Robust Deep learning methods inspired by signal processing algorithms

Ana-Antonia Neacșu

Robust Deep Learning Methods Inspired from Signal Processing Algorithms

Metode robuste de învățare profundă inspirate din algoritmi de procesare de semnal

Méthodes robustes d'apprentissage profond inspirées d'algorithmes de traitement du signal

Thèse de doctorat de l'université Paris-Saclay et de l'Université Nationale des Sciences et Technologies Politehnica Bucarest

École doctorale n° 580 Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat : Génie Informatique, Automatique et Traitement du Signal
Graduate School : Informatique et sciences du numérique
Référent : CentraleSupelec

Thèse préparée à la unité de recherche Centre de Vision Numérique (Université Paris-Saclay, CentraleSupélec) sous la direction de Jean-Christophe PESQUET et

à l'Université Nationale des Sciences et Technologies Politehnica Bucarest

(UNSTPB) sous la direction de Corneliu BURILEANU

Thèse présentée et soutenue à Bucarest, le 4 Décembre 2023, par

Ana-Antonia Neacșu

Composition du Jury

membres du jury avec voix délibérative

Mihai Ciuc Professor,National University of Science and Technology Politehnica Bucharest (UNSTPB), Romania	Président
Nicu Sebe Professor, University of Trento, Italy	Rapporteur & Examinateur
Corneliu Rusu Professor, Technical University of Cluj-Napoca (UTCN), Romania	Rapporteur & Examinateur
Daniela Tărniceriu Professor, Technical University "Gh. Asachi" Iași (TUIASI), Romania	Examinatrice
Jean-Philippe Ovarlez Research Director at ONERA, Université Paris-Saclay, France	Examinateur
Franck Mamalet Senior Expert in Artificial Intelligence at IRT Saint Exupéry, Toulouse, France	Examinateur

Thèse de doctorat
NNT : 2023UPAST212

Trebuie să urci cât mai sus pentru a vedea cât mai departe.

C. Brâncuși

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Title: Robust deep learning methods inspired by signal processing algorithms
Keywords: robust optimization, Lipschitz stability, neural networks

Abstract: Understanding the importance of defense strategies against adversarial attacks has become paramount in ensuring the trustworthiness and resilience of neural networks. While traditional security measures focused on protecting data and software from external threats, the unique challenge posed by adversarial attacks lies in their ability to exploit the inherent vulnerabilities of the underlying machine learning algorithms themselves.

The first part of the thesis proposes new constrained learning strategies that ensure robustness against adversarial perturbations by controlling the Lipschitz constant of a classifier. We focus on nonnegative neural networks for which accurate Lipschitz bounds can be derived, and we propose different spectral norm constraints offering robustness guarantees from a theoretical viewpoint. We validate our solution in the context of gesture recognition based on Surface Electromyographic (sEMG) signals.

In the second part of the thesis, we propose a new class of neural networks (ACNN) which can be viewed as establishing a link between fully connected and convolutional networks, and we propose an iterative algorithm to control their robustness during training. Next, we extend our solution to the complex plane and address the problem of designing robust complex-valued neural networks by proposing a new architecture (RCFF-Net) for which we derive tight Lipschitz constant bounds. Both solutions are validated for audio denoising.

In the last part, we introduce ABBA Networks, a novel class of (almost) non-negative neural networks, which we show to be universal approximators. We derive tight Lipschitz bounds for both linear and convolutional layers, and we propose an algorithm to train robust ABBA networks. We show the effectiveness of the proposed approach in the context of image classification.

Titre: Méthodes robustes d'apprentissage profond inspirées d'algorithmes de traitement du signal
Mots clés: Optimisation robuste, stabilité Lipschitz, réseaux de neurones

Résumé: Comprendre l'importance des stratégies de défense contre les attaques adverses est devenu primordial pour garantir la fiabilité et la résilience des réseaux de neurones. Alors que les mesures de sécurité traditionnelles se focalisent sur la protection des données et des logiciels contre les menaces externes, le défi unique posé par les attaques adverses réside dans leur capacité à exploiter les vulnérabilités inhérentes aux algorithmes d'apprentissage automatique.

