2)} \hat{g}_{(j_1-k_1),(j_2-k_2)} + \hat{f}_{(-k_1-i_1-1),(k_2-i_2)} \hat{g}_{(j_1+k_1+1),(j_2-k_2)} \\ & + \hat{f}_{(k_1-i_1),(-k_2-i_2-1)} \hat{g}_{(j_1-k_1),(j_2+k_2+1)} + \hat{f}_{(-k_1-i_1-1),(-k_2-i_2-1)} \hat{g}_{(j_1+k_1+1),(j_2+k_2+1)} \\ & + \hat{f}_{(k_1-i_1+n),(-k_2-i_2-1)} \hat{g}_{(j_1-k_1-n),(j_2+k_2+1)} + \hat{f}_{(k_1-i_1+n),(k_2-i_2)} \hat{g}_{(j_1-k_1-n),(j_2-k_2)} \\ & + \hat{f}_{(k_1-i_1),(k_2-i_2+n)} \hat{g}_{(j_1-k_1),(j_2-k_2-n)} + \hat{f}_{(k_1-i_1+n),(k_2-i_2+n)} \hat{g}_{(j_1-k_1-n),(j_2-k_2-n)} \\ & + \hat{f}_{(-k_1-i_1-1),(k_2-i_2+n)} \hat{g}_{(j_1+k_1+1),(j_2-k_2-n)} \Big)\end{aligned} \tag{A.17}$$

where $P = \{(k_1,k_2) \mid k_1,k_2 \in \mathbb{Z}, 0 \le k_1 \le n-1, 0 \le k_2 \le n-1\}$.

Furthermore, we can observe the following:

$$(\mathbf{D}(f)\mathbf{D}(g))_{i,j} = \sum_{k=0}^{n^2} (\mathbf{D}(f))_{i,k} (\mathbf{D}(g))_{k,j} = \sum_{k_1,k_2 \in P} \hat{f}_{(k_1-i_1),(k_2-i_2)} \hat{g}_{(j_1-k_1),(j_2-k_2)}$$

$$\begin{aligned}\left(\mathbf{G}^{\alpha_1\top}(f^*)\mathbf{G}^{\alpha_1}(g)\right)_{i,j} &= \sum_{k_1,k_2 \in P} \hat{f}^*_{(k_1+i_1+1),(i_2-k_2)} \hat{g}_{(j_1+k_1+1),(j_2-k_2)} \\ &= \sum_{k_1,k_2 \in P} \hat{f}_{(-k_1-i_1-1),(k_2-i_2)} \hat{g}_{(j_1+k_1+1),(j_2-k_2)}\end{aligned}$$

$$\begin{aligned}\left(\mathbf{G}^{\alpha_2\top}(f^*)\mathbf{G}^{\alpha_2}(g)\right)_{i,j} &= \sum_{k_1,k_2 \in P} \hat{f}^*_{(i_1-k_1),(k_2+i_2+1)} \hat{g}_{(j_1-k_1),(j_2+k_2+1)} \\ &= \sum_{k_1,k_2 \in P} \hat{f}_{(k_1-i_1),(-k_2-i_2-1)} \hat{g}_{(j_1-k_1),(j_2+k_2+1)}\end{aligned}$$

$$\begin{aligned}\left(\mathbf{G}^{\alpha_3\top}(f^*)\mathbf{G}^{\alpha_3}(g)\right)_{i,j} &= \sum_{k_1,k_2 \in P} \hat{f}^*_{(k_1+i_1+1),(k_2+i_2+1)} \hat{g}_{(j_1+k_1+1),(k_2+j_2+1)} \\ &= \sum_{k_1,k_2 \in P} \hat{f}_{(-k_1-i_1-1),(-k_2-i_2-1)} \hat{g}_{(j_1+k_1+1),(k_2+j_2+1)}\end{aligned}$$

$$\begin{aligned}\left(\mathbf{G}^{\alpha_4\top}(f^*)\mathbf{G}^{\alpha_4}(g)\right)_{i,j} &= \sum_{k_1,k_2 \in P} \hat{f}^*_{(i_1-k_1-n),(k_2+i_2+1)} \hat{g}_{(j_1-k_1-n),(j_2+k_2+1)} \\ &= \sum_{k_1,k_2 \in P} \hat{f}_{(k_1-i_1+n),(-k_2-i_2-1)} \hat{g}_{(j_1-k_1-n),(j_2+k_2+1)}\end{aligned}$$

Let us define the matrix $\mathbf{J}_{n^2}$ of size $n^2 \times n^2$ as the anti-identity matrix. We have the following:

$$\begin{aligned}\left(\mathbf{G}^{\alpha_1\top}(f)\mathbf{G}^{\alpha_1}(g^*)\right)_{i,j} &= \sum_{k_1,k_2 \in P} \hat{f}_{(k_1+i_1+1),(i_2-k_2)} \hat{g}^*_{(j_1+k_1+1),(j_2-k_2)} \\ &= \sum_{k_1,k_2 \in P} \hat{f}_{(k_1+i_1+1),(i_2-k_2)} \hat{g}_{(-j_1-k_1-1),(k_2-j_2)} \\ \Leftrightarrow \left(\mathbf{J}_{n^2}\mathbf{G}^{\alpha_1\top}(f)\mathbf{G}^{\alpha_1}(g^*)\mathbf{J}_{n^2}\right)_{i,j} &= \sum_{k_1,k_2 \in P} \hat{f}_{(k_1-i_1+n),(k_2-i_2)} \hat{g}_{(j_1-k_1-n),(j_2-k_2)}\end{aligned}$$

$$
\begin{aligned}
\left(\mathbf{G}^{\alpha_2\top}(f)\mathbf{G}^{\alpha_2}(g^*)\right)_{i,j} &= \sum_{k_1,k_2\in P} \hat{f}_{(i_1-k_1),(k_2+i_2+1)}\hat{g}^*_{(j_1-k_1),(j_2+k_2+1)} \\
&= \sum_{k_1,k_2\in P} \hat{f}_{(i_1-k_1),(k_2+i_2+1)}\hat{g}_{(k_1-j_1),(-j_2-k_2-1)} \\
\Leftrightarrow \left(\mathbf{J}^{n^2}\mathbf{G}^{\alpha_2\top}(f)\mathbf{G}^{\alpha_2}(g^*)\mathbf{J}_{n^2}\right)_{i,j} &= \sum_{k_1,k_2\in P} \hat{f}_{(k_1-i_1),(k_2-i_2+n)}\hat{g}_{(j_1-k_1),(j_2-k_2-n)}
\end{aligned}
$$

$$
\begin{aligned}
\left(\mathbf{G}^{\alpha_3\top}(f)\mathbf{G}^{\alpha_3}(g^*)\right)_{i,j} &= \sum_{k_1,k_2\in P} \hat{f}_{(k_1+i_1+1),(k_2+i_2+1)}\hat{g}^*_{(j_1+k_1+1),(k_2+j_2+1)} \\
&= \sum_{k_1,k_2\in P} \hat{f}_{(k_1+i_1+1),(k_2+i_2+1)}\hat{g}_{(-j_1-k_1-1),(-k_2-j_2-1)} \\
\Leftrightarrow \left(\mathbf{J}_{n^2}\mathbf{G}^{\alpha_3\top}(f)\mathbf{G}^{\alpha_3}(g^*)\mathbf{J}_{n^2}\right)_{i,j} &= \sum_{k_1,k_2\in P} \hat{f}_{(k_1-i_1+n),(k_2-i_2+n)}\hat{g}_{(j_1-k_1-n),(-k_2+j_2-n)}
\end{aligned}
$$

$$
\begin{aligned}
\left(\mathbf{G}^{\alpha_4\top}(f)\mathbf{G}^{\alpha_4}(g^*)\right)_{i,j} &= \sum_{k_1,k_2\in P} \hat{f}_{(-k_1+i_1-n),(k_2+i_2+1)}\hat{g}^*_{(j_1-k_1-n),(j_2+k_2+1)} \\
&= \sum_{k_1,k_2\in P} \hat{f}_{(-k_1+i_1-n),(k_2+i_2+1)}\hat{g}_{(-j_1+k_1+n),(-j_2-k_2-1)} \\
\Leftrightarrow \left(\mathbf{J}_{n^2}\mathbf{G}^{\alpha_4\top}(f)\mathbf{G}^{\alpha_4}(g^*)\mathbf{J}_{n^2}\right)_{i,j} &= \sum_{k_1,k_2\in P} \hat{f}_{(-k_1-i_1-1),(k_2-i_2+n)}\hat{g}_{(j_1+k_1+1),(j_2-k_2-n)}
\end{aligned}
$$

Now, we can observe from Equation A.17 that:

$$
\mathbf{D}(fg) = \mathbf{D}(f)\mathbf{D}(g) + \sum_{p=0}^{3} \mathbf{G}^{\alpha_p\top}(f^*)\mathbf{G}^{\alpha_p}(g) + \mathbf{J}_{n^2}\left(\sum_{p=0}^{3} \mathbf{G}^{\alpha_p\top}(f)\mathbf{G}^{\alpha_p}(g^*)\right)\mathbf{J}_{n^2}. \tag{A.18}
$$

which concludes the proof. ■

Appendix B

Diagonal Circulant Neural Networks for Video Classification

Contents

This Appendix reports some additional experiments on video classification with diagonal-circulant neural networks. These experiments have been done in the context of the YouTube-8M[1] video classification challenge. This work was recognized as one of the 5 original approaches by the Google AI team that organized the workshop (Lee et al. 2018).

[1] https://www.kaggle.com/c/youtube8m

B.1 Introduction

Classification of unlabeled videos streams is one of the challenging tasks for machine learning algorithms. Research in this field has been stimulated by the recent release of several large annotated video datasets such as *Sports-1M* (Karpathy et al. 2014), *FCVID* (Jiang et al. 2018) or the *YouTube-8M* (Abu-El-Haija et al. 2016) dataset.

The naive approach to achieve video classification is to perform frame-by-frame image recognition, and to average the results before the classification step. However, it has been shown by Abu-El-Haija et al. (2016) and Miech et al. (2017) that better results can be obtained by building features across different frames and several deep learning architectures have been designed to learn embeddings for sets of frames. For example Deep Bag-of-Frames (DBoF) (Abu-El-Haija et al. 2016), NetVLAD (Arandjelović et al. 2016) or architectures based on Fisher Vectors (Perronnin & Dance, 2007).

The DBoF embedding layer processes videos in two steps. First, a learned transformation projects all the frames together into a high dimensional space. Then, a max (or average) pooling operation aggregates all the embedded frames into a single discriminative vector representation of the video. The NetVLAD embedding layer is built on a technique called *vector of locally aggregated descriptors* (VLAD) (Jégou et al. 2010). This technique that aggregates a large number of local frame descriptors into a compact representation using a codebook of visual words. In NetVlad, the codebook is directly learned end-to-end during training. Finally, NetFisherVector (NetFV) is inspired by Perronnin & Dance (2007) and uses first and second-order statistics as video descriptors also gathered in a codebook. The technique can benefit from deep learning by using a deep neural network to learn the codebook (Miech et al. 2017).

All the architectures mentioned above can be used to build video features in the sense of features that span across several frames, but they are not designed to exploit the sequential nature of videos and capture motion. In order to truly learn spatio-temporal features and account for motion in videos, several researchers have looked into recurrent neural networks (*e.g.* LSTM (Yue-Hei Ng et al. 2015; Li et al. 2017b)) and 3D convolutions (Karpathy et al. 2014) (in space and time). However, these approaches do not outperform non-sequential models, and the single best model proposed by Miech et al. (2017) (winner of the first *YouTube-8M* competition) is based on NetVLAD.

The *2nd YouTube-8M Video Understanding Challenge* includes a constraint on the model size and many competitors have been looking into building efficient memory

models with high accuracy. There are two kinds of techniques to reduce the memory required for training and/or inference in neural networks. The first kind aims at *compressing* an existing neural network into a smaller one, (thus it only impacts the size of the model at inference time). The second one aims at *constructing models that are compact* by design.

B.2 Compact Architecture using Diagonal and Circulant Matrices

Building on the matrix decomposition presented in Chapter 4, we introduce a compact neural network architecture for video classification where dense matrices have been replaced by products of circulant and diagonal matrices.

B.2.1 Base Model

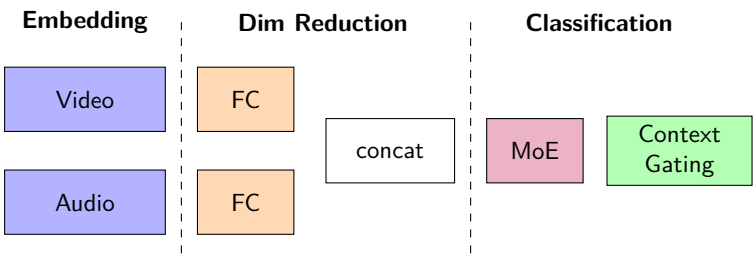


Figure B.1: Diagram of the architecture proposed by Miech et al. (2017) and used for the experiments. The sample goes through an embedding layer and is reduced with a Fully Connected layer. The results are then concatenated and classified with a Mixture-of-Experts and Context Gating layer.

We demonstrate the benefit of the decomposition into diagonal and circulant matrices using a base model which has been proposed by Miech et al. (2017). This architecture can be decomposed into three blocks of layers, as illustrated in Figure B.1. The first block of layers, composed of the Deep Bag-of-Frames embedding, is meant to process audio and video frames independently. The DBoF layer computes two embeddings: one for the audio and one for the video. In the next paragraph, we will only focus on describing the video embedding (The audio embedding is computed in a very similar way). We represent a video $\mathbf{V}$ as a set of m frames $\{\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(m)}\}$ where each frame $\mathbf{v}^{(i)} \in \mathbb{R}^k$ is a vector of visual features extracted from the frame image. In the context of the *YouTube-8M* competition, each $\mathbf{v}_i$ is a vector of 1024

visual features extracted using the last fully connected layer of an Inception network trained on ImageNet. The DBoF layer then embeds a video $\mathbf{V}$ into a vector $\mathbf{v}'$ drawn from a $\mathbf{p}$ dimensional vector space as follows:

$$\mathbf{v}' = \max\Big\{\mathbf{W}\mathbf{v}^{(i)} \mid \mathbf{v}^{(i)} \in \mathbf{V}\Big\} \tag{B.1}$$

where $\mathbf{W}$ is a matrix in $\mathbb{R}^{p\times k}$ (learned) and max is the element-wise maximum operator. We typically choose $p > k$, (*e.g.* $p = 8192$). Note that because this formulation is framed in term of sets, it can process videos of different lengths (*i.e.*, a different value of m). A second block of layers reduces the dimensionality of each embedding layer (audio and video), and merges the result into a single vector by using a simple concatenation operation. We chose to reduce the dimensionality of each embedding layer separately *before* the concatenation operation to avoid the concatenation of two high dimensional vectors.

Finally, the classification block uses a combination of Mixtures-of-Experts (MoE) and Context Gating to calculate the final probabilities. The Mixtures-of-Experts layer introduced by Jordan & Jacobs (1993) and proposed for video classification by Abu-El-Haija et al. (2016) is used to predict each label independently. It consists of a gating and experts networks which are concurrently learned. The gating network learns which experts to use for each label and the experts layers learn how to classify each label. The context gating operation was introduced by Miech et al. (2017) and captures dependencies among features and re-weight probabilities based on the correlation of the labels. For example, it can capture the correlation of the labels *ski* and *snow* and re-adjust the probabilities accordingly.

B.2.2 Robust Deep Bag-of-Frames pooling method

We propose a technique to extract more performance from the base model with DBoF embedding. The maximum pooling is sensitive to outliers and noise whereas the average pooling is more robust. We propose a method which consists in taking several samples of frames, applying the upsampling followed by the maximum pooling to these samples, and then averaging over all samples. More formally, assume a video contains m frames $\mathbf{v}_1, \ldots, \mathbf{v}_m \in \mathbb{R}^{1024}$. We first draw n random samples $\mathbf{S}^{(1)} \ldots \mathbf{S}^{(n)}$ of size k from the set $\Big\{\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(m)}\Big\}$. The output of the robust-DBoF layer is:

$$\frac{1}{n}\sum_{i=1}^{n} \max\{\mathbf{W}\mathbf{v} : \mathbf{v} \in S_i\} \tag{B.2}$$

Depending on n and k, this pooling method is a trade-off between the max pooling and the average pooling. Thus, it is more robust to noise, as will be shown in the experiments section.

B.2.3 Compact Representation of the Base Model

In order to build compact layers, we use the diagonal-circulant matrix decomposition. The layers are represented as follows:

$$\phi(\mathbf{x}) = \rho\left(\left[\prod_{i=1}^{m} \mathbf{D}^{(i)}\mathbf{C}^{(i)}\right]\mathbf{x} + \mathbf{b}\right) \tag{B.3}$$

where the parameters of each matrix $\mathbf{D}^{(i)}$ and $\mathbf{C}^{(i)}$ are trained using a gradient based optimization algorithm, and m defines the number of factors. Increasing the value of m increases the number of trainable parameters and therefore the modeling capabilities of the layer. In our experiments, we chose the number of factors m empirically to achieve the best trade-off between model size and accuracy.

To measure the impact of the size of the model and its accuracy, we represent layers in their compact form independently. Given that circulant and diagonal matrices are square, we use concatenation and slicing to achieve the desired dimension. As such, with $m = 1$, the weight matrix of size 1024×8192 of the video embedding is represented by a concatenation of 8 DC matrices and the weight matrix of size 8192×512 is represented by a single DC matrix with size 8192×8192 and the resulting output is sliced at the 512 dimension. We denote layers in their classic form as *"Dense"* and layers represented with circulant and diagonal factors as *"Compact"*.

B.2.4 Leveraging Architectural Diversity

In order to benefit from architectural diversity, we also devise a single model architecture that combines different types of embedding layers. As we can see in Figure B.2, video and audio frames are processed by several embedding layers before being reduced by a series of compact fully connected layers. The output of the compact fully connected layers are then averaged, concatenated and fed into the final classification block. Figure B.9 shows the result of different models given the number of parameters. The use of circulant matrices allow us to fit this model in GPU memory. For example, the diversity model with a NetVLAD embedding (cluster size of 256) and NetFV embedding (cluster size of 128) has 160 millions parameters (600 Mo) in the compact version and 728M (2.7 Go) in the dense version.

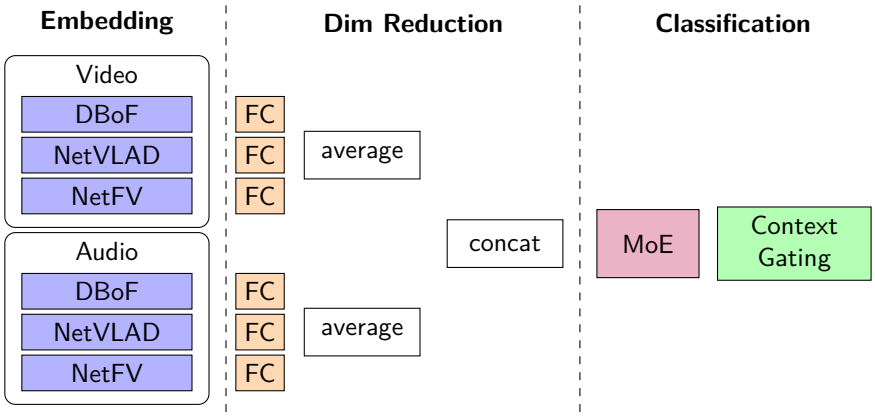


Figure B.2: Diagram of architecture with several embeddings devised to leverage the diversity of an Ensemble in a single model.

B.3 Experiments

In this section, we first evaluate the pooling technique proposed in Section B.2.2. Then, we conduct experiments to evaluate the accuracy of our compact models. In particular, we investigate which layer benefits the most from a circulant representation and show that our approach where both the diagonal and the circulant is learned performs better than the approach from Cheng et al. (2015) for the video classification problem. Finally, we compare all our models on a two dimensional size *vs.* accuracy scale in order to evaluate the trade-off between size and accuracy of each one of our models. All the figures of this section can be found at the end of the chapter.

Experimental Setup All the experiments of this appendix have been done in the context of the *2nd YouTube-8M Video Understanding Challenge* with the *YouTube-8M* dataset. We trained our models with the full training set and 70% of the validation set which corresponds to a total of 4 822 555 examples. We used the data augmentation technique proposed by Skalic et al. (2017) to virtually double the number of inputs. The method consists in splitting the videos into two equal parts. This approach is motivated by the observation that a human could easily label the video by watching either the beginning or the ending of the video. All our experiments are developed with TensorFlow Framework (Abadi et al. 2015). We trained our models with the CrossEntropy loss and used Adam optimizer with a 0.0002 learning rate and a 0.8 exponential decay every 4 million examples. All fully connected layers are composed of 512 units. DBoF, NetVLAD and NetFV are respectively 8192, 64 and 64 of cluster size for video frames and 4096, 32, 32 for audio frames. We used 4 mixtures

for the MoE Layer. We used all the available 150 frames and robust max pooling introduced in Section B.2.2 for the DBoF embedding. In order to stabilize and accelerate the training, we used batch normalization before each nonlinear activation and gradient clipping. We used the GAP (Global Average Precision), as used in the *2nd YouTube-8M Video Understanding Challenge*, to compare our experiments. The GAP metric is defined as follows:

$$\text{GAP} = \sum_{i=1}^{P} p(i)\Delta r(i) \tag{B.4}$$

where P is the number of final predictions, $p(i)$ the precision, and $r(i)$ the recall. We limit our evaluation to 20 predictions for each video. All experiments have been realized on a cluster of 12 nodes. Each node has 160 POWER8 processor, 128 Go of RAM and 4 Nividia Titan P100.