Dans la première partie de la thèse, nous proposons de nouvelles stratégies d'apprentissage contraint qui garantissent la robustesse vis-à-vis des perturbations adverses, en contrôlant la constante de Lipschitz d'un classifeur. Nous concentrons notre attention sur les réseaux de neurones positifs pour lesquels des bornes de Lipschitz précises peuvent être déduites, et nous proposons différentes contraintes de norme spectrale offrant des garanties de robustesse, d'un point de vue théorique. Nous validons notre solution dans le contexte de la reconnaissance de gestes basée sur des signaux électromyographiques de surface (sEMG).

Dans la deuxième partie de la thèse, nous proposons une nouvelle classe de réseaux de neurones (ACNN) qui peut être considérée comme un intermédiaire entre les réseaux entièrement connectés et ceux convolutionnels. Nous proposons un algorithme itératif pour contrôler la robustesse pendant l'apprentissage. Ensuite, nous étendons notre solution au plan complexe et abordons le problème de la conception de réseaux de neurones robustes à valeurs complexes, en proposant une nouvelle architecture (RCFF-Net) pour laquelle nous obtenons des bornes fines de la constante de Lipschitz. Les deux solutions sont validées en débruitage audio.

Dans la dernière partie, nous introduisons les réseaux ABBA, une nouvelle classe de réseaux de neurones (presque) positifs, dont nous démontrons les propriétés d'approximation universelle. Nous déduisons des bornes fines de Lipschitz pour les couches linéaires ou convolutionnelles, et nous proposons un algorithme pour entraîner des réseaux ABBA robustes. Nous démontrons l'efficacité de l'approche proposée dans le contexte de la classification d'images.

Acknolegements

First, I want to thank the members of the jury for their invaluable time and effort in meticulously examining my work.

I would like to extend my deepest gratitude to my two advisors, Professor Corneliu Burileanu and Professor Jean-Christophe Pesquet, for their unwavering guidance and support throughout my doctoral journey. Professor Corneliu Burileanu, your mentorship over the last nine years has been truly transformative. Your vision, wisdom, and dedication have not only shaped my academic and research endeavors but have also played an instrumental role in shaping the person I have become today. I am profoundly grateful for your belief in my potential and the opportunities you have provided.

Professor Jean-Christophe Pesquet, your mentorship in the art of conducting top-tier research and writing good papers has been invaluable. Your guidance has not only refined my research skills but has also instilled in me a deep appreciation for the pursuit of excellence in academia. I am thankful for your unwavering support, your insightful feedback on my work, and the countless lessons I have learned under your guidance.

Next, I would like to express my heartfelt appreciation to my colleagues from Campus: Cristina, Marian, Vlad, Cristi, Andrei, Ramona, Dragoș, and all the others whose camaraderie has made this academic journey not only inspiring but also very enjoyable. Working alongside such talented individuals has been a privilege, and I extend my special thanks to Vlad for his invaluable assistance in a crucial part of my PhD work. Vlad, your expertise and dedication have been instrumental in the success of this research, and I am truly grateful for your collaboration and for our friendship. My appreciation also goes to Simona, whose mental support and encouragement have been a source of strength throughout this journey. Your positive presence has made the challenges of pursuing a PhD more manageable, and I am deeply thankful for your friendship. To all my colleagues, I appreciate your patience and understanding, especially when my response to your requests was often, "After I finish my PhD." Your support and understanding have been a tremendous source of motivation.

A great thank you goes to my parents, Luminița and Mihai, and to my sister Carla, your unending love has been the pillar of strength that sustained me through the challenges of pursuing a PhD. I also want to offer a special thanks to my grandmother Maria, my number one fan, whose prayers and blessings have been a constant source of inspiration. Your faith in me has been a guiding light. To my

family, your encouragement has propelled me forward, and your love has been my constant refuge, I am very lucky to have you close.

To my dearest Georgian, I appreciate your love and support throughout this incredibly challenging journey. Your belief in me and our shared dreams have been the driving force behind my determination, and I am very grateful for your commitment to us and to our success. Thank you for being my pillar, my confidant, and my source of endless love and strength.

Abstract

In the ever-evolving landscape of artificial intelligence, neural networks have emerged as a formidable tool for solving complex tasks, ranging from image recognition and natural language processing to autonomous driving and medical diagnostics. These powerful approaches have revolutionized various industrial areas, providing unprecedented levels of accuracy and efficiency. However, with their growing significance and prevalence, they have also become a prime target for malicious actors seeking to exploit their vulnerabilities.