Robust Deep Bag-of-Frames pooling method We evaluate the performance of our Robust DBoF embedding. In accordance with the work of Abu-El-Haija et al. (2016), we find that average pooling performs better than maximum pooling. Figure B.3 shows that the proposed robust maximum pooling method outperforms both maximum and average pooling.

Model	**#Parameters**	**CR**	**GAP@20**	**Diff.**
Dense Model	45M	–	**0.846**	–
Compact DBoF	36M	18.4	0.838	-0.008
Compact FC	41M	9.2	0.845	**-0.001**
Compact MoE	12M	72.0	0.805	-0.041

Table B.1: Effect of the compactness of different layers.

Impact of Circulant Matrices on Different Layers This series of experiments aims at understanding the effect of compactness over different layers. Table B.1 shows the result in terms of the number of weights, compression ratio (CR) with respect to the dense model and GAP score. In these experiments, for speeding-up the training phase, we did not use the audio features and exploited only the video information. The compact fully connected layer achieves a compression rate of 9.5 while having a very similar performance, whereas the compact DBoF and MoE achieve a higher compression rate at the expense of accuracy. Figure B.4 shows that the model with a compact FC converges faster than the dense model. The model with a compact

DBoF shows a big variance over the validation GAP which can be associated with a difficulty to train. The model with a compact MoE is more stable but at the expense of its performance.

Another series of experiments investigates the effect of adding factors of diagonal-circulant layers. Table B.2 shows that there is no gain in accuracy even if the number of weights increases. It also shows that adding factors has an important effect on the speed of training. On the basis of this result, *i.e.* given the performance and compression ratio, we can consider that representing the fully connected layer of the base model in a compact fashion can be a good trade-off.

#Factors	**FC Layer**		**GAP@20**
	#Parameters	**CR**	
–	6.29M	–	0.861
1	12K	99.8	0.861
3	73K	98.8	0.861
6	147K	97.6	0.859

Table B.2: Evolution of the number of parameters and accuracy according to the number of factors.

Comparison with Related Works Circulant matrices have already been used in neural networks by Cheng et al. (2015). They proposed to replace fully connected layers by a circulant and diagonal matrices where the circulant matrix is learned by a gradient based optimization algorithm and the diagonal matrix is random with values in {-1, 1}. We compare our more general framework with their approach. Figure B.5 shows the validation GAP according to the number of epochs of the base model with a compact fully connected layer implemented with both approaches.

Compact Baseline Model with Different Embeddings To compare the performance and the compression ratio we can expect, we consider different settings where the compact fully connected layer is used together with different embeddings. Figures B.6, B.7, B.8 and Table B.3 show the performance of the base model with DBoF, NetVLAD and NetFV embeddings with a *Dense* and *Compact* layer. Notice that we can get a bigger compression rate with NetVLAD and NetFV due to the fact that the output of the embedding is in a higher dimensional space which implies a larger weight matrix for the fully connected layer. Although the compression rate is higher, it is at the expense of accuracy.

Embedding	Method	#Parameters	CR	GAP@20
DBoF	FC Dense	65M	–	0.861
	FC Circulant	59M	9.56	0.861
NetVLAD	FC Dense	86M	–	0.864
	FC Circulant	50M	41.1	0.851
NetFisher	FC Dense	122M	–	0.860
	FC Circulant	51M	58.1	0.848

Table B.3: Impact of the compression of the fully connected layer of the model architecture with Audio and Video features vector and different types of embeddings.

Tradeoff Between Model Size and Accuracy To conclude our experimental evaluation, we compare all our models in terms of size and accuracy. The results are presented in Figure B.9. As we can see in this figure, the most compact models are obtained with the circulant NetVLAD and NetFV. We can also see that the complex architectures described in Section B.2.4 (DBoF + NetVLAD) achieve top performance but at the cost of a large number of weights. Finally, the best trade-off between size and accuracy is obtained using the DBoF embedding layer and achieves a GAP of 0.861 for only 60 millions weights.

B.4 Concluding Remarks

In this appendix, we demonstrated that circulant matrices and diagonal matrices can be a great tool to design compact neural network architectures for video classification tasks. Our experiments demonstrate that the best trade-off between size and accuracy is obtained using circulant DBoF embedding layers. We investigated a model with multiple embeddings to leverage the performance of an Ensemble but found it ineffective. The good performance of Ensemble models, *i.e.*, why aggregating different distinct models performs better that incorporating all the diversity in a single architecture is still an open problem. Our future work will be devoted to address this challenging question and to pursue our effort to devise compact models achieving the same accuracy as larger one, and to study their theoretical properties.

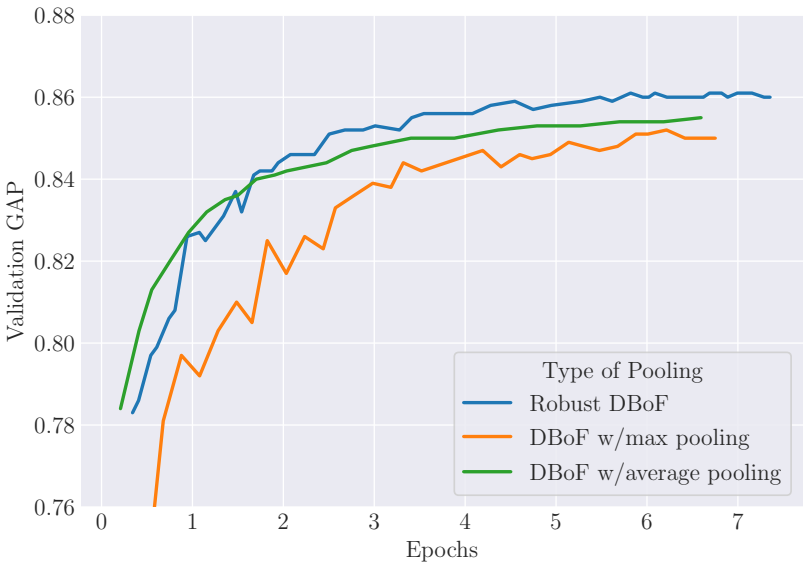


Figure B.3: Impact of *robust DBoF* with $n = 10$ and $k = 15$ on the Deep Bag-of-Frames embedding compared to max and average pooling.

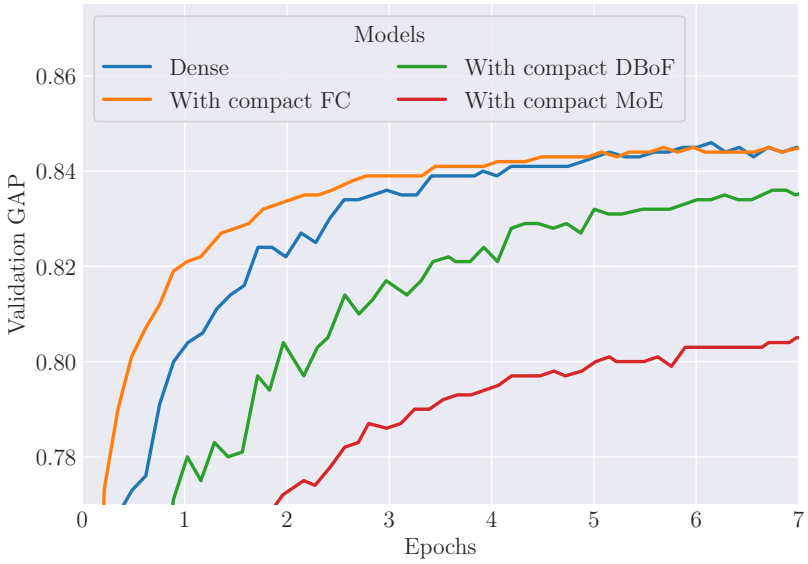


Figure B.4: GAP score of models according to the number of epochs for different compact models.

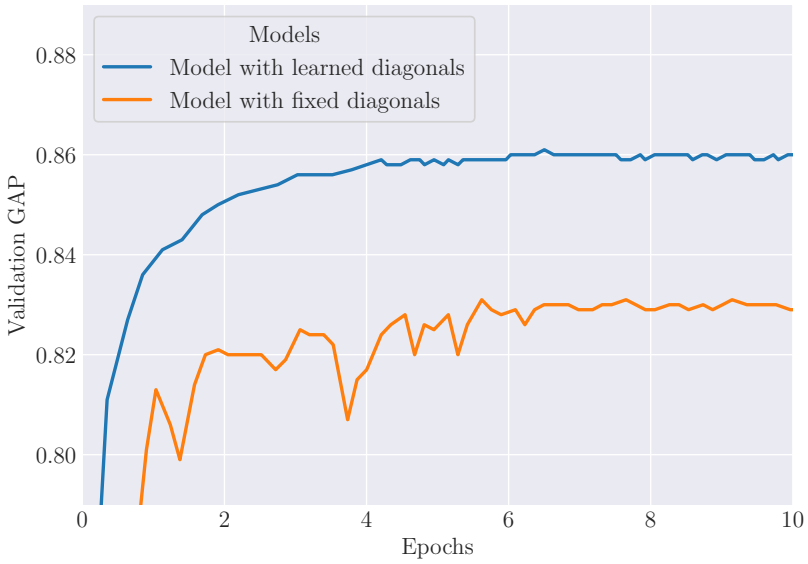


Figure B.5: GAP difference between the approach proposed by Cheng et al. (2015) where the diagonals from the decomposition are initialized from the set $\{-1, +1\}$ and kept fixed and our approach where the values of the diagonals are learned.

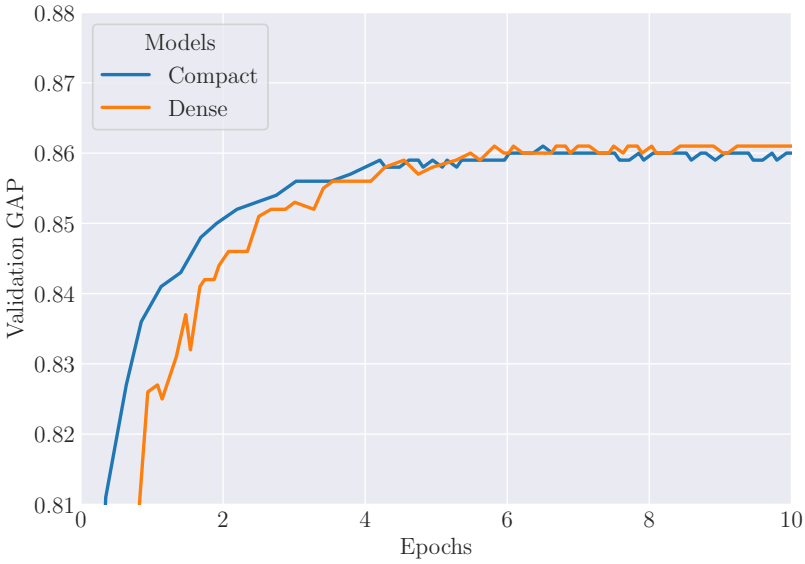


Figure B.6: GAP score of models with compact DBoF embedding and dense fully connected layer.

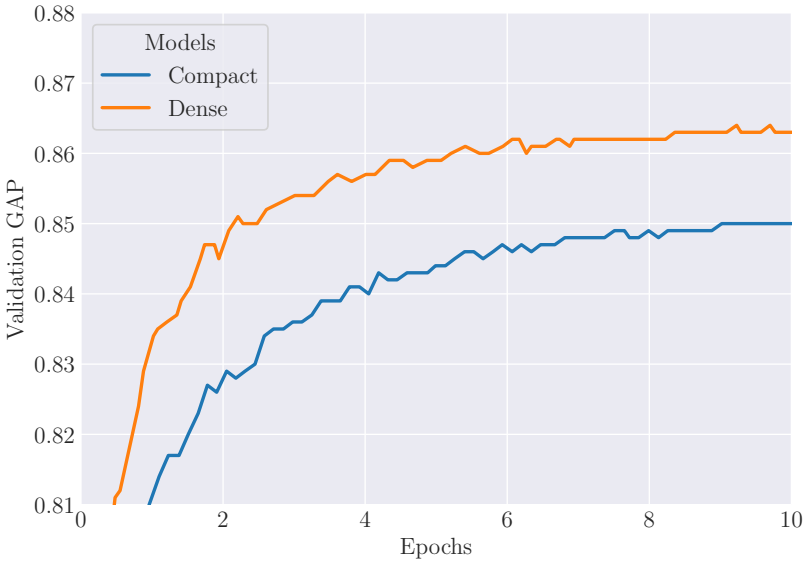


Figure B.7: GAP score of models with compact NetVLAD embedding and dense fully connected layer.

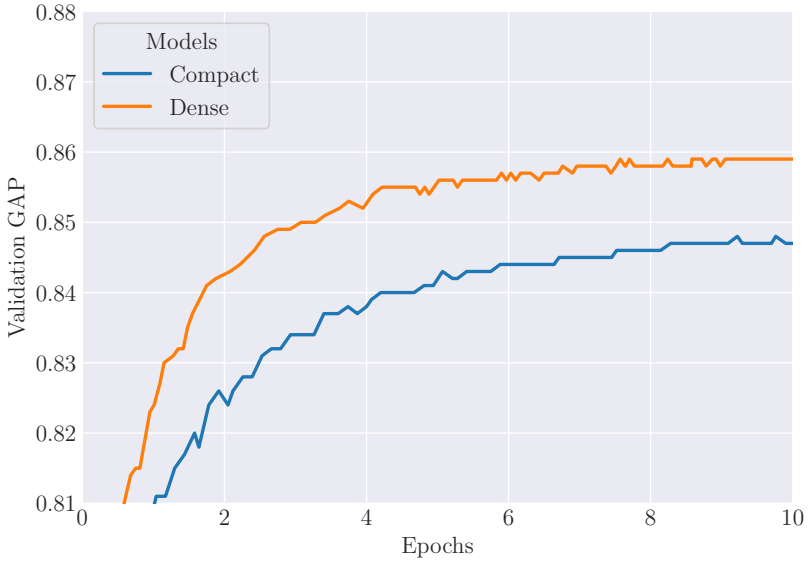


Figure B.8: GAP score of models with compact NetFV embedding and dense fully connected layer.

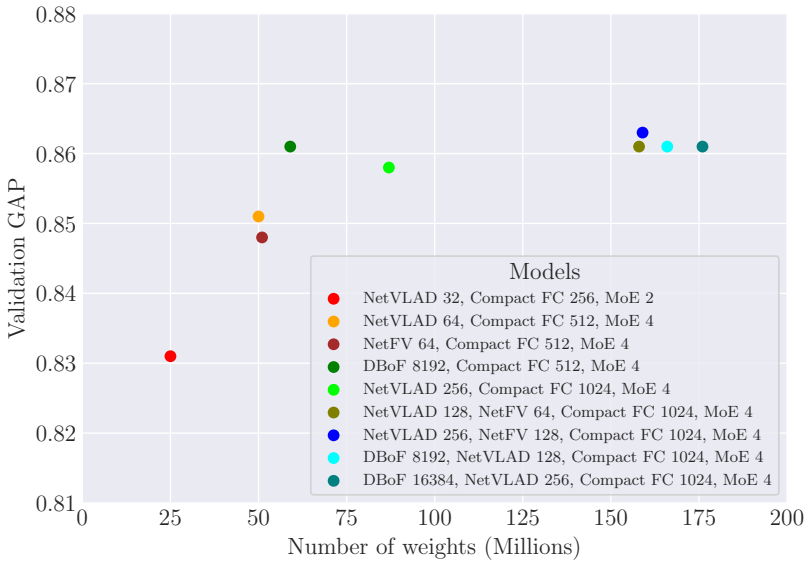


Figure B.9: Comparison between different models with compact fully connected layers.

Appendix C

Theoretical Evidence for Adversarial Robustness through Randomization

Contents

This Appendix concerns a collaboration with Rafael Pinot, Laurent Meunier, Hisashi Kashima, Florian Yger, Cédric Gouy-Pailler and Jamal Atif. This work has been published at the Conference on Neural Information Processing Systems (NeurIPS) 2019. It investigates the theory of robustness against adversarial attacks. It focuses on the family of randomization techniques that consist in injecting noise in the network at inference time. All proofs of this appendix can be found in the long version of the paper.[1] For simplification, we left the notation as in the original paper.

[1] https://arxiv.org/abs/1902.01148

C.1 Introduction

Adversarial attacks are some of the most puzzling and burning issues in modern machine learning. An adversarial attack refers to a small, imperceptible change of an input maliciously designed to fool the result of a machine learning algorithm. Since the seminal work of Szegedy et al. (2014) exhibiting this intriguing phenomenon in the context of deep learning, a wealth of results have been published on designing attacks (Goodfellow et al. 2015; Kurakin et al. 2016; Moosavi-Dezfooli et al. 2016; Papernot et al. 2016a; Carlini & Wagner, 2017; Moosavi-Dezfooli et al. 2017) and defenses (Goodfellow et al. 2015; Papernot et al. 2016b; Meng & Chen, 2017; Guo et al. 2018; Madry et al. 2018; Samangouei et al. 2018)), or on trying to understand the very nature of this phenomenon (Fawzi et al. 2016, 2018a,b; Simon-Gabriel et al. 2018). Most methods remain unsuccessful to defend against powerful adversaries (Carlini & Wagner, 2017; Athalye et al. 2018; Madry et al. 2018). Among the defense strategies, randomization has proven effective in some contexts. It consists in injecting random noise (both during training and inference phases) inside the network architecture, *i.e.*, at a given layer of the network. Noise can be drawn either from Gaussian (Lecuyer et al. 2018; Liu et al. 2018; Rakin et al. 2018), Laplace (Lecuyer et al. 2018), Uniform (Xie et al. 2018), or Multinomial (Dhillon et al. 2018) distributions. Remarkably, most of the considered distributions belong to the Exponential family. Albeit these significant efforts, several theoretical questions remain unanswered. Among these, we tackle the following, for which we provide principled and theoretically-founded answers:

Q1: To what extent does a noise drawn from the Exponential family preserve robustness (in a sense to be defined) to adversarial attacks?

A1: We introduce a definition of robustness to adversarial attacks that is suitable to the randomization defense mechanism. As this mechanism can be described as a non-deterministic querying process, called probabilistic mapping in the sequel, we propose a formal definition of robustness relying on a metric/divergence between probability measures. A key question arises then about the appropriate metric/divergence for our context. This requires tools for comparing divergences w.r.t. the introduced robustness definition. Renyi divergence turned out to be a measure of choice, since it satisfies most of the desired properties (coherence, strength, and computational tractability). Finally, thanks to the existing links between the Renyi divergence and the Exponential family, we were able to prove that methods based on noise

injection from the Exponential family ensures robustness to adversarial examples (cf. Theorem C.1).

Q2: Can we guarantee a good accuracy under attack for classifiers defended with this kind of noise?

A2: We present an upper bound on the drop of accuracy (under attack) of the methods defended with noise drawn from the Exponential family (*cf.* Theorem C.2). Then, we illustrate this result by training different randomized models with Laplace and Gaussian distributions on CIFAR-10 dataset. These experiments highlight the trade-off between accuracy and robustness that depends on the amount of noise one injects in the network. Our theoretical and experimental conclusion is that randomized defenses are competitive (with the current state-of-the-art (Madry et al. 2018)) given the intensity of noise injected in the network.

Outline of the chapter: We present in Section C.2 the related work on randomized defenses to adversarial examples. Section C.3 introduces the definition of robustness relying on a metric/divergence between probability measures, and discusses the key role of the Renyi divergence. We state in Section C.4 our main results on the robustness and accuracy of Exponential family-based defenses. Section C.5 presents extensive experiments supporting our theoretical findings. Section C.6 provides concluding remarks.

C.2 Related works

Injecting noise into algorithms to improve their robustness has been used for ages in detection and signal processing tasks (Mitaim & Kosko, 1998; Zozor & Amblard, 1999; Chapeau-Blondeau & Rousseau, 2004). It has also been extensively studied in several machine learning and optimization fields, *e.g.* robust optimization (Ben-Tal et al. 2009) and data augmentation techniques (Perez & Wang, 2017). Recently, noise injection techniques have been adopted by the adversarial defense community, especially for neural networks, with very promising results. Randomization techniques are generally oriented towards one of the following objectives: experimental robustness or provable robustness.

Experimental robustness: The first technique explicitly using randomization at inference time as a defense appeared during the 2017 NeurIPS defense challenge (Xie

et al. 2018). This method uniformly samples over geometric transformations of the image to select a substitute image to feed the network. Then Dhillon et al. (2018) proposed to use stochastic activation pruning based on a multinomial distribution for adversarial defense. Several works (Liu et al. 2018; Rakin et al. 2018) propose to inject Gaussian noise directly on the activation of selected layers both at training and inference time. While these works hypothesize that noise injection makes the network robust to adversarial perturbations, they do not provide any formal justification on the nature of the noise they use or on the loss of accuracy/robustness of the network.

Provable robustness: Lecuyer et al. (2018) proposed a randomization method by exploiting the link between differential privacy (Dwork, Roth et al. 2014) and adversarial robustness. Their framework, called "randomized smoothing" [2], inherits some theoretical results from the differential privacy community allowing them to evaluate the level of accuracy under attack of their method. Initial results by Lecuyer et al. (2018) have been refined by Li et al. (2018), and by Cohen et al. (2019). Our work belongs to this line of research. However, our framework does not treat exactly the same class of defenses. Notably, we provide theoretical arguments supporting the defense strategy based on randomization techniques relying on the exponential family, and derive a new bound on the adversarial generalization gap, which completes the results obtained so far on certified robustness. Furthermore, our focus is on the network randomized by noise injection, "randomized smoothing" instead uses this network to create a *new* classifier robust to attacks.

Since the initial discovery of adversarial examples, a wealth of non randomized defense approaches have also been proposed, inspired by various machine learning domains such as adversarial training (Goodfellow et al. 2015; Madry et al. 2018), image reconstruction (Meng & Chen, 2017; Samangouei et al. 2018) or robust learning (Goodfellow et al. 2015; Madry et al. 2018). Even if these methods have their own merits, a thorough evaluation made by Athalye et al. (2018) shows that most defenses can be easily broken with known powerful attacks (Carlini & Wagner, 2017; Chen et al. 2018; Madry et al. 2018). Adversarial training, which consists in training a model directly on adversarial examples, came out as the best defense in average. Defense based on randomization could be overcome by the Expectation Over Transformation technique proposed by Athalye et al. (2017) which consists in taking the expectation over the network to craft the perturbation. In this chapter, to ensure that our results are not biased by obfuscated gradients, we follow the principles provided by (Athalye et al.

[2]Name introduced by Cohen et al. (2019) after the work of (Lecuyer et al. 2018).

2018; Carlini et al. 2019) and evaluate our randomized networks with this technique. We show that randomized defenses are still competitive given the intensity of noise injected in the network.

C.3 General definitions of risk and robustness

C.3.1 Risk, robustness and probabilistic mappings

Let us consider two spaces $\mathcal{X}$ (with norm $\|\cdot\|_{\mathcal{X}}$), and $\mathcal{Y}$. We consider the classification task that seeks a hypothesis (classifier) $h : \mathcal{X} \to \mathcal{Y}$ minimizing the risk of h w.r.t. some ground-truth distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. The risk of h w.r.t. $\mathcal{D}$ is defined as

$$\text{Risk}(h) \triangleq \mathbb{E}_{(x,y)\sim\mathcal{D}}[\mathbb{1}(h(x) \neq y)].$$

Given a classifier $h : \mathcal{X} \to \mathcal{Y}$, and some input $x \in \mathcal{X}$ with true label $y_{true} \in \mathcal{Y}$, to generate an adversarial example, the adversary seeks a τ such that $h(x+\tau) \neq y_{true}$, with some budget α over the perturbation (*i.e.*, with $\|\tau\|_{\mathcal{X}} \leq \alpha$). α represents the maximum amount of perturbation one can add to x without being spotted (the perturbation remains humanly imperceptible). The overall goal of the adversary is to find a perturbation crafting strategy that both maximizes the risk of h, and keeps the values of $\|\tau\|_{\mathcal{X}}$ small. To measure this risk "under attack" we define the notion of adversarial α-radius risk of h w.r.t. $\mathcal{D}$ as follows

$$\text{Risk}_\alpha(h) \triangleq \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\sup_{\|\tau\|_{\mathcal{X}}\leq\alpha} \mathbb{1}(h(x+\tau) \neq y)\right] . \tag{C.1}$$

In practice, the adversary does not have any access to the ground-truth distribution. The literature proposed several surrogate versions of $\text{Risk}_\alpha(h)$ (see Diochnos et al. (2018) for more details) to overcome this issue. We focus our analysis on the one used by Szegedy et al. (2014) and Fawzi et al. (2018) denoted α-radius prediction-change risk of h w.r.t. $\mathcal{D}_{\mathcal{X}}$ (marginal of $\mathcal{D}$ for $\mathcal{X}$), and defined as

$$\text{PC-Risk}_\alpha(h) \triangleq \mathbb{P}_{x\sim\mathcal{D}_{\mathcal{X}}}[\exists\tau \in \text{B}(\alpha) \text{ s.t. } h(x+\tau) \neq h(x)] \tag{C.2}$$

where for any $\alpha \geq 0$, $\quad \text{B}(\alpha) \triangleq \{\tau \in \mathcal{X} \text{ s.t. } \|\tau\|_{\mathcal{X}} \leq \alpha\}$.

As we will inject some noise in our classifier in order to defend against adversarial attacks, we need to introduce the notion of "probabilistic mapping". Let $\mathcal{Y}$ be

the output space, and $\mathcal{F}_\mathcal{Y}$ a *σ-algebra* over $\mathcal{Y}$. Let us also denote $\mathcal{P}(\mathcal{Y})$ the set of probability measures over $(\mathcal{Y}, \mathcal{F}_\mathcal{Y})$.

Definition C.1 (Probabilistic mapping)**.** *Let $\mathcal{X}$ be an arbitrary space, and $(\mathcal{Y}, \mathcal{F}_\mathcal{Y})$ a measurable space. A* probabilistic mapping *from $\mathcal{X}$ to $\mathcal{Y}$ is a mapping $\mathrm{M} : \mathcal{X} \to \mathcal{P}(\mathcal{Y})$. To obtain a numerical output out of this* probabilistic mapping*, one needs to sample y according to $\mathrm{M}(x)$.*

This definition does not depend on the nature of $\mathcal{Y}$ as long as $(\mathcal{Y}, \mathcal{F}_\mathcal{Y})$ is measurable. In that sense, $\mathcal{Y}$ could be either the label space or any intermediate space corresponding to the output of an arbitrary hidden layer of a neural network. Moreover, any mapping can be considered as a probabilistic mapping, whether it explicitly injects noise (see Dhillon et al. (2018), Lecuyer et al. (2018) and Rakin et al. (2018)) or not. In fact, any deterministic mapping can be considered as a probabilistic mapping, since it can be characterized by a Dirac measure. Accordingly, the definition of a probabilistic mapping is fully general and equally treats networks with or without noise injection. There exists no definition of robustness against adversarial attacks that comply with the notion of probabilistic mappings. We settle that by generalizing the notion of prediction-change risk initially introduced by Diochnos et al. (2018) for deterministic classifiers. Let M be a probabilistic mapping from $\mathcal{X}$ to $\mathcal{Y}$, and $d_{\mathcal{P}(\mathcal{Y})}$ some metric/divergence on $\mathcal{P}(\mathcal{Y})$. We define the (α, ϵ)-radius prediction-change risk of M w.r.t. $\mathcal{D}_\mathcal{X}$ and $d_{\mathcal{P}(\mathcal{Y})}$ as

$$\text{PC-Risk}_\alpha(\mathrm{M}, \epsilon) \triangleq \mathbb{P}_{x \sim \mathcal{D}_\mathcal{X}} \Big[\exists \tau \in B(\alpha) \text{ s.t. } d_{\mathcal{P}(\mathcal{Y})}(\mathrm{M}(x+\tau), \mathrm{M}(x)) > \epsilon \Big] \ . \tag{C.3}$$

These three generalized notions allow us to analyze noise injection defense mechanisms (Theorems C.1, and C.2). We can also define adversarial robustness (and later adversarial gap) thanks to these notions.

Definition C.2 (Adversarial robustness)**.** *Let $d_{\mathcal{P}(\mathcal{Y})}$ be a metric/divergence on $\mathcal{P}(\mathcal{Y})$. The probabilistic mapping* M *is said to be $d_{\mathcal{P}(\mathcal{Y})}$-$(\alpha, \epsilon, \gamma)$ robust if*

$$\text{PC-Risk}_\alpha(\mathrm{M}, \epsilon) \leq \gamma \ . \tag{C.4}$$

It is difficult in general to show that a classifier is $d_{\mathcal{P}(\mathcal{Y})}$-$(\alpha, \epsilon, \gamma)$ robust. However, we can derive some bounds for particular divergences that will ensure robustness up to a certain level (Theorem C.1). It is worth noting that our definition of robustness depends on the considered metric/divergence between probability measures.

Lemma C.1 gives some insights on the monotony of the robustness according to the parameters, and the probability metric/divergence at hand.

Lemma C.1. *Let* M *be a probabilistic mapping, and let* d_1 *and* d_2 *be two metrics on* $\mathcal{P}(\mathcal{Y})$*. If there exists a non decreasing function* $\phi : \mathbb{R} \to \mathbb{R}$ *such that* $\forall \mu_1, \mu_2 \in \mathcal{P}(\mathcal{Y})$*,* $d_1(\mu_1, \mu_2) \leq \phi(d_2(\mu_1, \mu_2))$*, then the following assertion holds:*

$$\text{M } \textit{ is } d_2\text{-}(\alpha, \epsilon, \gamma)\textit{-robust} \implies \text{M } \textit{ is } d_1\text{-}(\alpha, \phi(\epsilon), \gamma)\textit{-robust} \tag{C.5}$$

As suggested in Definition C.2 and Lemma C.1, any given choice of metric/divergence will instantiate a particular notion of adversarial robustness and it should be carefully selected.

C.3.2 On the choice of the metric/divergence for robustness

The aforementioned formulation naturally raises the question of the choice of the metric used to defend against adversarial attacks. The main notions that govern the selection of an appropriate metric/divergence are *coherence*, *strength*, and *computational tractability*. A metric/divergence is said to be coherent if it naturally fits the task at hand (*e.g.* classification tasks are intrinsically linked to discrete/trivial metrics, conversely to regression tasks). The strength of a metric/divergence refers to its ability to cover (dominate) a wide class of others in the sense of Lemma C.1. In the following, we will focus on both the total variation metric and the Renyi divergence, that we consider as respectively the most coherent with the classification task using probabilistic mappings, and the strongest divergence. We first discuss how total variation metric is *coherent* with randomized classifiers but suffers from computational issues. Hopefully, the Renyi divergence provides good guarantees about adversarial robustness, enjoys nice *computational properties*, in particular when considering Exponential family distributions, and is *strong* enough to dominate a wide range of metrics/divergences including total variation.

Let μ_1 and μ_2 be two measures in $\mathcal{P}(\mathcal{Y})$, both dominated by a third measure ν. The trivial distance $d_T(\mu_1, \mu_{\triangleq} \mathbb{1}(\mu_1 \neq \mu_2)$ is the simplest distance one can define between μ_1 and μ_2. In the deterministic case, it is straightforward to compute (since the numerical output of the algorithm characterizes its associated measure), but this is not the case in general. In fact one might not have access to the true distribution of the mapping, but just to the numerical outputs. Therefore, one needs to consider more sophisticated metrics/divergences, such as the total variation distance

$d_{TV}(\mu_1, \mu_2) \triangleq \sup_{Y \in \mathcal{F}_\mathcal{Y}} |\mu_1(Y) - \mu_2(Y)|$. The total variation distance is one of the most broadly used probability metrics. It admits several very simple interpretations, and is a very useful tool in many mathematical fields such as probability theory, Bayesian statistics, coupling or transportation theory. In transportation theory, it can be rewritten as the solution of the Monge-Kantorovich problem with the cost function $c(y_1, y_2) = \mathbb{1}(y_1 \neq y_2)$: $\inf \int_{\mathcal{Y}^2} \mathbb{1}(y_1 \neq y_2) \, \mathrm{d}\pi(y_1, y_2)$, where the infimum is taken over all joint probability measures π on $(\mathcal{Y} \times \mathcal{Y}, \mathcal{F}_\mathcal{Y} \otimes \mathcal{F}_\mathcal{Y})$ with marginals μ_1 and μ_2. According to this interpretation, it seems quite natural to consider the total variation distance as a relaxation of the trivial distance on $[0, 1]$ (see the book of Villani (2008) for details). In the deterministic case, the total variation and the trivial distance coincides. In general, the total variation allows a finer analysis of the probabilistic mappings than the trivial distance. But it suffers from a high computational complexity. In the following of the chapter we will show how to ensure robustness regarding TV distance.

Finally, denoting by g_1 and g_2 the respective probability distributions w.r.t. ν, the Renyi divergence of order λ (Rényi, 1961) writes as

$$d_{R,\lambda}(\mu_1, \mu_2) \triangleq \frac{1}{\lambda - 1} \log \int_\mathcal{Y} g_2(y) \left(\frac{g_1(y)}{g_2(y)} \right)^\lambda \mathrm{d}\nu(y). \tag{C.6}$$

The Renyi divergence is a generalized measure defined on the interval $(1, \infty)$, where it equals the Kullback-Leibler divergence when $\lambda \to 1$ (that will be denoted d_{KL}), and the maximum divergence when $\lambda \to \infty$. It also has the very special property of being non decreasing w.r.t. λ. This divergence is very common in machine learning, especially in its Kullback-Leibler form as it is widely used as the loss function (cross entropy) of classification algorithms. It enjoys the desired properties since it bounds the TV distance, and is tractable. Furthermore, Proposition C.1 proves that Renyi-robustness implies TV-robustness, making it a suitable surrogate for the trivial distance.

Proposition C.1 (Renyi-robustness implies TV-robustness)**.** *Let* M *be a probabilistic mapping, then* $\forall \lambda \geq 1$*:*

$$\mathrm{M} \text{ is } d_{R,\lambda}\text{-}(\alpha, \epsilon, \gamma)\text{-robust} \implies \mathrm{M} \text{ is } d_{TV}\text{-}(\alpha, \epsilon', \gamma)\text{-robust} \tag{C.7}$$

$$\text{with } \epsilon' = \min\left(\frac{3}{2}\left(\sqrt{1 + \frac{4\epsilon}{9}} - 1 \right)^{1/2}, \frac{\exp(\epsilon + 1) - 1}{\exp(\epsilon + 1) + 1} \right) . \tag{C.8}$$

A crucial property of Renyi-robustness is the *Data processing inequality.* It is a well-known inequality from information theory which states that *"post-processing cannot increase information"* (Beaudry & Renner, 2012; Cover & Thomas, 2012). In our case, if we consider a Renyi-robust probabilistic mapping, composing it with a deterministic mapping maintains Renyi-robustness with the same level.

Proposition C.2 (Data processing inequality)**.** *Let us consider a probabilistic mapping* $\mathrm{M} : \mathcal{X} \to \mathcal{P}(\mathcal{Y})$*. Let us also denote* $\rho : \mathcal{Y} \to \mathcal{Y}'$ *a deterministic function. If* $U \sim \mathrm{M}(x)$ *then the probability measure* $M'(x)$ *s.t* $\rho(U) \sim M'(x)$ *defines a probabilistic mapping* $M' : \mathcal{X} \to \mathcal{P}(\mathcal{Y}')$*. For any* $\lambda > 1$ *if* M *is* $d_{R,\lambda}$*-*$(\alpha, \epsilon, \gamma)$ *robust then* M' *is also is* $d_{R,\lambda}$*-*$(\alpha, \epsilon, \gamma)$ *robust.*

Data processing inequality will allow us later to inject some additive noise in any layer of a neural network and to ensure Renyi-robustness.

C.4 Defense mechanisms based on Exponential family noise injection

C.4.1 Robustness through Exponential family noise injection

For now, the question of which class of noise to add is treated *ad hoc.* We choose here to investigate one particular class of noise closely linked to the Renyi divergence, namely Exponential family distributions, and demonstrate their interest. Let us first recall what the Exponential family is.

Definition C.3 (Exponential family)**.** *Let* Θ *be an open convex set of* $\mathbb{R}^n$*, and* $\theta \in \Theta$*. Let* ν *be a measure dominated by* μ *(either by the Lebesgue or counting measure), it is said to be part of the* Exponential family *of parameter* θ *(denoted* $E_F(\theta, t, k)$*) if it has the following probability density function*

$$p_F(z, \theta) = \exp\{\langle t(z), \theta\rangle - u(\theta) + k(z)\} \tag{C.9}$$

where $t(z)$ *is a sufficient statistic,* k *a carrier measure (either for a Lebesgue or a counting measure) and* $u(\theta) = \log \int_z \exp\{< t(z), \theta > + k(z)\}\, \mathrm{d}z$*.*

To show the robustness of randomized networks with noise injected from the Exponential family, one needs to define the notion of sensitivity for a given deterministic function:

Definition C.4 (Sensitivity of a function). *For any $\alpha \geq 0$ and for any $\|\cdot\|_A$ and $\|\cdot\|_B$ two norms, the α-sensitivity of f w.r.t. $\|\cdot\|_A$ and $\|\cdot\|_B$ is defined as*

$$\Delta_\alpha^{A,B}(f) \triangleq \sup_{x,y\in\mathcal{X},\|x-y\|_A\leq\alpha} \|f(x)-f(y)\|_B \ . \tag{C.10}$$

Let us consider an n-layer feedforward neural network $\mathcal{N}(\ \cdot\ ) = \phi^n \circ \cdots \circ \phi^1(\ \cdot\ )$. For any $i \in [n]$, we define $\mathcal{N}_{|i}(\ \cdot\ ) = \phi^i \circ \cdots \circ \phi^1(\ \cdot\ )$ the neural network truncated at layer i. Theorem C.1 shows that, injecting noise drawn from an Exponential family distribution ensures robustness to adversarial example attacks in the sense of Definition C.2.

Theorem C.1 (Exponential family ensures robustness). *Let us denote $\mathcal{N}_X^i(\ \cdot\ ) = \phi^n \circ \cdots \circ \phi^{i+1}(\mathcal{N}_{|i}(\ \cdot\ ) + X)$ with X a random variable. Let us also consider two arbitrary norms $\|\cdot\|_A$ and $\|\cdot\|_B$ respectively on $\mathcal{X}$ and on the output space of $\mathcal{N}_X^i$.*

- *If $X \sim E_F(\theta, t, k)$ where t and k have non-decreasing modulus of continuity ω_t and ω_k. Then for any $\alpha \geq 0$, $\mathcal{N}_X^i(\ \cdot\ )$ defines a probabilistic mapping that is $d_{R,\lambda}$-(α, ϵ) robust with $\epsilon = \|\theta\|_2 \omega_t^{B,2}(\Delta_\alpha^{A,B}(\mathcal{N}_{|i})) + \omega_k^{B,1}(\Delta_\alpha^{A,B}(\mathcal{N}_{|i}))$ where $\|\cdot\|_2$ is the norm corresponding to the scalar product in the definition of the exponential family density function and $\|\cdot\|_1$ is the absolute value on $\mathbb{R}$.* [3]

- *If X is a centered Gaussian random variable with a non degenerated matrix parameter Σ. Then for any $\alpha \geq 0$, $\mathcal{N}_X^i(\ \cdot\ )$ defines a probabilistic mapping that is $d_{R,\lambda}$-(α, ϵ) robust with $\epsilon = \frac{\lambda \Delta_\alpha^{A,2}(\phi)^2}{2\sigma_{min}(\Sigma)}$ where $\|\cdot\|_2$ is the canonical Euclidean norm on $\mathbb{R}^n$.*

In simpler words, the previous theorem ensures stability in the neural network when injecting noise w.r.t. the distribution of the output. Intuitively, if two inputs are close w.r.t. $\|\cdot\|_A$, the output distributions of the network will be close in the sense of Renyi divergence. It is well known that in the case of deterministic neural networks, the Lipschitz constant becomes bigger as the number of layers increases (Gouk et al. 2018). By injecting noise at layer i, the notion of robustness only depends on the sensitivity of the first i layers of the network and not the following ones. In that sense, randomization provides a more precise control on the "continuity" of the neural network. In the next section, we show that thanks to the notion of robustness w.r.t. probabilistic mappings, one can bound the loss of accuracy of a randomized neural network when it is attacked.

[3] The notion of continuity modulus is defined in the arxiv version of this chapter: https://arxiv.org/abs/1902.01148.

C.4.2 Bound on the generalization gap under attack

The notions of risk and adversarial risk can easily be generalized to encompass probabilistic mappings.

Definition C.5 (Risks for probabilistic mappings). *Let* M *be a probabilistic mapping from* $\mathcal{X}$ *to* $\mathcal{Y}$*, the risk and the* α*-radius adversarial risk of* M *w.r.t.* $\mathcal{D}$ *are defined as*

$$\mathrm{Risk}(\mathrm{M}) \triangleq \mathbb{E}_{(x,y)\sim\mathcal{D}}\Big[\mathbb{E}_{y'\sim\mathrm{M}(x)}\big[\mathbb{1}(y' \neq y)\big]\Big] \tag{C.11}$$

$$\mathrm{Risk}_\alpha(\mathrm{M}) \triangleq \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\sup_{\|\tau\|_{\mathcal{X}}\leq\alpha} \mathbb{E}_{y'\sim\mathrm{M}(x+\tau)}\big[\mathbb{1}(y' \neq y)\big]\right] . \tag{C.12}$$

The definition of adversarial risk for a probabilistic mapping can be matched with the concept of Expectation over Transformation (EoT) attacks (Athalye et al. 2018). Indeed, EoT attacks aim at computing the best opponent in expectation for a given random transformation. In the adversarial risk definition, the adversary chooses the perturbation which has the greatest probability to fool the model, which is a stronger objective than the EoT objective. Theorem C.2 provides a bound on the gap between the adversarial risk and the regular risk:

Theorem C.2 (Adversarial generalization gap bound in the randomized setting). *Let* M *be the probabilistic mapping at hand. Let us suppose that* M *is* $d_{R,\lambda}$*-*(α, ϵ) *robust for some* $\lambda \geq 1$ *then:*

$$|\,\mathrm{Risk}_\alpha(\mathrm{M}) - \mathrm{Risk}(\mathrm{M})| \leq 1 - e^{-\epsilon}\mathbb{E}_x\Big[e^{-H(\mathrm{M}(x))}\Big] \tag{C.13}$$

where H *is the Shannon entropy* $H(p) = -\sum_i p_i \log(p_i)$.