Adversarial attacks, a class of techniques designed to deceive and manipulate neural networks, pose a significant threat to the integrity and reliability of AI systems. These attacks involve the intentional introduction of imperceptible perturbations into input data, causing neural networks to misclassify or produce erroneous outputs. By exploiting these weaknesses, adversaries can undermine the functionality of AI models, leading to potentially catastrophic consequences in critical applications such as cybersecurity, autonomous vehicles, and healthcare. Adversarial noise can also be the result of non-intended actions.

To safeguard the reliability of neural networks, researchers and engineers have delved into developing robust defense mechanisms that can withstand and mitigate the impact of adversarial attacks. This thesis contributes to the development of a new and efficient defense mechanism to guarantee the robustness of feed-forward neural networks against adversarial attacks.

Chapter 3 shows a novel method for constructing a robust Automatic Gesture Recognition system using forearm-level Surface Electromyographic (sEMG) signals. By modulating the Lipschitz constant of the classifier, we propose new constrained learning strategies that ensure robustness against adversarial perturbations. We concentrate on nonnegative neural networks for which precise Lipschitz bounds can be derived, and we propose various spectral norm constraints that provide theoretical guarantees of robustness. The experimental results on publicly accessible datasets demonstrate that a satisfactory balance between accuracy and stability is obtained. Moreover, the robustness of our proposed solution is supported by a real-life scenario experiment.

The second contribution of this thesis, described in Chapter 4, introduces two new neural network architectures. The first one (ACNN) generalizes the concept of the convolutional kernel by imposing a certain band-structure on linear layers, while the second one (RCFF-Net) is inspired by CapsuleNets and operates in the complex space. For both networks, we derive tight Lipschitz bounds. In the complex case,

our analysis is grounded on new theoretical results on Lipschitzian nonlinear systems. We propose constrained training algorithms to control the robustness of the models. We experiment on a regression task, namely denoising audio clips corrupted by Gaussian noise, showing that our strategy to ensure the stability of neural networks is not limited to classification problems.

Finally, in Chapter 5, starting from linear algebra considerations, we present ABBA neural networks, a novel class of (almost) non-negative neural networks offering a panel of desirable properties. In particular, we show that these networks are universal approximators while retaining the benefits of non-negative weighted networks. In the fully connected and convolutional cases, we derive tight Lipschitz bounds. By precisely controlling the Lipschitz constant of the network during the training phase, we propose a method for designing ABBA networks that are resilient to adversarial perturbations. We demonstrate that our method outperforms other state-of-the-art white-box adversary defenses. On four benchmark datasets, image classification experiments are conducted.

Résumé

Dans le paysage en constante évolution de l'intelligence artificielle, les réseaux de neurones apparaissent comme un outil formidable pour résoudre des tâches complexes, allant de la reconnaissance d'images et du traitement du langage naturel à la conduite autonome et au diagnostic médical. Ces puissantes approches ont révolutionné divers secteurs d'activité en offrant des niveaux de précision et d'efficacité sans précédent. Cependant, en raison de leur importance croissante et de leur prévalence, ils sont également devenus une cible de choix pour des entités malveillantes cherchant à exploiter leurs vulnérabilités.

Les attaques adverses, une catégorie de techniques conçues pour tromper et manipuler les réseaux de neurones, posent une menace certaine pour l'intégrité et la fiabilité des systèmes d'IA. Ces attaques consistent à introduire intentionnellement des perturbations imperceptibles dans les données d'entrée, ce qui entraîne une mauvaise classification ou des sorties erronées des réseaux de neurones. En exploitant ces faiblesses, les adversaires peuvent compromettre le fonctionnement des modèles d'IA, entraînant des conséquences potentiellement catastrophiques dans des applications critiques telles que la cybersécurité, les véhicules autonomes et le domaine de la santé. Les bruits adverses peuvent aussi advenir de manière accidentelle.

Pour garantir la fiabilité des réseaux de neurones, les chercheurs et ingénieurs se sont penchés sur le développement de mécanismes de défense robustes capables de résister ou d'atténuer l'impact des attaques adverses. Cette thèse contribue au développement de nouveaux mécanismes de défense efficaces pour garantir la robustesse des réseaux de neurones face aux attaques adverses.