This theorem gives a control on the loss of accuracy under attack w.r.t. the robustness parameter ϵ and the entropy of the predictor. It provides a trade-off between the quantity of noise added in the network and the accuracy under attack. Intuitively, when the noise increases, for any input, the output distribution tends towards the uniform distribution, then, $\epsilon \to 0$ and $H(\mathrm{M}(x)) \to \log(K)$, and the risk and the adversarial risk both tends to $\frac{1}{K}$ where K is the number of classes in the classification problem. On the opposite, if no noise is injected, for any input, the output distribution is a Dirac distribution, then, if the prediction for the adversarial example is not the same as for the regular one, $\epsilon \to \infty$ and $H(\mathrm{M}(x)) \to 0$. Hence, the noise needs to be designed both to preserve accuracy and robustness to adversarial attacks. In the

Section C.5, we give an illustration of this bound when M is a neural network with noise injection at input level as presented in Theorem C.1.

C.5 Experiments

To illustrate our theoretical findings, we train randomized neural networks with a simple method which consists in injecting a noise drawn from an Exponential family distribution in the image during training and inference. This section aims to answer **Q2** stated in the introduction, by tackling the following sub-questions:

Q2.1: How does the randomization impact the accuracy of the network? And, how does the theoretical trade-off between accuracy and robustness apply in practice?

Q2.2: What is the accuracy under attack of randomized neural networks against powerful iterative attacks? And how does randomized neural networks compare to state-of-the-art defenses given the intensity of the injected noise?

Experimental setup We present our results and analysis on CIFAR-10 (Krizhevsky & Hinton, 2009). We used a Wide ResNet architecture (Zagoruyko & Komodakis, 2016) which is a variant of the ResNet model proposed by He et al. (2016). We use 28 layers with a widen factor of 10. We train all networks for 200 epochs, a batch size of 400, dropout 0.3 and Leaky Relu activation with a slope on $\mathbb{R}^-$ of 0.1. We minimize the Cross Entropy Loss with Momentum 0.9 and use a piecewise constant learning rate of 0.1, 0.02, 0.004 and 0.00008 after respectively 7500, 15000 and 20000 steps. The networks achieve for CIFAR10 and 100 a TOP-1 accuracy of 95.8% and 79.1% respectively on test images.

To transform these classical networks to probabilistic mappings, we inject noise drawn from Laplace and Gaussian distributions, each with various standard deviations. While the noise could theoretically be injected anywhere in the network, we inject the noise on the image for simplicity. More experiments with noise injected in the first layer of the network are presented in the supplementary material. To evaluate our models under attack, we use three powerful iterative attacks with different norms: *ElasticNet* attack (EAD) (Chen et al. 2018) with ℓ_1 distortion, *Carlini&Wagner* attack (C&W) (Carlini & Wagner, 2017) with ℓ_2 distortion and *Projected Gradient Descent* attack (PGD) (Madry et al. 2018) with ℓ_∞ distortion. All standard deviations and attack intensities are in between -1 and 1. Precise descriptions of our numerical

experiments and of the attacks used for evaluation are deferred to the supplementary material.

Attacks against randomized defenses: It has been pointed out by Athalye et al. (2017) and Carlini et al. (2019) that in a white box setting, an attacker with a complete knowledge of the system will know the distribution of the noise injected in the network. As such, to create a stronger adversarial example, the attacker can take the expectation of the loss or the logits of the randomized network during the computation of the attack. This technique is called Expectation Over Transformation (EoT) and we use a Monte Carlo method with 80 simulations to approximate the best perturbation for a randomized network.

Trade-off between accuracy and intensity of noise (Q2.1): When injecting noise as a defense mechanism, regardless of the distribution it is drawn from, we observe (as in Figure C.1) that the accuracy decreases when the noise intensity grows. In that sense, noise needs to be calibrated to preserve both accuracy and robustness against adversarial attacks, *i.e.*, it needs to be large enough to preserve robustness and small enough to preserve accuracy. Figure C.1 shows the loss of accuracy on CIFAR10 from 0.95 to 0.82 (respectively 0.95 to 0.84) with noise drawn from a Gaussian distribution (respectively Laplace) with a standard deviation from 0.01 to 0.5. Figure C.2 and C.3 illustrate the theoretical lower bound on accuracy under attack of Theorem C.2 for different distributions and standard deviations. The term in entropy of Theorem C.2 has been estimated using a Monte Carlo method with 10^4 simulations. The trade-off between accuracy and robustness from Theorem C.2 thus appears w.r.t. the noise intensity. With small noises, the accuracy is high, but the guaranteed accuracy drops fast w.r.t. the magnitude of the adversarial perturbation. Conversely, with bigger noises, the accuracy is lower but decreases slowly w.r.t. the magnitude of the adversarial perturbation. These Figures also show that Theorem C.2 gives strong accuracy guarantees against small adversarial perturbations. Next paragraph shows that in practice, randomized networks achieve much higher accuracy under attack than the theoretical bound, and keep this accuracy against much larger perturbations.

Performance of randomized networks under attacks and comparison to state of the art (Q2.2): While Figure C.2 and C.3 illustrated a theoretical robustness against growing adversarial perturbations, Table C.1 illustrates this trade-

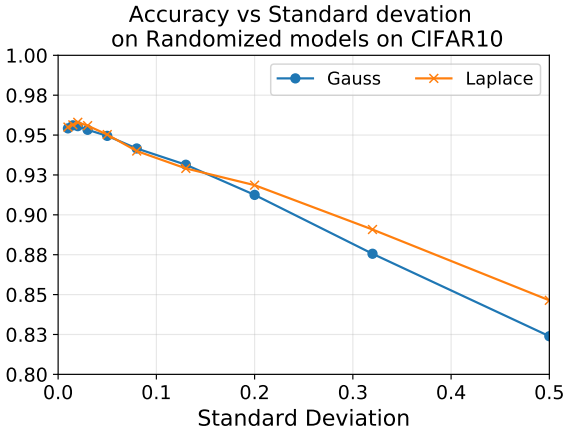


Figure C.1: Impact of the standard deviation of the injected noise on accuracy in a randomized model on CIFAR-10 dataset with a Wide ResNet architecture.

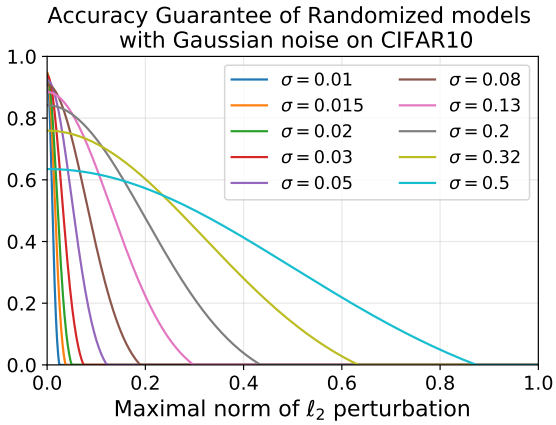


Figure C.2: Illustration of the guaranteed accuracy of different randomized models with Gaussian noises given the norm of the adversarial perturbation.

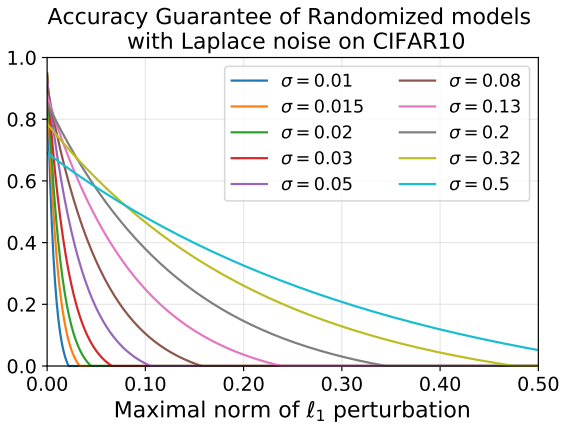


Figure C.3: Illustration of the guaranteed accuracy of different randomized models with Laplace noises given the norm of the adversarial perturbation.

Distribution	Sd	Natural	ℓ_1 EAD 60	ℓ_2 C&W 60	ℓ_∞ PGD 20
–	–	0.958	0.035	0.034	0.384
Normal	0.01	0.954	0.193	0.294	0.408
	0.50	0.824	0.448	0.523	0.587
Laplace	0.01	0.955	0.208	0.313	0.389
	0.50	0.846	0.464	0.494	0.589

Table C.1: Accuracy under attack on the CIFAR-10 dataset with a randomized Wide ResNet architecture. We compare the accuracy on natural images and under attack with different noise over 3 iterative attacks (the number of steps is next to the name) made with 80 Monte Carlo simulations to compute EoT attacks. The first line is the baseline, no noise has been injected.

off experimentally. It compares the accuracy obtained under attack by a deterministic network with the one obtained by randomized networks with Gaussian and Laplace noises both with low (0.01) and high (0.5) standard deviations. Randomized networks with a small noise lead to no loss in accuracy with a small robustness while high noise leads to a higher robustness at the expense of loss of accuracy ($\sim$ 11 points).

Attack	Steps	Madry et al.	Normal		Laplace	
			0.32	**0.5**	**0.32**	**0.5**
–	–	0.873	0.876	0.824	0.891	0.846
ℓ_∞ – PGD	20	0.456	0.566	0.587	0.576	0.589
ℓ_2 – C&W	30	0.468	0.512	0.489	0.502	0.479

Table C.2: Accuracy under attack of randomized neural network with different distributions and standard deviations versus adversarial training by Madry et al. (2018). The PGD attack has been made with 20 step, an epsilon of 0.06 and a step size of 0.006 (input space between -1 and $+1$). The Carlini&Wagner attack uses 30 steps, 9 binary search steps and a 0.01 learning rate. The first line refers to the baseline without attack.

Finally, Table C.2 compares the accuracy and the accuracy under attack of randomized networks with Gaussian and Laplace distributions for different standard deviations against adversarial training from Madry et al. (2018). We observe that the accuracy on natural images of both noise injection methods are similar to the one from Madry et al. (2018). Moreover, both methods are more robust than adversarial training to PGD and C&W attacks. As with all the experiments, to construct an EoT attack, we use 80 Monte Carlo simulations at every step of PGD and C&W attacks. These experiments show that randomized defenses can be competitive given

the intensity of noise injected in the network. Note that these experiments have been led with EoT of size 80. For much bigger sizes of EoT these results would be mitigated. Nevertheless, the accuracy would never drop under the bounds illustrated in the Figures C.2 and C.3, since Theorem C.2 gives a bound that on the worst case attack strategy (including EoT).

C.6 Concluding Remarks

This chapter brings new contributions to the field of provable defenses to adversarial attacks. Principled answers have been provided to key questions on the interest of randomization techniques, and on their loss of accuracy under attack. The obtained bounds have been illustrated in practice by conducting thorough experiments on baseline datasets such as CIFAR. We show in particular that a simple method based on injecting noise drawn from the Exponential family is competitive compared to baseline approaches while leading to provable guarantees. Future work will focus on investigating other noise distributions belonging or not to the Exponential family, combining randomization with more sophisticated defenses and on devising new tight bounds on the adversarial generalization gap.

Appendix D

Advocating for Multiple Defense Strategies against Adversarial Examples

Contents

This Appendix concerns a collaboration with Rafael Pinot, Laurent Meunier and Benjamin Negrevergne. This work has been published in the European Conference on Machine Learning Workshop for CyberSecurity. It conducts a geometrical analysis of defense mechanisms designed to protect neural networks against. This work shows that neural networks designed to be robust against one type of adversarial example offers poor robustness against other types of attacks.

D.1 Introduction

We have seen that deep neural networks are vulnerable to *adversarial examples*. Because it is difficult to characterize the space of visually imperceptible variations of a natural image, existing adversarial attacks use surrogates that can differ from one attack to another. For example, Goodfellow et al. (2015) use the ℓ_∞ norm to measure the distance between the original image and the adversarial image whereas Carlini & Wagner (2017) use the ℓ_2 norm. When the input dimension is low, the choice of the norm is of little importance because the ℓ_∞ and ℓ_2 balls overlap by a large margin, and the adversarial examples lie in the same space. An important insight in this chapter is to observe that the overlap between the two balls diminishes exponentially quickly as the dimensionality of the input space increases. For typical image datasets with large dimensionality, the two balls are mostly disjoint. As a consequence, the ℓ_∞ and the ℓ_2 adversarial examples lie in different areas of the space, and it explains why ℓ_∞ defense mechanisms perform poorly against ℓ_2 attacks and vice versa.

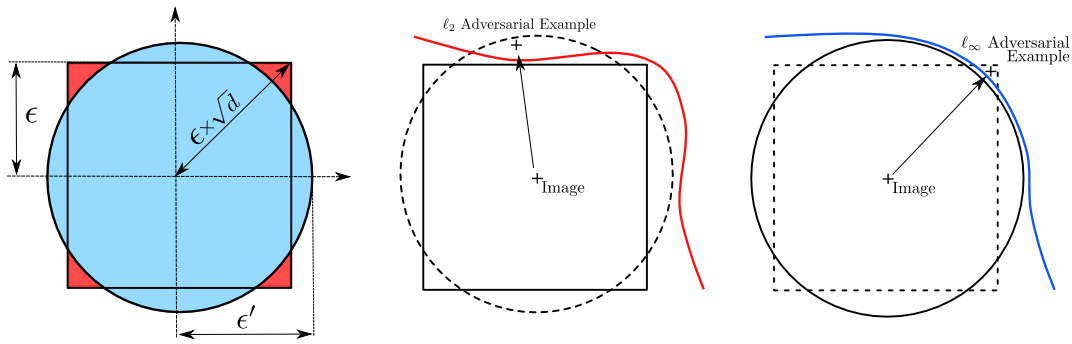
Building on this insight, we advocate for designing models that incorporate defense mechanisms against both ℓ_∞ and ℓ_2 attacks and review several ways of mixing existing defense mechanisms. In particular, we evaluate the performance of *Mixed Adversarial Training* (MAT) (Goodfellow et al. 2015) which consists of augmenting training batches using *both* ℓ_∞ and ℓ_2 adversarial examples, and *Randomized Adversarial Training* (RAT) (Salman et al. 2019), a solution to benefit from the advantages of both ℓ_∞ adversarial training, and ℓ_2 randomized defense.

D.2 No Free Lunch for Adversarial Defenses

In this Section, we show both theoretically and empirically that defenses mechanisms intending to defend against ℓ_∞ attacks cannot provide suitable defense against ℓ_2 attacks. Our reasoning is perfectly general; hence we can similarly demonstrate the reciprocal statement, but we focus on this side for simplicity.

D.2.1 Theoretical analysis

Let us consider a classifier f_∞ that is provably robust against adversarial examples with maximum ℓ_∞ norm of value ϵ_∞. It guarantees that for any input-output pair $(x, y) \sim \mathcal{D}$ and for any perturbation τ such that $\|\tau\|_\infty \leq \epsilon_\infty$, f_∞ is not misled by the perturbation, *i.e.*, $f_\infty(x + \tau) = f_\infty(x)$. We now focus our study on the performance of this classifier against adversarial examples bounded with a ℓ_2 norm of value ϵ_2.



(a) 2D representation of the ℓ_∞ and ℓ_2 balls of respective radius ϵ and ϵ'

(b) A classifier trained with ℓ_∞ adversarial perturbations (materialized by the red line) remains vulnerable to ℓ_2 attacks.

(c) A classifier trained with ℓ_2 adversarial perturbations (materialized by the blue line) remains vulnerable to ℓ_∞ attacks.

Figure D.1: 2-dimensional representation of ℓ_∞ and ℓ_2 balls

Using Figure D.1a, we observe that any ℓ_2 adversarial example that is also in the ℓ_∞ ball, will not fool f_∞. Conversely, if it is outside the ball, we have no guarantee.

To characterize the probability that such an ℓ_2 perturbation fools an ℓ_∞ defense mechanism in the general case (*i.e.*, any dimension d), we measure the ratio between the volume of the intersection of the ℓ_∞ ball of radius ϵ_∞ and the ℓ_2 ball of radius ϵ_2. As Theorem D.1 shows, this ratio depends on the dimensionality d of the input vector x, and rapidly converges to zero when d increases. Therefore a defense mechanism that protects against all ℓ_∞ bounded adversarial examples is unlikely to be efficient against ℓ_2 attacks.

Theorem D.1 (Probability of the intersection goes to 0)**.**
Let

$$B_{2,d}(\epsilon) \triangleq \left\{\tau \in \mathbb{R}^d \mid \|\tau\|_2 \leq \epsilon\right\} \tag{D.1}$$

and

$$B_{\infty,d}(\epsilon') \triangleq \left\{\tau \in \mathbb{R}^d \mid \|\tau\|_\infty \leq \epsilon'\right\}. \tag{D.2}$$

If for all d, we select ϵ and ϵ' such that $\mathrm{Vol}(B_{2,d}(\epsilon)) = \mathrm{Vol}(B_{\infty,d}(\epsilon'))$*, then*

$$\frac{\mathrm{Vol}(B_{2,d}(\epsilon) \bigcap B_{\infty,d}(\epsilon'))}{\mathrm{Vol}(B_{\infty,d}(\epsilon'))} \to 0 \text{ when } d \to \infty. \tag{D.3}$$

Proof of Theorem D.1. Without loss of generality, let us fix $\epsilon = 1$. One can show that for all d,

$$\mathrm{Vol}\left(B_{2,d}\left(\frac{2}{\sqrt{\pi}}\Gamma\left(\frac{d}{2}+1\right)^{1/d}\right)\right) = \mathrm{Vol}(B_{\infty,d}(1)) \tag{D.4}$$

where Γ is the gamma function. Let us denote

$$r_2(d) = \frac{2}{\sqrt{\pi}}\Gamma\left(\frac{d}{2}+1\right)^{1/d}. \tag{D.5}$$

If we denote $\mathcal{U}$, the uniform distribution on $B_{\infty,d}(1)$, we get:

$$\frac{\mathrm{Vol}(B_{2,d}(r_2(d)) \bigcap B_{\infty,d}(1))}{\mathrm{Vol}(B_{\infty,d}(1))} \tag{D.6}$$

$$= \mathbb{P}_{\mathbf{x}\sim\mathcal{U}}[x \in B_{2,d}(r_2(d))] = \mathbb{P}_{\mathbf{x}\sim\mathcal{U}}\left[\sum_{i=1}^{d}|x_i|^2 \leq r_2(d)^2\right] \tag{D.7}$$

$$= \mathbb{P}_{\mathbf{x}\sim\mathcal{U}}\left[\sum_{i=1}^{d}|\mathbf{x}_i|^2 - \mathbb{E}_{\mathbf{x}\sim\mathcal{U}}\left[\sum_{i=1}^{d}|\mathbf{x}_i|^2\right] \leq r_2(d)^2 - \mathbb{E}_{\mathbf{x}\sim\mathcal{U}}\left[\sum_{i=1}^{d}|\mathbf{x}_i|^2\right]\right] \tag{D.8}$$

Note that when d is sufficiently large we get

$$r_2(d)^2 - \mathbb{E}_{\mathbf{x}\sim\mathcal{U}}\left[\sum_{i=1}^{d}|\mathbf{x}_i|^2\right] = r_2(d)^2 - \frac{d}{3} < 0 \tag{D.9}$$

Then, with Hoeffding inequality, we finally obtain:

$$\frac{\mathrm{Vol}(B_{2,d}(r_2(d)) \bigcap B_{\infty,d}(1))}{\mathrm{Vol}(B_{\infty,d}(1))} \leq \exp\left(-\frac{\left(r_2(d)^2 - \frac{d}{3}\right)^2}{d}\right) \tag{D.10}$$

Then, thanks to Stirling's formula

$$r_2(d) \underset{d\to\infty}{\sim} \sqrt{\frac{2}{\pi e}}d^{1/2}. \tag{D.11}$$

Then the ratio between the volume of the intersection of the ball and the volume of the ball converges towards 0 when d goes to ∞. ■

Theorem D.1 states that, when d is large enough, ℓ_2 bounded perturbations have a null probability of being also in the ℓ_∞ ball of the same volume. As a consequence, for any value of d that is large enough, a defense mechanism that offers full protection against ℓ_∞ adversarial examples is not guaranteed to offer any protection against ℓ_2 attacks [1].

Dataset	**Dim. (d)**	**Vol. of the intersection**
–	2	$e^{-0.183}$ (≈ 0.83)
MNIST	784	$e^{-7.344}$
CIFAR	3072	$e^{-29.76}$
ImageNet	150528	$e^{-1478.71}$

Table D.1: Bounds of Theorem D.1 on the volume of the intersection of ℓ_2 and ℓ_∞ balls at equal volume for typical image classification datasets. When $d = 2$, the bound is $10^{-0.183} \approx 0.83$.

Note that this result defeats the 2-dimensional intuition: if we consider a 2 dimensional problem setting, the ℓ_∞ and the ℓ_2 balls have an important overlap (as illustrated in Figure D.1a) and the probability of sampling at the intersection of the two balls is bounded by approximately 83%. However, as we increase the dimensionality d, this probability quickly becomes negligible, even for very simple image datasets such as MNIST. An instantiation of the bound for classical image datasets is presented in Table D.1. The probability of sampling at the intersection of the ℓ_∞ and ℓ_2 balls is close to zero for any realistic image setting. In large dimensions, the volume of the corner of the ℓ_∞ ball is much bigger than it appears in Figure D.1a.

D.2.2 No Free Lunch in Practice

Our theoretical analysis shows that if adversarial examples were uniformly distributed in a high-dimensional space, then any mechanism that perfectly defends against ℓ_∞ adversarial examples has a null probability of protecting against ℓ_2-bounded adversarial attacks. Although existing defense mechanisms do not necessarily assume such a distribution of adversarial examples, we demonstrate that whatever distribution they use, it offers no favorable bias with respect to the result of Theorem D.1. As we discussed in Chapter 5, there are two distinct attack settings: loss maximization (PGD) and perturbation minimization (C&W). Our analysis is mainly focusing on

[1] Theorem D.1 can easily be extended to any two balls with different norms. For clarity, we restrict to the case of ℓ_∞ and ℓ_2 norms.

loss maximization attacks. However, these attacks have a very strict geometry[2]. This is why, to present a deeper analysis of the behavior of adversarial attacks and defenses, we also present a set of experiments that use perturbation minimization attacks.