Le Chapitre 3 présente une nouvelle méthode pour construire un système robuste de reconnaissance automatique de gestes en utilisant des signaux électromyographiques de surface (sEMG) au niveau de l'avant-bras. En modulant la constante de Lipschitz du classifieur, nous proposons de nouvelles stratégies d'apprentissage contraint qui garantissent la robustesse vis-à-vis des perturbations adverses. Nous nous focalisons sur les réseaux de neurones positifs pour lesquels des bornes de Lipschitz précises peuvent être déduites, et nous proposons différentes contraintes de norme spectrale offrant des garanties théoriques de robustesse. Les résultats expérimentaux sur des ensembles de données publiquement accessibles aboutissent à un compromis satisfaisant entre précision et stabilité. De plus, la robustesse de la solution proposée est étayée par une expérimentation en situation réelle.

La deuxième contribution de cette thèse, exposée dans le Chapitre 4, présente deux nouvelles architectures de réseaux de neurones. La première (ACNN) généralise le concept de noyau convolutif en imposant une certaine structure bande aux couches linéaires, tandis que la seconde (RCFF-Net) est inspirée par les CapsuleNets et opère dans le domaine complexe. Pour les deux réseaux, nous calculons des bornes de Lipschitz fines. Dans le cas complexe, notre analyse est basée sur de nouveaux résultats théoriques concernant les systèmes non linéaires lipschitzien. Nous proposons des algorithmes d'apprentissage contraint pour contrôler la robustesse des modèles. Des expériences sont menées sur une tâche de régression, à savoir le débruitage de clips audio corrompus par un bruit gaussien, démontrant ainsi que notre stratégie de stabilisation des réseaux de neurones n'est pas limitée aux problèmes de classification.

Enfin, dans le Chapitre 5, partant de considérations d'algèbre linéaire, nous présentons les réseaux ABBA, une nouvelle classe de réseaux de neurones (presque) positifs vérifiant un ensemble de propriétés désirables. En particulier, nous montrons que ces réseaux sont des approximateurs universels préservant les avantages des réseaux à poids positifs. Dans les cas entièrement connecté et convolutionnel, nous obtenons des bornes de Lipschitz fines. En contrôlant précisément la constante de Lipschitz du réseau pendant la phase d'apprentissage, nous proposons une méthode pour concevoir des réseaux ABBA résistant aux attaques adverses. Nous démontrons que notre méthode surpasse d'autres défenses de l'état de l'art, contrant les perturbations adversaires de type "boîte blanche". Des expériences de classification d'images sont menées sur quatre ensembles de données de référence.

Contents

Acronyms

ACNN	: Adaptive Convolutive Neural Network
AE	: Autoencoders
AGR	: Automatic Gesture Recognition
AI	: Artificial Intelligence
ALMA	: Augumented Lagrangian Method for Adversarial
AT	: Adversarial Training
CC	: Cross-Correlation
CDL	: Complex Dense Layer
CNN	: Convolutional Neural Network
CVNN	: Complex-Valued Neural Network
C&W	: Carlini and Wagner
DDN	: Decoupling Direction Norm
Diag	: Diagonal Layer
DFB	: Dual Forward-Backward
DNN	: Deep Neural Network
FCN	: Fully Connected Network
FGSM	: Fast-Gradient Sign Method
FISTA	: Fast Iterative Shrinkage-Thresholding Algorithm
FMN	: Fast Minimum Norm
GAN	: Generative Adversial Network
GloRo-Net	: Globally-Robust Network
GM	: Gradient Matching
HCI	: Human Robot Interaction
IoT	: Internet of Things
IRDC-Net	: Inception Residual Dilated Convolution Network
ISEMG	: Integrated Square-root EMG
ISTFT	: Inverse Short Time Fourier Transform
JSMA	: Jacobian-based Saliency Map Attack
L-BFGS	: Limited-Memory Broyden-Fletcher-Goldfarb-Shanno
l.s.c.	: lower-semicontinous
MAV	: Mean Average Value
MIMO	: Multi Input Multi Output
MMD	: Maximum Mean Discrepancy
MRI	: Magnetic Resonance Imaging
MSE	: Mean Squared Error
NLP	: Natural Language Processing
NN	: Neural Network
NMF	: Non-Negative Matrix Factorization
NP	: Non-Polynomial

PAC	: Probably Approximately Correct
PCA	: Principal Component Analysis
PGD	: Projected Gradianet Descent
PSNR	: Peak Signal to Noise Ratio
RADAR	: Radio Detection and Ranging
RCFF-Net	: Robust Complex Feed Forward Network
ReLU	: Rectified Linear Unit
RMS	: Root Mean Square
ROT	: Rotation Layer
SDP	: Semi-Definite Program
sEMG	: surface Electromyography
SGD	: Stochastic Gradient Descent
SLR	: Sign Language Recognition
SNR	: Signal to Noise Ratio
STFT	: Short Time Fourier Transform
SVD	: Singular Value Decomposition
VR	: Virtual Reality
WL	: Waveform Length
ZCR	: Zero Crossing Rate