	Attack PGD-ℓ_2		**Attack PGD-ℓ_∞**	
	Unprotected	**AT-ℓ_∞**	**Unprotected**	**AT-ℓ_2**
Average ℓ_2 norm	0.830	0.830	1.400	1.640
Average ℓ_∞ norm	0.075	0.200	0.031	0.031

Table D.2: Average norms of PGD-ℓ_2 and PGD-ℓ_∞ adversarial examples with and without ℓ_∞ adversarial training on CIFAR-10 ($d = 3072$).

Adversarial training vs. loss maximization attacks To demonstrate that ℓ_∞ adversarial training is not robust against PGD-ℓ_2 attacks we measure the evolution of ℓ_2 norm of adversarial examples generated with PGD-ℓ_∞ between an unprotected model and a model trained with AT-ℓ_∞, *i.e.*, AT where adversarial examples are generated with PGD-ℓ_∞ [3]. Results are presented in Table D.2.

The analysis is unambiguous: the average ℓ_∞ norm of a bounded ℓ_2 perturbation more than double between an unprotected model and a model trained with AT PGD-ℓ_∞. This phenomenon perfectly reflects the illustration of Figure D.1c. The attack will generate an adversarial example on the corner of the ℓ_∞ ball thus increasing the ℓ_∞ norm while maintaining the same ℓ_2 norm. We can observe the same phenomenon with AT-ℓ_2 against PGD-ℓ_∞ attack (see Figure D.1b and Table D.2). PGD-ℓ_∞ attack increases the ℓ_2 norm while maintaining the same ℓ_∞ perturbation thus generating the perturbation in the upper area.

As a consequence, we cannot expect adversarial training ℓ_∞ to offer any guaranteed protection against ℓ_2 adversarial examples .

Adversarial training vs. perturbation minimization attacks. To better capture the behavior of ℓ_2 adversarial examples, we now study the performances of an ℓ_2 perturbation minimization attack (C&W) with and without AT-ℓ_∞. It allows us to understand in which area C&W discovers adversarial examples and

[2]Due to the projection operator, all PGD attacks saturate the constraint, which makes them all lies in a very small part of the ball.

[3]To do so, we use the same experimental setting as in Section D.3 with ϵ_∞ and ϵ_2 such that the volumes of the two balls are equal.

the impact of AT-ℓ_∞. In high dimensions, the red corners (see Figure D.1a) are very far away from the ℓ_2 ball. Therefore, we hypothesize that a large proportion of the ℓ_2 adversarial examples will remain unprotected. To validate this assumption, we measure the proportion of adversarial examples inside of the ℓ_2 ball before and after ℓ_∞ adversarial training. The results are presented in Figure D.2 (left: without adversarial training, right: with adversarial training).

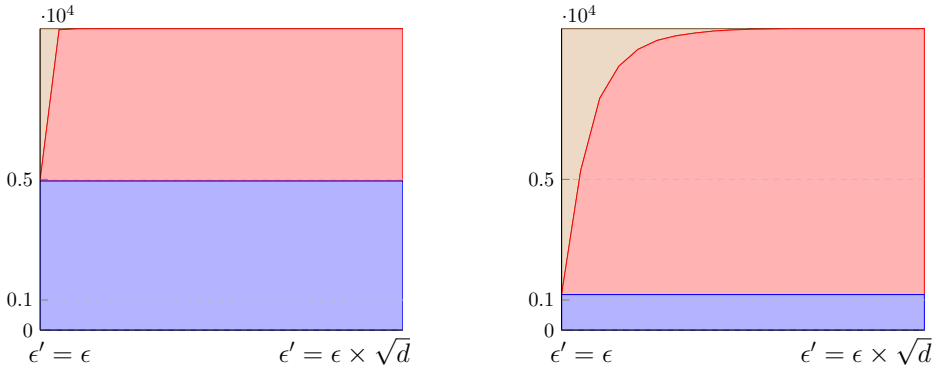


Figure D.2: Comparison of the number of adversarial examples found by C&W, inside the ℓ_∞ ball (lower, blue area), outside the ℓ_∞ ball but inside the ℓ_2 ball (middle, red area) and outside the ℓ_2 ball (upper gray area). ϵ is set to 0.3 and ϵ' varies along the x-axis. Left: without adversarial training, right: with adversarial training. Most adversarial examples have shifted from the ℓ_∞ ball to the cap of the ℓ_2 ball, but remain at the same ℓ_2 distance from the original example.

On both charts, the blue area represents the proportion of adversarial examples that are inside the ℓ_∞ ball. The red area represents the adversarial examples that are outside the ℓ_∞ ball but still inside the ℓ_2 ball (valid ℓ_2 adversarial examples). Finally, the brown-beige area represents the adversarial examples that are beyond the ℓ_2 bound. The radius ϵ' of the ℓ_2 ball varies along the x-axis from ϵ' to $\epsilon'\sqrt{d}$. On the left chart (without adversarial training) most ℓ_2 adversarial examples generated by C&W are inside both balls. On the right chart most of the adversarial examples have been shifted out the ℓ_∞ ball. This is the expected consequence of ℓ_∞ adversarial training. However, these adversarial examples remain in the ℓ_2 ball, *i.e.*, they are in the cap of the ℓ_2 ball. These examples are equally good from the ℓ_2 perspective. This means that even after adversarial training, it is still easy to find good ℓ_2 adversarial examples, making the ℓ_2 robustness of AT-ℓ_∞ almost null.

	Baseline	AT		MAT		NI		RAT-ℓ_∞		RAT-ℓ_2	
		ℓ_∞	ℓ_2	Max	Rand	$\mathcal{N}$	$\mathcal{U}$	$\mathcal{N}$	$\mathcal{U}$	$\mathcal{N}$	$\mathcal{U}$
Natural	0.94	0.85	0.85	0.80	0.80	0.79	0.87	0.74	0.80	0.79	0.87
PGD-ℓ_∞	0.00	0.43	0.37	0.37	0.40	0.23	0.22	0.35	0.40	0.23	0.22
PGD-ℓ_2	0.00	0.37	0.52	0.50	0.55	0.34	0.36	0.43	0.39	0.34	0.37

Table D.3: Comprehensive list of results consisting of the accuracy of several defense mechanisms against ℓ_2 and ℓ_∞ attacks.

D.3 Reviewing Defenses Against Multiple Attacks

Adversarial attacks have been an active topic in the machine learning community since their discovery (Globerson & Roweis, 2006; Biggio et al. 2013; Szegedy et al. 2014). Many attacks have been developed. Most of them solve a loss maximization problem with either ℓ_∞ (Goodfellow et al. 2015; Kurakin et al. 2016; Madry et al. 2018), ℓ_2 (Kurakin et al. 2016; Carlini & Wagner, 2017; Madry et al. 2018), ℓ_1 (Tramèr & Boneh, 2019) or ℓ_0 (Papernot et al. 2016a) surrogate norms. As we showed, these norms are really different in high dimension. Hence, defending against one norm-based attack is not sufficient to protect against another one. In order to solve this problem, we review several strategies to build defenses against multiple adversarial attacks. These strategies are based on the idea that both types of defense must be used simultaneously in order for the classifier to be protected against multiple attacks.

D.3.1 Experimental Setting

To compare the robustness provided by the different defense mechanisms, we use strong adversarial attacks and a conservative setting: the attacker has a total knowledge of the parameters of the model (white-box setting) and we only consider untargeted attacks (a misclassification from one target to any other will be considered as adversarial). To evaluate defenses based on Noise Injection, we use *Expectation Over Transformation* (EOT), the rigorous experimental protocol proposed by Athalye et al. (2017) and later used by Athalye et al. (2018) and Carlini et al. (2019) to identify flawed defense mechanisms.

To attack the models, we use state-of-the-art algorithms PGD. We run PGD with 20 iterations to generate adversarial examples and with 10 iterations when it is used for adversarial training. The maximum ℓ_∞ bound is fixed to 0.031 and the maximum ℓ_2 bound is fixed to 0.83. We chose these values so that the ℓ_∞ and the ℓ_2 balls have similar volumes. Note that 0.83 is slightly above the values typically used in previous

publications in the area, meaning the attacks are stronger, and thus more difficult to defend against.

All experiments are conducted on CIFAR-10 with the Wide-Resnet 28-10 architecture. We use the training procedure and the hyper-parameters described in the original paper by Zagoruyko & Komodakis (2016). Training time varies from 1 day (AT) to 2 days (MAT) on 4 GPUs-V100 servers.

D.3.2 MAT – Mixed Adversarial Training

Earlier results have shown that AT-ℓ_p improves the robustness against corresponding ℓ_p-bounded adversarial examples, and the experiments we present in this section corroborate this observation (See Table D.3, column: AT). Building on this, it is natural to examine the efficiency of *Mixed Adversarial Training* (MAT) against mixed ℓ_∞ and ℓ_2 attacks. MAT is a variation of AT that uses both ℓ_∞-bounded adversarial examples and ℓ_2-bounded adversarial examples as training examples. As discussed by Tramèr & Boneh (2019), there are several possible strategies to mix the adversarial training examples. The first strategy (MAT-Rand) consists in randomly selecting one adversarial example among the two most damaging ℓ_∞ and ℓ_2, and to use it as a training example:

MAT-Rand :

$$\min_{\Omega} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\mathbb{E}_{p\sim\mathcal{U}(\{2,\infty\})} \max_{\|\boldsymbol{\tau}\|_p\leq\epsilon} L(N_\Omega(\mathbf{x}+\boldsymbol{\tau}),y)\right]. \tag{D.12}$$

An alternative strategy is to systematically train the model with the most damaging adversarial example (ℓ_∞ or ℓ_2):

MAT-Max :

$$\min_{\Omega} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\max_{p\in\{2,\infty\}} \max_{\|\boldsymbol{\tau}\|_p\leq\epsilon} L(N_\Omega(\mathbf{x}+\boldsymbol{\tau}),y)\right]. \tag{D.13}$$

The accuracy of MAT-Rand and MAT-Max are reported in Table D.3 (Column: MAT). As expected, we observe that MAT-Rand and MAT-Max offer better robustness both against PGD-ℓ_2 and PGD-ℓ_∞ adversarial examples than the original AT does. More generally, we can see that AT is a good strategy against loss maximization attacks, and thus it is not surprising that MAT is a good strategy against mixed loss maximization attacks. However efficient in practice, MAT (for the same reasons as

AT) lacks theoretical arguments. In order to get the best of both worlds, Salman et al. (2019) proposed to mix adversarial training with randomization.

D.3.3 RAT – Randomized Adversarial Training

We now examine the performance of Randomized Adversarial Training (RAT) first introduced by Salman et al. (2019). This technique mixes Adversarial Training with Noise Injection. The corresponding loss function is defined as follows:

$$\min_{\Omega} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\max_{\|\tau\|_p\leq\epsilon} L\Big(\tilde{N}_\Omega(\mathbf{x}+\tau), y)\Big)\right]. \tag{D.14}$$

where $\tilde{N}_\Omega$ is a randomized neural network with noise injection as described in Appendix C, and $\|\cdot\|_p$ define which kind of AT is used. For each setting, we consider two noise distributions, Gaussian and Uniform as we did with NI. We also consider two different Adversarial training AT-ℓ_∞ as well as AT-ℓ_2.

The results of RAT are reported in Table D.3 (Columns: RAT-ℓ_∞ and RAT-ℓ_2). We can observe that RAT-ℓ_∞ offers the best extra robustness with both noises, which is consistent with previous experiments, since AT is generally more effective against ℓ_∞ attacks whereas NI is more effective against ℓ_2-attacks. Overall, RAT-ℓ_∞ and a noise from uniform distribution offers the best performances but is still weaker than MAT-Rand. These results are also consistent with the literature, since adversarial training (and its variants) is the best defense against adversarial examples so far.

D.4 Concluding Remarks

In this chapter, we tackled the problem of protecting neural networks against multiple attacks crafted from different norms. We demonstrated and gave a geometrical interpretation to explain why most defense mechanisms can only protect against one type of attack. Then we reviewed existing strategies that mix defense mechanisms in order to build models that are robust against multiple adversarial attacks. We conduct a rigorous and full comparison of *Randomized Adversarial Training* and *Mixed Adversarial Training* as defenses against multiple attacks.

We could argue that both techniques offer benefits and limitations. We have observed that MAT offers the best empirical robustness against multiples adversarial attacks but this technique is computationally expensive which hinders its use in large-scale applications. Randomized techniques have the important advantage of

providing theoretical guarantees of robustness and being computationally cheaper. However, the certificate provided by such defenses is still too small for strong attacks. Furthermore, certain Randomized defenses also suffer from the curse of dimensionality as recently shown by Kumar et al. (2020).

Although, randomized defenses based on noise injection seem limited in terms of accuracy under attack and scalability, they could be improved either by Learning the best distribution to use or by leveraging different types of randomization such as discrete randomization first proposed by Pinot et al. (2020). We believe that these certified defenses are the best solution to ensure the robustness of classifiers deployed into real-world applications.

Appendix E

Résumé de la thèse en Français

Contenus

E.1 Introduction

E.1.1 Contexte et Motivation

L'une des percées les plus remarquables de l'apprentissage profond s'est produite en 2012 lors de la compétition de reconnaissance d'image ImageNet (Russakovsky et al. 2015). Cette compétition vise à évaluer différents algorithmes pour la détection d'objets et la classification d'images. En 2012, Krizhevsky et al. ont obtenu la première place et ont battu tous les autres participants avec une marge de plus de 10,8% grâce à un réseau de neurones appelé AlexNet. Les raisons principales de ce succès sont doubles. Premièrement, ils ont utilisé un réseau de neurones convolutif (CNN) avec plus de 60 millions de paramètres, qui était l'un des plus grands modèles de l'époque. Deuxièmement, ils ont conçu une architecture spécifique pour exploiter deux cartes graphiques en parallèle (GPU) afin d'accélérer les opérations arithmétiques, ce qui leur a permis de réduire considérablement le temps d'apprentissage du réseau. La figure E.1 montre un schéma de l'architecture d'AlexNet qui se compose de cinq

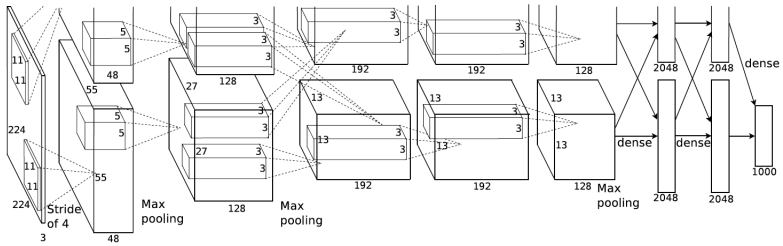


Figure E.1: L'architecture de réseau de neurones convolutifs (AlexNet), proposée par Krizhevsky et al. (2012), qui a remporté la compétition de reconnaissance d'image ImageNet en 2012.

couches convolutives avec deux des couches entièrement connectées à la fin. La figure montre également la répartition de la charge de travail entre les deux GPU.

Après l'introduction d'AlexNet, de nombreuses architectures avec un nombre croissant de paramètres ont été développées. Cette augmentation du nombre de paramètres a conduit à une augmentation de la précision des modèles, dépassant même les performances humaines, sur l'ensemble de données d'ImageNet (He et al. 2015). Le Tableau E.1 montre une liste des différentes architectures de pointe avec leur taille et leur précision. Comme on peut le voir, la précision des modèles s'améliore généralement au prix de la taille du modèle. Pour les modèles de vision par ordinateur, Tan & Le (2019) ont montré que la relation entre la taille du modèle et la précision semble obéir à une loi de puissance. Cette relation a également été observée pour les réseaux neuronaux de traitement du langage naturel (NLP) (Kaplan et al. 2020; Rosenfeld et al. 2020) aidés par la disponibilité de larges ensembles de données tels que le Common Crawl (Raffel et al. 2020) qui constitue près d'un trillion de mots.

Grâce à leur taille et à leur précision accrue, les réseaux de neurones profonds atteignent désormais des performances de pointe dans divers domaines tels que la reconnaissance d'images (LeCun et al. 1998; Krizhevsky et al. 2012; He et al. 2016; Tan & Le, 2019), la détection d'objets (Liu et al. 2016; Redmon et al. 2016; Redmon & Farhadi, 2017), le traitement du langage naturel (Merity et al. 2016; Vaswani et al. 2017; Radford et al. 2019; Brown et al. 2020), speech recognition (Hinton et al. 2012; Abdel-Hamid et al. 2014; Yu & Deng, 2016), le domaine de la santé (Faust et al. 2018) etc. Les modèles de vision par ordinateur et de traitement du langage naturel ont atteint des performances suffisantes pour être utilisés dans des applications du monde réel telles

Auteurs	Modèles	#Params	TOP-5 Préc.
Krizhevsky et al. (2012)	AlexNet	61 M	84.7%
Simonyan & Zisserman (2014)	VGG	144 M	92.0%
He et al. (2016)	ResNet-152	60 M	93.8%
Szegedy et al. (2017)	Inception-ResNet-v2	56 M	95.1%
Xie et al. (2017)	ResNeXt-101	84 M	95.6%
Hu et al. (2018)	SENet	146 M	96.2%
Real et al. (2019)	AmoebaNet-A	469 M	96.7%
Huang et al. (2019)	AmoebaNet-B	556 M	97.0%

(a) Modèles de reconnaissance d'images

Auteurs	Modèles	#Params
Peters et al. (2018)	ELMo	94 M
Radford et al. (2018)	GPT	110 M
Devlin et al. (2019)	BERT	340 M
Yang et al. (2019)	XLNet (Large)	340 M
Liu et al. (2019)	RoBERTa (Large)	355 M
Radford et al. (2019)	GPT-2	1 B
Shoeybi et al. (2019)	MegatronLM	8 B
Raffel et al. (2020)	T5-11B	11 B
Rosset (2020)	T-NLG	17 B
Brown et al. (2020)	GPT-3	175 B
Fedus et al. (2021)	Switch Transformers	1 T

(b) Modèles de traitement automatique des langues

Tableau E.1 : Évolution du nombre de paramètres des modèles de reconnaissance d'image et de traitement du langage naturel développés dans les années qui ont suivi l'architecture AlexNet.

que les véhicules autonomes (Fagnant & Kockelman, 2015; Sharma & Zheng, 2021), la traduction (Wu et al. 2016), les assistants vocaux (Li et al. 2017a), etc.

Cependant, la précision des modèles ne devrait pas être la seule préoccupation, lorsqu'ils sont mis en œuvre dans un processus de décision critique, les réseaux de neurones doivent être compacts, efficaces et sécurisés. Bien que précis, les grands réseaux de neurones n'ont souvent pas ces propriétés. En effet, l'entraînement de modèles de pointe sur des tâches de reconnaissance d'image ou de traitement du langage naturel nécessite des gigaoctets de mémoire et peut prendre plusieurs mois sur un seul GPU (Krizhevsky et al. 2012; Brown et al. 2020). Par exemple, le modèle GPT-3 proposé par Brown et al. (2020), culmine à 175 milliards de paramètres et

l'entraînement durerait 355 ans sur un seul GPU et coûterait $4 600 000 sur une plateforme de cloud computing (Li, 2020). Il a également été estimé par Strubell et al. (2019) que la formation et le développement du modèle Transformer proposé par Vaswani et al. (2017) avec l'optimisation des hyperparamètres émettraient environ 284 019 kg de CO_2 alors qu'une vie humaine consomme en moyenne seulement 5 000 kg de CO_2 pendant un an. En outre, avec l'essor des smartphones et des objets connectés aux ressources de calcul et de mémoire limitées, les réseaux de neurones doivent également être efficaces pendant la phase d'inférence, c'est-à-dire, la phase d'exécution du modèle. De plus, avec la préoccupation croissante concernant la confidentialité des données, des méthodes telles que l'"apprentissage collaboratif" gagnent du terrain. L'apprentissage collaboratif consiste à entraîner un modèle sur plusieurs appareils décentralisés (par exemple les smartphones) avec des échantillons de données locales. Cela permet d'éviter l'étape de centralisation de toutes les données des utilisateurs sur un seul serveur, ce qui permet de résoudre le problème de la confidentialité des données. Ainsi, la construction de réseaux de neurones compacts et efficaces reste un objectif important afin de réduire le temps d'entraînement, de diminuer les coûts et de permettre une R&D plus rapide.

En plus d'être compacts et efficients, les réseaux de neurones doivent également être sécurisés. En raison de leur grande complexité et expressivité, les larges réseaux de neurones sont instables aux petites perturbations. Ainsi, cette instabilité mène à des vulnérabilités face aux *exemples antagonistes*, c'est-à-dire aux variations imperceptibles des exemples naturels, conçus pour tromper délibérément les modèles (Globerson & Roweis, 2006; Biggio et al. 2013; Szegedy et al. 2014). La Figure E.3 présente un exemple antagoniste sur une image. La petite perturbation (au centre) est ajoutée à l'image originale (à gauche), ce qui donne une image contradictoire (à droite). Ce comportement peut causer de graves problèmes de sécurité lorsque des réseaux neuronaux sont utilisés pour des prises de décisions critiques (par exemple, les décisions judiciaires, les voitures autonomes, etc.).

Cette thèse se concentre sur l'entraînement de réseaux de neurones qui sont non seulement précis, mais aussi compacts, efficients, faciles à entraîner, fiables et robustes aux exemples antagonistes.

E.1.2 Problématiques et Contributions

Les réseaux de neurones, qui trouvent leurs racines dans les travaux de McCulloch & Pitts (1943) and Rosenblatt (1958), peuvent être décrits analytiquement comme une composition de fonctions linéaires entrelacées avec des fonctions non linéaires

$$\begin{pmatrix} a & & & \\ & b & & \\ & & c & \\ & & & d \end{pmatrix} \qquad \begin{pmatrix} a & b & c & d \\ e & a & b & c \\ f & e & a & b \\ g & f & e & a \end{pmatrix} \qquad \begin{pmatrix} ae & af & ag & ah \\ be & bf & bg & bh \\ ce & cf & cg & ch \\ de & df & dg & dh \end{pmatrix} \qquad \begin{pmatrix} a & a^2 & a^3 & a^4 \\ b & b^2 & b^3 & b^4 \\ c & c^2 & c^3 & c^4 \\ d & d^2 & d^3 & d^4 \end{pmatrix}$$

diagonal Toeplitz Low Rank Vandermonde

Figure E.2: Exemples de matrices structurées.

(également appelées fonctions d'activation). Plus formellement, un réseau de neurones est une fonction $N_\Omega : \mathbb{R}^n \to \mathbb{R}^m$ paramétrée par un ensemble de poids Ω de la forme:

$$N_\Omega(\mathbf{x}) = \psi^{(p)} \circ \rho \circ \psi^{(p-1)} \cdots \circ \psi^{(2)} \circ \rho \circ \psi^{(1)}(\mathbf{x}) \ . \tag{E.1}$$

Ici, p correspond à la *profondeur* du réseau (c'est-à-dire, le nombre de couches) et ρ est une fonction non linéaire. Enfin, chaque $\psi^{(i)}$ est une fonction linéaire multidimensionnelle $\psi^{(i)} : \mathbf{x} \mapsto \mathbf{W}^{(i)}\mathbf{x} + \mathbf{b}^{(i)}$ paramétrée par une matrice de poids $\mathbf{W}^{(i)}$ et un biais $\mathbf{b}^{(i)}$ et Ω est l'union des paramètres de chaque couche.

Si les réseaux de neurones n'ont pas de restriction sur les matrices de poids $\mathbf{W}^{(i)}$, on dit que les couches sont *entièrement connectées*. En règle générale, les réseaux de neurones entièrement connectés ont un grand nombre de paramètres. Par exemple, un réseau de neurones entièrement connecté avec p couches et des n neurones sur chaque couche ($\mathbf{W}^{(i)} \in \mathbb{R}^{n \times n}$) aura $\mathcal{O}(pn(n+1))$ paramètres. Comme les dimensions d'entrée et de sortie sont généralement importantes (par exemple, le jeu de données ImageNet a une dimension d'entrée de $224^2 \times 3$ et une sortie de 1000), les réseaux de neurones entièrement connectés avec peu de couches peuvent facilement accumuler des centaines de millions de paramètres. Il a été montré que ce type de réseau est peu performant, car l'entraînement n'optimise pas suffisamment bien les paramètres en raison d'un grand espace de recherche. En outre, l'entraînement est long et complexe, ce qui les rend peu pratiques pour un certain nombre de cas d'usage (smartphones, objets connectés, etc.). Pour réduire le nombre de paramètres sur chaque couche, de nombreux chercheurs ont mis au point des opérations linéaires spécifiques qui réduisent le nombre de paramètres et ont de nombreuses propriétés intéressantes.

Les réseaux de neurones convolutifs (CNN), qui utilisent des opérations linéaires spécialisées et plus compactes, sont considérés comme l'état de l'art concernant les tâches de vision par ordinateur (LeCun et al. 1998; Krizhevsky et al. 2012; He et al. 2016; Tan & Le, 2019). Ces réseaux neuronaux convolutifs utilisent des matrices de

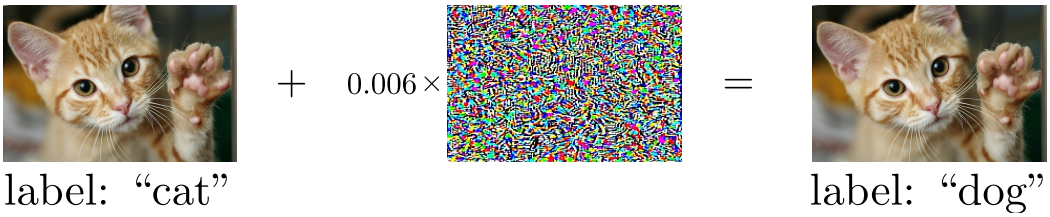


Figure E.3: Exemple d’exemple antagoniste avec une image.

poids spécifiques qui permettent une invariance du traitement par translation, ce qui est souhaitable pour traiter des images. Alors qu’une couche linéaire classique avec une matrice dense a $n \times n$ paramètres, une couche convolutionnelle n’a que $k \times k$ paramètres avec k qui correspond à la taille du noyau et qui est généralement petite (3 ou 5 pour les couches convolutionnelles classiques). Un réseau neuronal convolutif est le type le plus courant de réseau neuronal *structuré*. En effet, l’opération de convolution peut être représentée par une matrice structurée, c’est-à-dire, une matrice qui peut être représentée avec moins de n^2 de paramètres.

En plus d’offrir une représentation plus compacte, la structure de certaines matrices peut être exploitée afin d’obtenir de meilleurs algorithmes pour multiplier la matrice avec un vecteur, cela permet d’optimiser la mémoire et de réduire le nombre d’opérations réalisées. En se basant sur le succès des réseaux de neurones convolutifs, les chercheurs ont étudié et proposé d’autres types de réseaux basés sur des matrices de poids avec différentes structures (Sindhwani et al. 2015; Moczulski et al. 2016). La Figure E.2 montre différents types de matrices structurées qui ont été utilisées pour l’apprentissage profond. Bien que les réseaux de neurones convolutifs soient état de l’art pour les tâches de vision par ordinateur, il reste à savoir si d’autres types de réseaux structurés pourraient être utiles à d’autres types d’applications et quel type de structure pourraient fournir à la fois précision et efficacité de calcul.

Les contributions de cette thèse se situent à l’intersection de l’algèbre linéaire, l’analyse de Fourier et de l’apprentissage profond. En conséquence, nous construisons des réseaux de neurones compacts et sécurisés en exploitant les propriétés des matrices structurées issues de la famille de Toeplitz. Ci-après, nous détaillons nos contributions.

Entraînement de Réseaux de Neurones Compacts

Comme première contribution, nous étudions les réseaux de neurones dans lesquels les matrices de poids sont le produit des matrices diagonales et circulantes. Les

matrices circulantes sont un type particulier de matrice de Toeplitz. Cette nouvelle architecture compacte permet de remplacer les réseaux de neurones entièrement connectés tout en maintenant la performance. Outre une analyse théorique de leur expressivité, nous introduisons de nouvelles techniques pour l'entraînement de ces modèles : nous concevons une procédure d'initialisation et proposons une utilisation intelligente des fonctions de non-linéarité afin de faciliter leur entraînement. Nous montrons que ces modèles sont plus précis que les autres approches structurées tout en nécessitant deux fois moins de poids que les meilleures approches. Enfin, nous entraînons des réseaux de neurones profonds basés sur des matrices diagonales et circulantes sur un ensemble de données de classification vidéo qui contient plus de 3.8 millions d'exemples.

L'analyse expérimentale des réseaux de neurones profonds basée sur des matrices diagonales et circulantes sur l'ensemble de données de classification vidéo a été publiée dans le cadre de l'**Atelier sur la reconnaissance de vidéo de la Conférence Européenne de Vision par Ordinateur**. Ce travail a été réalisé dans le cadre de la compétition *YouTube-8M* organisée par Google. Ensuite, l'analyse théorique de l'expressivité de ces réseaux a été publiée dans un deuxième article dans le cadre de la **24e Conférence Européenne sur l'Intelligence Artificielle.**

Entraînement de Réseaux de Neurones Robustes

Comme deuxième contribution, nous proposons une procédure pour entraîner des réseaux de neurones robustes en étudiant les propriétés de la structure des convolutions. Nous concevons une nouvelle borne supérieure des valeurs singulières des couches de convolution, qui est à la fois précise et facile à calculer. Notre travail est basé sur le résultat de Gray (2006) qui indique qu'une borne supérieure des valeurs singulières des matrices de Toeplitz peut être calculée à partir de la transformée de Fourier inverse de la séquence caractéristique de ces matrices. De notre analyse découle immédiatement un algorithme de régularisation de la constante de Lipschitz d'une couche convolutive, et par extension de la constante de Lipschitz de l'ensemble du réseau. Enfin, nous utilisons notre approche pour améliorer la robustesse des réseaux de neurones convolutifs. Des travaux récents ont montré que les méthodes empiriques telles que l'entraînement contradictoire offrent une faible généralisation (Schmidt et al. 2018) et peuvent être améliorées en appliquant une régularisation Lipschitz (Farnia et al. 2019). Pour illustrer l'avantage de notre nouvelle méthode, nous entraînons des réseaux de neurones avec la régularisation Lipschitz et montrons qu'elle offre une amélioration significative de robustesse.

Le principal résultat des travaux décrits dans le Chapitre 5 a été publié dans le cadre de la **35e Conférence AAAI sur l'Intelligence Artificielle**. D'autres contributions conjointes ont également été publiées sur le thème des réseaux neuronaux robustes. La première, publiée dans le cadre de la **Conférence en Intelligence Artificielle et Neurosciences Computationnelles**, étudie l'efficacité de l'injection de bruit à l'entraînement et à l'inférence dans le réseau pour protéger contre les attaques adverses. Dans ce travail, nous montrons que le bruit tiré de la famille exponentielle offre une protection garantie contre les attaques adverses. La deuxième contribution conjointe, publiée dans le cadre de l'**Atelier de Cybersécurité de la Conférence Européenne de l'Apprentissage Automatique**, effectue une analyse géométrique des mécanismes de défense destinés à protéger les réseaux neuronaux contre différents types d'attaques. Ce travail montre que les réseaux neuronaux conçus pour être robustes contre un type d'attaque adverse offrent peu de protection contre d'autres types d'attaques.

E.2 Réseaux de Neurones Compacts basés sur les matrices Diagonales et Circulantes

Ces dernières années, la conception de réseaux neuronaux compacts et performants a été un sujet de recherche actif. Ce domaine est motivé par des applications pratiques dans les systèmes embarqués (pour réduire l'empreinte mémoire (Sainath & Parada, 2015)), l'apprentissage fédéré et distribué (pour réduire la communication (Konečný et al. 2016)), etc. Outre un certain nombre d'applications pratiques, la question de savoir si les modèles doivent réellement être aussi larges ou si des réseaux plus petits peuvent atteindre une précision similaire est également une question de recherche importante.

Les matrices structurées sont au cœur même de la plupart des travaux sur les réseaux compacts. Dans ces modèles, les matrices de poids dense sont remplacées par des matrices ayant une structure précise (par exemple, les matrices de rang faible, les matrices de Toeplitz, les matrices circulantes, LDR, etc.) Malgré des efforts importants (Cheng et al. (2015) and Moczulski et al. (2016)), les performances des modèles compacts sont encore loin d'atteindre une précision acceptable motivant leur utilisation dans des scénarios du monde réel. Cela soulève plusieurs questions sur l'efficacité de ces modèles et sur notre capacité à les entraîner. En particulier, deux questions principales appellent à investigation :

1. Quelle est l'expressivité des couches structurées par rapport aux couches denses ?
2. Comment entraîner efficacement des réseaux neuronaux profonds avec un grand nombre de couches structurées ?

Dans cette thèse, nous nous efforçons de répondre à ces questions en étudiant les réseaux neuronaux basés sur les matrices diagonales et circulantes (*a.k.a.* DCNN), qui sont des réseaux neuronaux profonds dans lesquels les matrices de poids sont le produit des matrices diagonales et circulantes. L'idée d'utiliser ensemble des matrices diagonales et circulantes vient d'une série de résultats en algèbre linéaire par Müller-Quade et al. (1998) et Huhtanen & Perämäki (2015).

Pour répondre à la première question, nous proposons une analyse de l'expressivité des DCNN en étendant les résultats obtenus par Huhtanen & Perämäki (2015) qui indique que toute matrice peut être décomposée en un produit de $2n - 1$ matrices diagonales et circulantes. Nous introduisons une nouvelle borne sur le nombre de produits requis pour approcher une matrice qui dépend de son rang. Sur la base de ce résultat, nous démontrons qu'un DCNN avec une largeur limitée et une faible profondeur peut être autant expressif que n'importe quel réseau de neurones dense avec des activations ReLU.

Pour répondre à la deuxième question, nous décrivons d'abord une procédure d'initialisation pour les DCNN qui permet au signal de se propager à travers le réseau sans disparaître ou exploser. En outre, nous fournissons un certain nombre d'expériences pour expliquer le comportement des DCNN et montrer l'impact du nombre de non-linéarités dans le réseau sur le taux de convergence et la précision. En combinant toutes ces connaissances, nous sommes en mesure de former des DCNN de grande taille et de grande profondeur. Pour finir, nous démontrons les bonnes performances de ces réseaux dans le contexte de la reconnaissance de vidéo.

E.3 Constante de Lipschitz des Couches Convolutionnelles

Ces dernières années ont vu un intérêt croissant pour la régularisation Lipschitz des réseaux de neurones, dans le but d'améliorer leur généralisation (Bartlett et al. 2017), leur robustesse aux attaques adverses (Tsuzuku et al. 2018; Farnia et al. 2019), ou leurs capacités de génération (par exemple pour les GANs : Arjovsky et al. (2017) and Miyato et al. (2018)). Malheureusement, le calcul exact de la constante de Lipschitz d'un réseau de neurones est un problème NP-complet (Virmaux & Scaman, 2018)

et en pratique, les techniques existantes telles que celles proposées par Virmaux & Scaman (2018), Fazlyab et al. (2019) ou Latorre et al. (2020) sont difficiles à mettre en œuvre pour les réseaux neuronaux à plus d'une ou deux couches, ce qui entrave leur utilisation dans les applications d'apprentissage profond.

Pour surmonter cette difficulté au lieu de calculer la constante globale, la plupart des travaux se sont concentrés sur le calcul de la constante de Lipschitz des *couches* du réseau. Le produit des constantes de Lipschitz de chaque couche est une borne supérieure de la constante de Lipschitz de l'ensemble du réseau, et elle peut être utilisée comme substitut pour effectuer une régularisation Lipschitz. Comme la plupart des fonctions d'activation courantes (telles que la ReLU) ont une constante de Lipschitz égale à un, la principale difficulté consiste à calculer la constante de Lipschitz de l'application linéaire sous-jacente qui est égale à sa plus grande valeur singulière. Les travaux dans ce domaine de recherche s'appuient principalement sur un célèbre algorithme itératif appelé *méthode de la puissance itérée* (Golub & Van der Vorst, 2000) utilisée pour approximer la valeur singulière maximale d'une fonction linéaire. Bien que générique et précise, cette technique est également coûteuse en termes de calcul, ce qui en empêche son utilisation pour l'entraînement de larges réseaux de neurones.

Dans cette thèse, nous introduisons une nouvelle borne supérieure des valeurs singulières des couches de convolution, qui est à la fois précise et facile à calculer. Au lieu d'utiliser la méthode de la puissance itérée pour approximer cette valeur, nous nous appuyons sur la théorie des matrices de Toeplitz et ses liens avec l'analyse de Fourier. Notre travail est basé sur le résultat de Gray (2006) qui indique qu'une borne supérieure des valeurs singulières des matrices de Toeplitz peut être calculée à partir de la transformée de Fourier inverse de la séquence caractéristique de ces matrices. Nous étendons d'abord ce résultat aux matrices de Toeplitz par blocs de Toeplitz (c'est-à-dire une matrice de Toeplitz par blocs où chaque bloc est également Toeplitz) et ensuite aux opérateurs convolutionnels. De notre analyse découle immédiatement un algorithme de régularisation de la constante de Lipschitz d'une couche convolutive, et par extension de la constante de Lipschitz de l'ensemble du réseau. Nous étudions théoriquement l'approximation de cet algorithme et montrons expérimentalement qu'il est plus efficace et plus précis que les approches concurrentes.

Enfin, nous illustrons notre approche sur la robustesse aux exemples antagonistes. Des travaux récents ont montré que les méthodes empiriques, telles que la formation contradictoire (*Adversarial Training* ou AT), offrent une faible généralisation (Schmidt et al. 2018) et peuvent être améliorées en appliquant une régularisation Lipschitz (Far-

nia et al. 2019). Pour illustrer les avantages de notre nouvelle méthode, nous entraînons un large réseau de neurones avec AT et la régularisation Lipschitz et montrons qu'elle offre une amélioration significative par rapport à un entraînement contradictoire seul et par rapport aux autres méthodes de régularisation Lipschitz. En résumé, nous apportons les trois contributions suivantes :

1. Nous proposons une nouvelle borne supérieure des valeurs singulières des couches convolutionnelles en nous appuyant sur la théorie des matrices de Toeplitz et ses liens avec l'analyse de Fourier.

2. Nous proposons un algorithme efficace pour calculer cette borne qui permet son utilisation dans le contexte des réseaux neuronaux convolutifs.

3. Nous utilisons notre méthode pour régulariser la constante de Lipschitz des réseaux de neurones et montrons qu'elle permet un gain significatif de robustesse face aux attaques adverses.

References

Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org (see page 130).

Abdel-Hamid, O., A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu (2014). "Convolutional neural networks for speech recognition". *IEEE/ACM Transactions on audio, speech, and language processing* (see pages 2, 168).

Absil, P.-A., R. Mahony, and R. Sepulchre (2009). *Optimization algorithms on matrix manifolds*. Princeton University Press (see page 51).

Abu-El-Haija, S., N. Kothari, J. Lee, A. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan (2016). "YouTube-8M: A Large-Scale Video Classification Benchmark". *arXiv preprint arXiv:1609.08675* (see pages 81–84, 126, 128, 131).

Anil, C., J. Lucas, and R. Grosse (2019). "Sorting out Lipschitz Function Approximation". In: *Proceedings of the 36th International Conference on Machine Learning* (ICML) (see page 51).

Anthony, M. and P. L. Bartlett (1999). *Neural Network Learning: Theoretical Foundations*. Cambridge University Press (see page 32).

Arandjelović, R., P. Gronat, A. Torii, T. Pajdla, and J. Sivic (2016). "NetVLAD: CNN architecture for weakly supervised place recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) (see page 126).

Arjovsky, M., S. Chintala, and L. Bottou (2017). "Wasserstein Generative Adversarial Networks". In: *International Conference on Learning Representations* (ICLR) (see page 175).

Arora, S., N. Cohen, W. Hu, and Y. Luo (2019). "Implicit regularization in deep matrix factorization". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see page 73).

Athalye, A., N. Carlini, and D. Wagner (2018). "Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples". In: *Proceedings of the 35th International Conference on Machine Learning* (ICML) (see pages 140, 142, 149, 162).

Athalye, A., L. Engstrom, A. Ilyas, and K. Kwok (2017). "Synthesizing Robust Adversarial Examples". *arXiv preprint arXiv:1707.07397* (see pages 142, 151, 162).

Avram, F. (1988). "On bilinear forms in Gaussian random variables and Toeplitz matrices". *Probability Theory and Related Fields* (see page 14).

Ba, J. and R. Caruana (2014). "Do Deep Nets Really Need to be Deep?" In: *Advances in Neural Information Processing Systems* (NeurIPS) (see page 39).

Bahdanau, D., K. Cho, and Y. Bengio (2015). "Neural machine translation by jointly learning to align and translate". In: *International Conference on Learning Representations* (ICLR) (see page 2).

Bartlett, P. L., D. J. Foster, and M. J. Telgarsky (2017). "Spectrally-normalized margin bounds for neural networks". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see pages 32–34, 45, 50, 175).

Bartlett, P. L., V. Maiorov, and R. Meir (1998). "Almost linear VC-dimension bounds for piecewise polynomial networks". *Neural computation* (see page 32).

Basor, E. (2017). "Asymptotics of determinants of block Toeplitz matrices". *Random Matrices: Theory and Applications* 6:04, page 1740003 (see page 113).

Beaudry, N. J. and R. Renner (2012). "An Intuitive Proof of the Data Processing Inequality". *Quantum Info. Comput.* (see page 147).

Ben-David, S., N. Eiron, and P. M. Long (2003). "On the difficulty of approximately maximizing agreements". *Journal of Computer and System Sciences* (see page 27).

Ben-Tal, A., L. El Ghaoui, and A. Nemirovski (2009). *Robust optimization*. Princeton University Press (see page 141).

Bengio, Y., P. Simard, and P. Frasconi (1994). "Learning Long-term Dependencies with Gradient Descent is Difficult". *IEEE Transactions on Neural Networks* (see page 28).

Bibi, A., B. Ghanem, V. Koltun, and R. Ranftl (2019). "Deep Layers as Stochastic Solvers". In: *International Conference on Learning Representations* (ICLR) (see page 53).

Biggio, B., I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli (2013). "Evasion attacks against machine learning at test time". In: *Joint European conference on machine learning and knowledge discovery in databases*. Springer (see pages 4, 29, 162, 170).

Björck, Å. and C. Bowie (1971). "An iterative algorithm for computing the best estimate of an orthogonal matrix". *SIAM Journal on Numerical Analysis* (see page 51).

Böttcher, A. and B. Silbermann (2012). *Introduction to large truncated Toeplitz matrices*. Springer Science & Business Media (see page 14).

Brown, T. B., B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, G. Herbert-Voss Arieland Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei (2020). "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see pages 2, 3, 38, 112, 115, 168, 169).

Buciluă, C., R. Caruana, and A. Niculescu-Mizil (2006). "Model Compression". In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM (see page 38).

Carlini, N., A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, and A. Madry (2019). "On Evaluating Adversarial Robustness". *arXiv preprint arXiv:1902.06705* (see pages 109, 143, 151, 162).

Carlini, N. and D. Wagner (2017). "Towards evaluating the robustness of neural networks". In: *IEEE Symposium on Security and Privacy* (see pages 30, 31, 109, 110, 140, 142, 150, 156, 162).

Chapeau-Blondeau, F. and D. Rousseau (2004). "Noise-enhanced performance for an optimal Bayesian estimator". *IEEE Transactions on Signal Processing* (see page 141).

Chen, P.-Y., Y. Sharma, H. Zhang, J. Yi, and C.-J. Hsieh (2018). "EAD: Elastic-Net Attacks to Deep Neural Networks via Adversarial Examples". In: *Proceedings of the 32nd AAAI Conference on Artificial Intelligence* (see pages 142, 150).

Chen, W., J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen (2015). "Compressing Neural Networks with the Hashing Trick". In: *Proceedings of the 32nd International Conference on Machine Learning* (ICML) (see pages 81, 82).

Cheng, Y., F. X. Yu, R. S. Feris, S. Kumar, A. Choudhary, and S. F. Chang (2015). "An Exploration of Parameter Redundancy in Deep Networks with Circulant Projections". In: *IEEE International Conference on Computer Vision* (ICCV) (see pages 41, 42, 45, 57, 67, 130, 132, 135, 174).

Choromanski, K., M. Rowland, W. Chen, and A. Weller (2019). "Unifying orthogonal monte carlo methods". In: *Proceedings of the 36th International Conference on Machine Learning* (ICML) (see page 44).

Cisse, M., P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier (2017). "Parseval Networks: Improving Robustness to Adversarial Examples". In: *Proceedings of the 34th International Conference on Machine Learning* (ICML) (see page 50).

Cohen, J., E. Rosenfeld, and Z. Kolter (2019). "Certified Adversarial Robustness via Randomized Smoothing". In: *Proceedings of the 36th International Conference on Machine Learning* (ICML) (see page 142).

Collins, M. D. and P. Kohli (2014). "Memory Bounded Deep Convolutional Networks". arXiv preprint arXiv:1412.1442 (see page 39).

Cooley, J. W. and J. W. Tukey (1965). "An algorithm for the machine calculation of complex Fourier series". *Mathematics of computation* (see page 12).

Couellan, N. (2019). "The Coupling Effect of Lipschitz Regularization in Deep Neural Networks". *arXiv preprint arXiv:1904.06253* (see page 50).

Courbariaux, M., Y. Bengio, and J.-P. David (2015). "BinaryConnect: Training Deep Neural Networks with Binary Weights During Propagations". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see page 38).

Cover, T. M. and J. A. Thomas (2012). *Elements of information theory*. John Wiley & Sons (see page 147).

Cybenko, G. (1989). "Approximation by Superpositions of a Sigmoidal Function". *Mathematics of control, signals and systems* 2:4, pages 303–314 (see page 66).

Dai, B., C. Zhu, B. Guo, and D. Wipf (2018). "Compressing Neural Networks using the Variational Information Bottleneck". In: *Proceedings of the 35th International Conference on Machine Learning* (ICML) (see pages 38, 39).

Dao, T., A. Gu, M. Eichhorn, A. Rudra, and C. Re (2019). "Learning Fast Algorithms for Linear Transforms Using Butterfly Factorizations". In: *Proceedings of the 36th International Conference on Machine Learning* (ICML) (see page 44).

Dao, T., N. Sohoni, A. Gu, M. Eichhorn, A. Blonder, M. Leszczynski, A. Rudra, and C. Ré (2020). "Kaleidoscope: An Efficient, Learnable Representation For All Structured Linear Maps". In: *International Conference on Learning Representations* (ICLR) (see page 44).

Davis, P. (1979). *Circulant Matrices*. Monographs and textbooks in pure and applied mathematics. Wiley (see page 11).

De La Chevrotiere, G. (2009). "Finding the maximum modulus of a polynomial on the polydisk using a generalization of steckins lemma". *SIAM Undergraduate Research Online* (see page 99).

Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei (2009). "Imagenet: A Large-Scale Hierarchical Image Database". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) (see page 41).

Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (NAACL) (see pages 3, 169).

Dhillon, G. S., K. Azizzadenesheli, J. D. Bernstein, J. Kossaifi, A. Khanna, Z. C. Lipton, and A. Anandkumar (2018). "Stochastic Activation Pruning for Robust Adversarial Defense". In: *International Conference on Learning Representations* (ICLR) (see pages 140, 142, 144).

Diochnos, D., S. Mahloujifar, and M. Mahmoody (2018). "Adversarial Risk and Robustness: General Definitions and Implications for the Uniform Distribution". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see pages 143, 144).

Dumoulin, V. and F. Visin (2016). "A Guide to Convolution Arithmetic for Deep Learning". *arXiv preprint arXiv:1603.07285* (see page 17).

Dwork, C., A. Roth et al. (2014). "The algorithmic foundations of differential privacy". *Foundations and Trends® in Theoretical Computer Science* (see page 142).

Evci, U., T. Gale, J. Menick, P. S. Castro, and E. Elsen (2020). "Rigging the Lottery: Making All Tickets Winners". In: *Proceedings of the 37th International Conference on Machine Learning* (ICML) (see page 39).

Fagnant, D. J. and K. Kockelman (2015). "Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations". *Transportation Research Part A: Policy and Practice* (see page 169).

Farnia, F., J. Zhang, and D. Tse (2019). "Generalizable Adversarial Training via Spectral Normalization". In: *International Conference on Learning Representations* (ICLR) (see pages 7, 32, 34, 35, 45, 50, 52, 53, 56, 88, 102, 104, 105, 173, 175, 176).

Faust, O., Y. Hagiwara, T. J. Hong, O. S. Lih, and U. R. Acharya (2018). "Deep learning for healthcare applications based on physiological signals: A review". *Computer Methods and Programs in Biomedicine* (see page 168).

Fawzi, A., H. Fawzi, and O. Fawzi (2018a). "Adversarial vulnerability for any classifier". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see pages 140, 143).

Fawzi, A., S.-M. Moosavi-Dezfooli, and P. Frossard (2016). "Robustness of classifiers: from adversarial to random noise". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see page 140).

Fawzi, A., S.-M. Moosavi-Dezfooli, P. Frossard, and S. Soatto (2018b). "Empirical Study of the Topology and Geometry of Deep Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) (see page 140).

Fazlyab, M., M. Morari, and G. J. Pappas (2019a). "Safety Verification and Robustness Analysis of Neural Networks via Quadratic Constraints and Semidefinite Programming". *arXiv preprint arXiv:1903.01287* (see page 47).

Fazlyab, M., A. Robey, H. Hassani, M. Morari, and G. Pappas (2019b). "Efficient and Accurate Estimation of Lipschitz Constants for Deep Neural Networks". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see pages 47, 48, 176).

Fedus, W., B. Zoph, and N. Shazeer (2021). "Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity". *arXiv preprint arXiv:2101.03961* (see pages 3, 112, 169).

Feldman, V., V. Guruswami, P. Raghavendra, and Y. Wu (2012). "Agnostic Learning of Monomials by Halfspaces is Hard". *SIAM Journal on Computing* (see page 27).

Frankle, J. and M. Carbin (2018). "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks". In: *International Conference on Learning Representations* (ICLR) (see pages 38, 39).

Fukushima, K. and S. Miyake (1982). "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition". In: *Competition and cooperation in neural nets*. Springer, pages 267–285 (see page 6).

Garipov, T., D. Podoprikhin, A. Novikov, and D. Vetrov (2016). "Ultimate Tensorization: Compressing Convolutional and FC Layers Alike". *arXiv preprint arXiv:1611.03214* (see page 41).

Globerson, A. and S. Roweis (2006). "Nightmare at Test Time: Robust Learning by Feature Deletion". In: *Proceedings of the 23rd International Conference on Machine Learning* (ICML) (see pages 4, 29, 162, 170).

Glorot, X. and Y. Bengio (2010). "Understanding the Difficulty of Training Deep Feedforward Neural Networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (see page 73).

Golowich, N., A. Rakhlin, and O. Shamir (2018). "Size-Independent Sample Complexity of Neural Networks". In: *Conference on Learning Theory* (COLT) (see page 50).

Golub, G. H. and H. A. Van der Vorst (2000). "Eigenvalue computation in the 20th century". *Journal of Computational and Applied Mathematics* (see pages 51, 52, 56, 176).

Goodfellow, I., J. Shlens, and C. Szegedy (2015). "Explaining and Harnessing Adversarial Examples". In: *International Conference on Learning Representations* (ICLR) (see pages 30, 31, 105, 140, 142, 156, 162).

Gouk, H., E. Frank, B. Pfahringer, and M. Cree (2018). "Regularisation of Neural Networks by Enforcing Lipschitz Continuity". *arXiv preprint arXiv:1804.04368* (see pages 51, 52, 56, 148).

Goyal, S., A. Roy Choudhury, and V. Sharma (2019). "Compression of Deep Neural Networks by Combining Pruning and Low Rank Decomposition". In: *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)* (see page 73).

Gray, R. (2006). "Toeplitz and Circulant Matrices: A Review". *Foundations and Trends® in Communications and Information Theory* (see pages 7, 88, 89, 173, 176).

Green, J. (1999). "Calculating the maximum modulus of a polynomial using Steckin's lemma". *SIAM journal on numerical analysis* (see page 99).

Grenander, U., G. Szegö, and M. Kac (1958). "Toeplitz Forms and Their Applications". *Physics Today* (see pages 14, 89, 112).

Guggenheimer, H. W., A. S. Edelman, and C. R. Johnson (1995). "A simple estimate of the condition number of a linear system". *The College Mathematics Journal* 26:1, pages 2–5 (see page 113).

Guo, C., M. Rana, M. Cisse, and L. van der Maaten (2018). "Countering Adversarial Images using Input Transformations". In: *International Conference on Learning Representations* (ICLR) (see page 140).

Gupta, S., A. Agrawal, K. Gopalakrishnan, and P. Narayanan (2015). "Deep Learning with Limited Numerical Precision". In: *Proceedings of the 32nd International Conference on Machine Learning* (ICML) (see page 38).

Gutiérrez Gutiérrez, J. and P. Crespo (2012). "Block Toeplitz matrices: Asymptotic results and applications". *Foundations and Trends® in Communications and Information Theory* (see pages 15, 89, 93).

Han, S., H. Mao, and W. J. Dally (2016). "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding". In: *International Conference on Learning Representations* (ICLR) (see page 39).

Hanin, B. (2017). "Universal Function Approximation by Deep Neural Nets with Bounded Width and ReLU Activations". *arXiv preprint arXiv:1708.02691* (see page 66).

Harvey, N., C. Liaw, and A. Mehrabian (2017). "Nearly-tight VC-dimension bounds for piecewise linear neural networks". In: *Conference on Learning Theory* (see page 32).

He, K., X. Zhang, S. Ren, and J. Sun (2015). "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *IEEE International Conference on Computer Vision* (ICCV) (see pages 2, 168).

He, K., X. Zhang, S. Ren, and J. Sun (2016). "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) (see pages 2, 3, 6, 29, 53, 104, 109, 150, 168, 169, 171).

Hein, M. and M. Andriushchenko (2017). "Formal guarantees on the robustness of a classifier against adversarial manipulation". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see page 46).

Hinrichs, A. and J. Vybíral (2011). "Johnson-Lindenstrauss Lemma for Circulant Matrices". *Random Structures & Algorithms* (see page 42).

Hinton, G., L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath et al. (2012). "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups". *IEEE Signal Processing Magazine* (see pages 2, 29, 168).

Hinton, G., O. Vinyals, and J. Dean (2015). "Distilling the Knowledge in a Neural Network". In: *Neural Information Processing Systems – Deep Learning and Representation Learning Workshop* (see pages 39, 81, 82).

Hinton, G. E. (1987). "Learning translation invariant recognition in a massively parallel networks". In: *Parallel Architectures and Languages Europe*. Springer Berlin Heidelberg (see page 27).

Hitchcock, F. L. (1927). "The expression of a tensor or a polyadic as a sum of products". *Journal of Mathematics and Physics* (see page 40).

Hornik, K., M. Stinchcombe, and H. White (1989). "Multilayer feedforward networks are universal approximators". *Neural Networks* (see page 66).

Hu, J., L. Shen, and G. Sun (2018). "Squeeze-and-excitation networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) (see pages 3, 169).

Huang, G., Z. Liu, L. Van Der Maaten, and K. Q. Weinberger (2017). "Densely Connected Convolutional Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) (see page 104).

Huang, L., L. Liu, F. Zhu, D. Wan, Z. Yuan, B. Li, and L. Shao (2020). "Controllable Orthogonalization in Training DNNs". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) (see page 50).

Huang, Y., Y. Cheng, A. Bapna, O. Firat, D. Chen, M. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu et al. (2019). "GPipe: Efficient Training of giant Neural Networks using Pipeline Parallelism". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see pages 3, 169).

Huhtanen, M. and A. Perämäki (2015). "Factoring Matrices into the Product of Circulant and Diagonal Matrices". *Journal of Fourier Analysis and Applications* (see pages 42, 43, 58, 59, 62, 112, 175).

Huster, T., C.-Y. J. Chiang, and R. Chadha (2018). "Limitations of the Lipschitz constant as a defense against adversarial examples". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer (see page 50).

Iandola, F. N., S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer (2016). "SqueezeNet: AlexNet-level Accuracy with 50x Fewer Parameters and< 0.5 MB Model Size". *arXiv preprint arXiv:1602.07360* (see page 104).

Jaderberg, M., A. Vedaldi, and A. Zisserman (2014). "Speeding up Convolutional Neural Networks with Low Rank Expansions". In: *Proceedings of the British Machine Vision Conference* (see page 40).

Jain, A. K. (1989). *Fundamentals of digital image processing*. Englewood Cliffs, NJ: Prentice Hall (see pages 17, 92).

Jégou, H., M. Douze, C. Schmid, and P. Pérez (2010). "Aggregating Local Descriptors into a Compact Image Representation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) (see page 126).

Jia, K., D. Tao, S. Gao, and X. Xu (2017). "Improving Training of Deep Neural Networks via Singular Value Bounding". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) (see page 53).

Jiang, Y.-G., Z. Wu, J. Wang, X. Xue, and S.-F. Chang (2018). "Exploiting Feature and Class Relationships in Video Categorization with Regularized Deep Neural Networks". *IEEE Transactions on Pattern Analysis and Machine Intelligence* (see page 126).

Jing, L., Y. Shen, T. Dubcek, J. Peurifoy, S. Skirlo, Y. LeCun, M. Tegmark, and M. Soljačić (2017). "Tunable efficient unitary neural networks (eunn) and their application to rnns". In: *Proceedings of the 34th International Conference on Machine Learning* (ICML) (see page 44).

Jordan, M. I. and R. A. Jacobs (1993). "Hierarchical Mixtures of Experts and the EM Algorithm". In: *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)* (see pages 83, 128).

Kachurovskii, R. I. (1969). "Regular points, spectrum and eigenfunctions of nonlinear operators". *Dokl. Akad. Nauk SSSR* 188:2, pages 274–277 (see page 114).

Kailath, T., S.-Y. Kung, and M. Morf (1979). "Displacement ranks of matrices and linear equations". *Journal of Mathematical Analysis and Applications* (see page 19).

Kaplan, J., S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei (2020). "Scaling Laws for Neural Language Models". *arXiv preprint arXiv:2001.08361* (see pages 2, 168).

Karpathy, A., G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei (2014). "Large-Scale Video Classification with Convolutional Neural Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) (see page 126).

Kautsky, J. and R. Turcajová (1994). "A Matrix Approach to Discrete Wavelets". In: *Wavelet Analysis and Its Applications*. Elsevier (see page 51).

Koltchinskii, V. and D. Panchenko (2000). "Rademacher Processes and Bounding the Risk of Function Learning". In: *High Dimensional Probability II* (see page 25).

Konečný, J., H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon (2016). "Federated Learning: Strategies for Improving Communication Efficiency". In: *Neural Information Processing Systems – Private Multi-Party Machine Learning Workshop* (see page 174).

Krizhevsky, A. and G. Hinton (2009). *Learning Multiple Layers of Features from Tiny Images*. Technical report. Citeseer (see page 150).

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see pages 1–3, 6, 104, 167–169, 171).

Kumar, A., A. Levine, T. Goldstein, and S. Feizi (2020). "Curse of Dimensionality on Randomized Smoothing for Certifiable Robustness". *arXiv preprint arXiv:2002.03239* (see page 165).

Kurakin, A., I. Goodfellow, and S. Bengio (2016). "Adversarial Examples in the Physical World". *arXiv preprint arXiv:1607.02533* (see pages 140, 162).

Langford, J. and J. Shawe-Taylor (2002). "PAC-Bayes & margins". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see page 32).

Latorre, F., P. Rolland, and V. Cevher (2020). "Lipschitz Constant Estimation for Neural Networks via Sparse Polynomial Optimization". In: *International Conference on Learning Representations* (ICLR) (see pages 47, 176).

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998). "Gradient-based learning applied to document recognition". *Proceedings of the IEEE* (see pages 2, 6, 82, 168, 171).

Lecuyer, M., V. Atlidakais, R. Geambasu, D. Hsu, and S. Jana (2018). "Certified Robustness to Adversarial Examples with Differential Privacy". In: *2019 IEEE Symposium on Security and Privacy (SP)* (see pages 140, 142, 144).

Lee, J., W. Reade, R. Sukthankar, G. Toderici et al. (2018). "The 2nd youtube-8m large-scale video understanding challenge". In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops* (see page 125).

Lehoucq, R. B. and D. C. Sorensen (1996). "Deflation techniques for an implicitly restarted Arnoldi iteration". *SIAM Journal on Matrix Analysis and Applications* (see page 101).

Li, B., C. Chen, W. Wang, and L. Carin (2018). "Second-Order Adversarial Attack and Certifiable Robustness". *arXiv preprint arXiv:1809.03113* (see page 142).

Li, B., T. Sainath, A. Narayanan, J. Caroselli, M. Bacchiani, A. Misra, I. Shafran, H. Sak, G. Pundak, K. Chin, K. C. Sim, R. J. Weiss, K. Wilson, E. Variani, C. Kim, O. Siohan, M. Weintraub, E. McDermott, R. Rose, and M. Shannon (2017a). "Acoustic Modeling for Google Home". In: *INTERSPEECH* (see pages 2, 169).

Li, C. and C. J. R. Shi (2018). "Constrained Optimization Based Low-Rank Approximation of Deep Neural Networks". In: *European Conference on Computer Vision*. Springer (see page 73).

Li, C. (2020). "OpenAI's GPT-3 Language Model: A Technical Overview" (see pages 3, 170).

Li, F., C. Gan, X. Liu, Y. Bian, X. Long, Y. Li, Z. Li, J. Zhou, and S. Wen (2017b). "Temporal Modeling Approaches for Large-scale Youtube-8M Video Understanding". *CoRR* arXiv preprint arXiv:1707.04555 (see page 126).

Li, Q., S. Haque, C. Anil, J. Lucas, R. B. Grosse, and J.-H. Jacobsen (2019). "Preventing Gradient Attenuation in Lipschitz Constrained Convolutional Networks". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see page 51).

Li, Y., H. Yang, E. R. Martin, K. L. Ho, and L. Ying (2015). "Butterfly factorization". *Multiscale Modeling & Simulation* (see page 44).

Lin, J., Y. Rao, J. Lu, and J. Zhou (2017). "Runtime Neural Pruning". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see page 39).

Liu, B., M. Wang, H. Foroosh, M. Tappen, and M. Penksy (2015). "Sparse Convolutional Neural Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) (see page 39).

Liu, W., D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg (2016). "SSD: Single Shot MultiBox Detector". In: *European Conference on Computer Vision*. Springer (see pages 2, 168).

Liu, X., M. Cheng, H. Zhang, and C.-J. Hsieh (2018). "Towards robust neural networks via random self-ensemble". In: *European Conference on Computer Vision*. Springer (see pages 140, 142).

Liu, Y., M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov (2019). "Roberta: A Robustly Optimized BERT Pretraining Approach". *arXiv preprint arXiv:1907.11692* (see pages 3, 169).

Lu, Z., H. Pu, F. Wang, Z. Hu, and L. Wang (2017). "The Expressive Power of Neural Networks: A View from the Width". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see page 66).

Ma, X., P. Zhang, S. Zhang, N. Duan, Y. Hou, M. Zhou, and D. Song (2019). "A Tensorized Transformer for Language Modeling". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see page 41).

Maas, A. L., A. Y. Hannun, and A. Y. Ng (2013). "Rectifier Non-Linearities Improve Neural Network Acoustic Models". In: *In ICML Workshop on Deep Learning for Audio, Speech and Language Processing* (see page 29).

Madry, A., A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu (2018). "Towards Deep Learning Models Resistant to Adversarial Attacks". In: *International Conference*

on Learning Representations (ICLR) (see pages 30, 31, 105, 109, 140–142, 150, 153, 162).

Malach, E., G. Yehudai, S. Shalev-Shwartz, and O. Shamir (2019). "Proving the Lottery Ticket Hypothesis: Pruning is All You Need". In: *Proceedings of the 36th International Conference on Machine Learning* (ICML) (see page 39).

Mallat, S. (2010). "Recursive interferometric representations". In: *2010 18th European Signal Processing Conference* (see page 84).

Mathieu, M. and Y. LeCun (2014). "Fast approximation of rotations and hessians matrices". *arXiv preprint arXiv:1404.7195* (see page 44).

McCulloch, W. S. and W. Pitts (1943). "A logical calculus of the ideas immanent in nervous activity". *The bulletin of mathematical biophysics* 5:4 (see pages 4, 26, 170).

Mellempudi, N., A. Kundu, D. Mudigere, D. Das, B. Kaul, and P. Dubey (2017). "Ternary Neural Networks with Fine-Grained Quantization". *CoRR* arXiv preprint arXiv:1705.01462 (see page 39).

Meng, D. and H. Chen (2017). "Magnet: a two-pronged defense against adversarial examples". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM (see pages 140, 142).

Merity, S., C. Xiong, J. Bradbury, and R. Socher (2016). "Pointer Sentinel Mixture Models". In: *International Conference on Learning Representations* (ICLR) (see pages 2, 168).

Micikevicius, P., S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu (2018). "Mixed Precision Training". In: *International Conference on Learning Representations* (ICLR) (see page 38).

Miech, A., I. Laptev, and J. Sivic (2017). "Learnable Pooling with Context Gating for Video Classification" (see pages 83, 84, 126–128).

Mises, R. and H. Pollaczek-Geiringer (1929). "Praktische Verfahren der Gleichungsauflösung." *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik* (see page 52).

Mitaim, S. and B. Kosko (1998). "Adaptive stochastic resonance". *Proceedings of the IEEE* (see page 141).

Miyato, T., T. Kataoka, M. Koyama, and Y. Yoshida (2018). "Spectral Normalization for Generative Adversarial Networks". In: *International Conference on Learning Representations* (ICLR) (see pages 52, 56, 175).

Moczulski, M., M. Denil, J. Appleyard, and N. de Freitas (2016). "ACDC: A Structured Efficient Linear Layer". In: *International Conference on Learning Representations* (ICLR) (see pages 6, 42, 43, 45, 58, 78, 79, 172, 174).

Moosavi-Dezfooli, S.-M., A. Fawzi, O. Fawzi, and P. Frossard (2017). "Universal Adversarial Perturbations". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) (see page 140).

Moosavi-Dezfooli, S.-M., A. Fawzi, and P. Frossard (2016). "Deepfool: A Simple and Accurate Method to Fool Deep Neural Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) (see page 140).

Müller-Quade, J., H. Aagedal, T. Beth, and M. Schmid (1998). "Algorithmic Design of Diffractive Optical Systems for Information Processing". *Physica D: Nonlinear Phenomena* (see page 175).

Munkhoeva, M., Y. Kapushev, E. Burnaev, and I. Oseledets (2018). "Quadrature-based features for kernel approximation". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see page 44).

Nair, V. and G. E. Hinton (2010). "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *Proceedings of the 27th International Conference on Machine Learning* (ICML) (see page 28).

Neyshabur, B. (2017). "Implicit Regularization in Deep Learning". PhD Thesis. Toyota Technological Institute at Chicago (see page 50).

Neyshabur, B., S. Bhojanapalli, and N. Srebro (2018). "A PAC-Bayesian Approach to Spectrally-Normalized Margin Bounds for Neural Networks". In: *International Conference on Learning Representations* (ICLR) (see page 32).

Novikov, A., D. Podoprikhin, A. Osokin, and D. P. Vetrov (2015). "Tensorizing neural networks". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see page 41).

Oseledets, I. V. (2011). "Tensor-train decomposition". *SIAM Journal on Scientific Computing* (see pages 40, 41).

Pan, V. Y. and X. Wang (2003). "Inversion of displacement operators". *SIAM Journal on Matrix Analysis and Applications* (see page 21).

Pan, V. Y. (2001). *Structured Matrices and Polynomials: Unified Superfast Algorithms*. Springer-Verlag (see pages 20, 21, 44).

Pan, Y., J. Xu, M. Wang, J. Ye, F. Wang, K. Bai, and Z. Xu (2019). "Compressing recurrent neural networks with tensor ring for action recognition". In: *Proceedings of the 33rd AAAI Conference on Artificial Intelligence* (see page 41).

Papernot, N., P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami (2016a). "The limitations of deep learning in adversarial settings". In: *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE (see pages 140, 162).

Papernot, N., P. McDaniel, X. Wu, S. Jha, and A. Swami (2016b). "Distillation as a defense to adversarial perturbations against deep neural networks". In: *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE (see page 140).

Parker, D. S. (1995). "Random butterfly transformations with applications in computational linear algebra" (see page 44).

Parter, S. V. (1961). "Extreme eigenvalues of Toeplitz forms and applications to elliptic difference equations". *Transactions of the American Mathematical Society* (see page 14).

Perez, L. and J. Wang (2017). "The Effectiveness of Data Augmentation in Image Classification using Deep Learning". *arXiv preprint arXiv:1712.04621* (see page 141).

Perronnin, F. and C. Dance (2007). "Fisher Kernels on Visual Vocabularies for Image Categorization". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) (see page 126).

Peters, M. E., M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer (2018). "Deep Contextualized Word Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (NAACL) (see pages 3, 169).

Pfister, L. and Y. Bresler (2018). "Bounding Multivariate Trigonometric Polynomials with Applications to Filter Bank Design". *arXiv preprint arXiv:1802.09588* (see page 99).

Pinot, R., R. Ettedgui, G. Rizk, Y. Chevaleyre, and J. Atif (2020). "Randomization Matters. How to Defend Against Strong Adversarial Attacks". In: *Proceedings of the 37th International Conference on Machine Learning* (ICML) (see page 165).

Radford, A., K. Narasimhan, T. Salimans, and I. Sutskever (2018). "Improving language understanding by Generative Pre-Training" (see pages 3, 169).

Radford, A., J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever (2019). "Language Models are Unsupervised Multitask Learners". *OpenAI Blog* (see pages 2, 3, 29, 168, 169).

Raffel, C., N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu (2020). "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". *Journal of Machine Learning Research* (see pages 2, 3, 168, 169).

Rakin, A. S., Z. He, and D. Fan (2018). "Parametric Noise Injection: Trainable Randomness to Improve Deep Neural Network Robustness against Adversarial Attack". *arXiv preprint arXiv:1811.09310* (see pages 140, 142, 144).

Rastegari, M., V. Ordonez, J. Redmon, and A. Farhadi (2016). "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks". In: *European Conference on Computer Vision*. Springer (see page 39).

Real, E., A. Aggarwal, Y. Huang, and Q. V. Le (2019). "Regularized Evolution for Image Classifier Architecture Search". In: *Proceedings of the 33rd AAAI Conference on Artificial Intelligence* (see pages 3, 39, 169).

Redmon, J., S. Divvala, R. Girshick, and A. Farhadi (2016). "You Only Look Once: Unified, Real-Time Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) (see pages 2, 168).

Redmon, J. and A. Farhadi (2017). "YOLO9000: Better, Faster, Stronger". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) (see pages 2, 168).

Rényi, A. (1961). *On measures of entropy and information*. Technical report. Hungarian Academy of Sciences Budapest Hungary (see page 146).

Rice, L., E. Wong, and Z. Kolter (2020). "Overfitting in adversarially robust deep learning". In: *Proceedings of the 37th International Conference on Machine Learning* (ICML) (see page 7).

Rosenblatt, F. (1958). "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review* (see pages 4, 26, 170).

Rosenfeld, J. S., A. Rosenfeld, Y. Belinkov, and N. Shavit (2020). "A Constructive Prediction of the Generalization Error Across Scales". In: *International Conference on Learning Representations* (ICLR) (see pages 2, 168).

Rosset, C. (2020). "Turing-NLG: A 17-billion-parameter Language Model". *Microsoft Blog* (see pages 3, 169).

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). "Learning representations by back-propagating errors". *Nature* (see page 27).

Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei (2015). "ImageNet Large Scale Visual Recognition Challenge". *International Journal of Computer Vision* (IJCV) (see pages 1, 167).

Ryu, E., J. Liu, S. Wang, X. Chen, Z. Wang, and W. Yin (2019). "Plug-and-Play Methods Provably Converge with Properly Trained Denoisers". In: *Proceedings of the 36th International Conference on Machine Learning* (ICML) (see page 53).

Sadat, A., M. Ren, A. Pokrovsky, Y.-C. Lin, E. Yumer, and R. Urtasun (2019). "Jointly Learnable Behavior and Trajectory Planning for Self-Driving Vehicles" (see page 2).

Sainath, T. N., B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran (2013). "Low-Rank Matrix Factorization for Deep Neural Network Training with High-Dimensional Output Targets". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (see page 40).

Sainath, T. N. and C. Parada (2015). "Convolutional neural networks for small-footprint keyword spotting". In: *Sixteenth Annual Conference of the International Speech Communication Association* (see page 174).

Salman, H., J. Li, I. Razenshteyn, P. Zhang, H. Zhang, S. Bubeck, and G. Yang (2019). "Provably robust deep learning via adversarially trained smoothed classifiers". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see pages 156, 164).

Samangouei, P., M. Kabkab, and R. Chellappa (2018). "Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models". In: *International Conference on Learning Representations* (ICLR) (see pages 140, 142).

Sanchez, V., P. Garcia, A. M. Peinado, J. C. Segura, and A. J. Rubio (1995). "Diagonalizing Properties of the Discrete Cosine Transforms". *IEEE Transactions on Signal Processing* (see page 43).

Saxe, A., J. L. McClelland, and S. Ganguli (2014). "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks". In: *International Conference on Learning Representations* (ICLR) (see page 113).

Saxe, A. M., J. L. McClelland, and S. Ganguli (2013). "Exact Solutions to the Nonlinear Dynamics of Learning in Deep Linear Neural Networks". *arXiv preprint arXiv:1312.6120* (see page 73).

Schmid, M., R. Steinwandt, J. Müller-Quade, M. Rötteler, and T. Beth (2000). "Decomposing a Matrix into Circulant and Diagonal Factors". *Linear Algebra and its Applications* 306 (see page 59).

Schmidt, L., S. Santurkar, D. Tsipras, K. Talwar, and A. Madry (2018). "Adversarially robust generalization requires more data". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see pages 7, 88, 104, 173, 176).

Sedghi, H., V. Gupta, and P. Long (2018). "The Singular Values of Convolutional Layers". In: *International Conference on Learning Representations* (ICLR) (see pages 53–56, 73, 102, 103).

Serra, S. (1994). "Preconditioning strategies for asymptotically ill-conditioned block Toeplitz systems". *BIT Numerical Mathematics* 34:4 (see page 93).

Serra, S. (1997). "The extension of the concept of the generating function to a class of preconditioned Toeplitz matrices". *Linear algebra and its applications* (see page 14).

Shalev-Shwartz, S. and S. Ben-David (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press (see pages 24, 26).

Shannon, C. E. (1949). "Communication theory of secrecy systems". *The Bell system technical journal* (see page 30).

Sharma, A. and Z. Zheng (2021). In: *Automating Cities: Design, Construction, Operation and Future Impact*. Editor B. T. Wang and C. M. Wang. Springer Singapore, Singapore, pages 273–296 (see page 169).

Shoeybi, M., M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro (2019). "Megatron-lm: Training Multi-Billion Parameter Language Models using GPU Model Parallelism". *arXiv preprint arXiv:1909.08053* (see pages 3, 169).

Silver, D., J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton et al. (2017). "Mastering the game of go without human knowledge". *nature* 550:7676, pages 354–359 (see page 2).

Simon-Gabriel, C.-J., Y. Ollivier, B. Schölkopf, L. Bottou, and D. Lopez-Paz (2018). "Adversarial Vulnerability of Neural Networks Increases With Input Dimension". *arXiv preprint arXiv:1802.01421* (see page 140).

Simonyan, K. and A. Zisserman (2014). "Very Deep Convolutional Networks for Large-Scale Image Recognition". *arXiv preprint arXiv:1409.1556* (see pages 3, 41, 45, 104, 169).

Sindhwani, V., T. Sainath, and S. Kumar (2015). "Structured transforms for Small-Footprint Deep Learning". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see pages 6, 21, 43–45, 172).

Singla, S. and S. Feizi (2019). "Bounding Singular Values of Convolution Layers". *arXiv preprint arXiv:1911.10258* (see pages 53, 55, 56, 102, 103).

Skalic, M., M. Pekalski, and X. E. Pan (2017). "Deep Learning Methods for Efficient Large Scale Video Labeling". *arXiv preprint arXiv:1706.04572* (see page 130).

Spigler, S., M. Geiger, S. d'Ascoli, L. Sagun, G. Biroli, and M. Wyart (2019). "A jamming transition from under- to over-parametrization affects generalization in deep learning". *Journal of Physics A: Mathematical and Theoretical* (see page 5).

Strubell, E., A. Ganesh, and A. McCallum (2019). "Energy and Policy Considerations for Deep Learning in NLP". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (see pages 3, 170).

Szegedy, C., S. Ioffe, V. Vanhoucke, and A. A. Alemi (2017). "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". In: *Proceedings of the 31st AAAI Conference on Artificial Intelligence* (see pages 3, 169).

Szegedy, C., W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus (2014). "Intriguing properties of neural networks". In: *International Conference on Learning Representations* (ICLR) (see pages 4, 29, 45, 140, 143, 162, 170).

Szegö, G. (1915). "Ein Grenzwertsatz über die Toeplitzschen Determinanten einer reellen positiven Funktion". *Mathematische Annalen* (see pages 14, 113).

Tan, M. and Q. Le (2019). "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *Proceedings of the 36th International Conference on Machine Learning* (ICML) (see pages 2, 6, 38, 39, 53, 112, 168, 171).

Thomas, A., A. Gu, T. Dao, A. Rudra, and C. Ré (2018). "Learning Compressed Transforms with Low Displacement Rank". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see pages 44, 45, 81).

Tilli, P. (1997). "On the asymptotic spectrum of Hermitian block Toeplitz matrices with Toeplitz blocks". *Mathematics of computation* (see page 14).

Tilli, P. (1998). "Singular values and eigenvalues of non-Hermitian block Toeplitz matrices". *Linear algebra and its applications* (see page 14).

Tjandra, A., S. Sakti, and S. Nakamura (2017). "Compressing recurrent neural network with tensor train". In: *International Joint Conference on Neural Networks* (IJCNN) (see page 41).

Trabelsi, C., O. Bilaniuk, Y. Zhang, D. Serdyuk, S. Subramanian, J. F. Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, and C. J. Pal (2018). "Deep Complex Networks". In: *International Conference on Learning Representations* (ICLR) (see page 63).

Tramer, F., N. Carlini, W. Brendel, and A. Madry (2020). "On Adaptive Attacks to Adversarial Example Defenses". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see page 109).

Tramèr, F. and D. Boneh (2019). "Adversarial training and robustness for multiple perturbations". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see pages 162, 163).

Tsuzuku, Y., I. Sato, and M. Sugiyama (2018). "Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see pages 45, 46, 56, 105, 175).

Tucker, L. R. (1966). "Some mathematical notes on three-mode factor analysis". *Psychometrika* 31:3, pages 279–311 (see page 40).

Turing, A. (1950). "Computing Machinery and Intelligence". *Mind* 59, pages 433–460 (see page 1).

Tyrtyshnikov, E. and N. Zamarashkin (1998). "Spectra of multilevel Toeplitz matrices: advanced theory via simple matrix relationships". *Linear algebra and its applications* (see page 14).

Vapnik, V. N. and A. Y. Chervonenkis (2015). "On the uniform convergence of relative frequencies of events to their probabilities". In: *Measures of complexity*. Springer, pages 11–30 (see page 24).

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin (2017). "Attention is all you need". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see pages 2, 3, 112, 168, 170).

Villani, C. (2008). *Optimal transport: old and new*. Volume 338. Springer Science & Business Media (see page 146).

Virmaux, A. and K. Scaman (2018). "Lipschitz regularity of deep neural networks: analysis and efficient estimation". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see pages 46, 47, 49, 175, 176).

Vybíral, J. (2011). "A Variant of the Johnson–Lindenstrauss Lemma for Circulant Matrices". *Journal of Functional Analysis* (see page 42).

Wang, J., Y. Chen, R. Chakraborty, and S. X. Yu (2020). "Orthogonal Convolutional Neural Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) (see page 50).

Widom, H. (1965). *Toeplitz Matrices*. Editor I. Hirschman. Studies in Mathematics. Prentice Hall (see page 14).

Widom, H. (1976). "Asymptotic behavior of block Toeplitz matrices and determinants. II". *Advances in Mathematics* (see pages 93, 119).

Wu, Y., M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey et al. (2016). "Google's Neural Machine Translation System: Bridging the Gap Between Human and Machine Translation". *arXiv preprint arXiv:1609.08144* (see page 169).

Xiao, L., Y. Bahri, J. Sohl-Dickstein, S. Schoenholz, and J. Pennington (2018). "Dynamical Isometry and a Mean Field Theory of CNNs: How to Train 10,000-Layer Vanilla Convolutional Neural Networks". In: *Proceedings of the 35th International Conference on Machine Learning* (ICML) (see page 51).

Xie, C., J. Wang, Z. Zhang, Z. Ren, and A. Yuille (2018). "Mitigating Adversarial Effects Through Randomization". In: *International Conference on Learning Representations* (ICLR) (see pages 140, 141).

Xie, S., R. Girshick, P. Dollár, Z. Tu, and K. He (2017). "Aggregated residual transformations for deep neural networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) (see pages 3, 169).

Yang, Y., D. Krompass, and V. Tresp (2017). "Tensor-train recurrent neural networks for video classification". In: *Proceedings of the 34th International Conference on Machine Learning* (ICML) (see page 41).

Yang, Z., Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le (2019). "Xlnet: Generalized autoregressive pretraining for language understanding". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see pages 3, 169).

Yoshida, Y. and T. Miyato (2017). "Spectral Norm Regularization for Improving the Generalizability of Deep Learning". *arXiv preprint arXiv:1705.10941* (see pages 52, 56, 104).

Yu, D. and L. Deng (2016). *Automatic Speech Recognition*. Springer (see pages 2, 168).

Yu, F., S. Kumar, Y. Gong, and S.-F. Chang (2014). "Circulant Binary Embedding". In: *Proceedings of the 31st International Conference on Machine Learning* (ICML) (see page 42).

Yu, F. X., S. Kumar, H. Rowley, and S.-F. Chang (2015). "Compact Nonlinear Maps and Circulant Extensions". *arXiv preprint arXiv:1503.03893* (see page 42).

Yu, X., T. Liu, X. Wang, and D. Tao (2017). "On Compressing Deep Models by Low Rank and Sparse Decomposition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) (see pages 40, 79).

Yue-Hei Ng, J., M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici (2015). "Beyond Short Snippets: Deep Networks for Video Classification". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) (see page 126).

Zagoruyko, S. and N. Komodakis (2016). "Wide Residual Networks". In: *Proceedings of the British Machine Vision Conference* (BMVC) (see pages 104, 108, 150, 163).

Zhang, C., S. Bengio, M. Hardt, B. Recht, and O. Vinyals (2017). "Understanding Deep Learning Requires Rethinking Generalization". In: *International Conference on Learning Representations* (ICLR) (see page 32).

Zhang, F. (2011). *Matrix theory: basic results and techniques*. Springer Science & Business Media (see page 93).

Zhang, H., Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan (2019). "Theoretically Principled Trade-Off Between Robustness and Accuracy". *arXiv preprint arXiv:1901.08573* (see page 113).

Zhao, L., S. Liao, Y. Wang, Z. Li, J. Tang, and B. Yuan (2017). "Theoretical Properties for Neural Networks with Weight Matrices of Low Displacement Rank". In: *Proceedings of the 34th International Conference on Machine Learning* (ICML) (see page 62).

Zhou, H., J. Lan, R. Liu, and J. Yosinski (2019). "Deconstructing lottery tickets: Zeros, signs, and the supermask". In: *Advances in Neural Information Processing Systems* (NeurIPS) (see page 39).

Zoph, B., V. Vasudevan, J. Shlens, and Q. V. Le (2018). "Learning transferable architectures for scalable image recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) (see page 39).

Zozor, S. and P.-O. Amblard (1999). "Stochastic resonance in discrete time nonlinear AR(1) models". *IEEE transactions on Signal Processing* (see page 141).

ABSTRACT

Deep neural networks are state-of-the-art in a wide variety of tasks, however, they exhibit important limitations which hinder their use and deployment in real-world applications. When developing and training neural networks, the accuracy should not be the only concern, neural networks must also be cost-effective and reliable. Although accurate, large neural networks often lack these properties. This thesis focuses on the problem of training neural networks which are not only accurate but also compact, easy to train, reliable and robust to adversarial examples. To tackle these problems, we leverage the properties of structured matrices from the Toeplitz family to build compact and secure neural networks.

KEYWORDS

Deep Learning, Neural networks, Structured Matrices

RÉSUMÉ

Les réseaux de neurones profonds sont considérés comme étant état de l'art dans une grande variété de tâches, mais ils présentent des limites importantes qui entravent leur utilisation et leur déploiement. Lors du développement et l'entraînement de réseaux de neurones, la précision ne devrait pas être la seule préoccupation, ils se doivent aussi d'être efficaces et sécurisés. Bien que précis, les réseaux de neurones dotés de nombreux paramètres n'ont souvent pas ces propriétés. Cette thèse se concentre sur le problème de l'entraînement de réseaux de neurones qui ne sont pas seulement précis, mais aussi compacts, faciles à entraîner, fiables et robustes aux exemples contradictoires. Pour résoudre ces problèmes, nous exploitons les propriétés des matrices structurées de la famille de Toeplitz pour construire des réseaux de neurones compacts et sécurisés.

MOTS CLÉS

Apprentissage Profond, Réseaux de Neurones, Matrices Structurées