List of Tables

List of Figures

Chapter 1 – Introduction

1.1 . Context

Recently, machine learning methods have become ubiquitous tools in a wide range of tasks, because of their ability to solve a great variety of problems, ranging from simple regressions to complex multi-modal classification. These methods stand at the very core of *Artificial Intelligence* (AI). AI represents the marvel of nowadays technology and is used successfully in an ever-increasing number of areas impacting our lives, e.g. medicine [3], autonomous driving [4], natural language processing [5], human-computer interaction (HCI) [6], etc. However, deep neural networks, which are probably the most powerful methods, raise challenges in terms of implementation heaviness during the learning phase. Moreover, they appear as black boxes whose robustness is not always well-controlled [7, 8]. Developing trustworthy AI is essential to ensure that intelligent systems can be relied upon for critical decision-making without compromising ethical standards. To reach this goal, a critical issue to be addressed when developing real-life applications using neural networks is the correct evaluation and control of their robustness against possible adversarial attacks.

Adversarial inputs represent malicious input data that can fool machine learning models. The concept was highlighted in [9], where the authors showed that slightly altering data inputs that were correctly classified by the network can lead to a wrong classification [10, 11, 12, 13].
For example, [11] shows how voice interfaces can be fooled by creating carefully crafted artificial audio inputs of unintelligible voice that are miss-classified as specific vocal commands by the system. Also, [14] introduces several methods for generating adversarial examples on ImageNet that are so close to the original data that differences are indistinguishable for the human eye.

It must be emphasized that adversarial inputs are not necessarily artificially created with the intention to sabotage the system. They can also occur innately under different forms and can seriously flaw the performance of real-life applications based on pre-trained models [15]. Here are a few ways in which adversarial perturbations can arise naturally:

(i) *Sensor Noise*: Sensors used to collect data can introduce noise or small variations due to factors such as environmental conditions, calibration issues, or manufacturing imperfections. These variations can cause a model to misinterpret the input.

(ii) *Ambiguity in Data*: Natural data can sometimes be inherently ambiguous, leading to different interpretations. A slight change in how an object is presented might lead to a model making incorrect decisions. The problem is often emphasized when the number of available training data is limited.

(iii) *Occlusions and Transformations*: Objects in the real world can undergo occlusions, deformations, and transformations that alter their appearance. These changes might be minor to human perception, but can significantly impact the performance of a machine learning model.

(iv) *Unforeseen Context*: Context matters in understanding data, and slight changes in context can lead to different interpretations. A model trained to recognize an object in one context might fail when the context changes unexpectedly.

(v) *Human Perception Differences*: Human perception is not perfect and can vary. What appears obvious to one person might be unclear to another. These differences can lead to unexpected variations in labeling and input data.

(vi) *Adversarial Intent in Real World*: In some cases, there might be genuine adversarial intent in the data itself. For example, camouflage in nature could be seen as an adversarial strategy where an organism uses its appearance to deceive predators or prey.

(vii) *System Limitations*: Imperfections in data collection, transmission, or preprocessing stages can introduce subtle changes that may not be initially apparent but can affect the model behavior.

A better analysis of the stability properties of neural networks can be viewed as the first step towards a better understanding of the mathematical principles governing their functionalities.

The main goal of this thesis is to design new methods for training safe yet high-performance neural networks. Recent mathematical results show that it becomes easier to control the stability of neural networks by introducing suitable constraints on their weights. Nevertheless, this requires the management of constraints that are not necessarily convex in the training phase of the neural network. To this end, we *designed carefully crafted constraints* that we later used in the training process, to ensure the robustness of the neural network. As highlighted in [14], the *Lipschitz behaviour* of the network is tightly correlated with its robustness against adversarial attacks. This constant allows us to upper bound the output perturbation knowing the magnitude of the input one, for a given metric [16]. Controlling this constant leads to a feasible solution to assess the effect of adversarial attacks if accurately computed. However, computing the exact Lipschitz constant even

for a shallow neural network is a non-polynomial *NP-hard* problem. So the main difficulty is to *find ways of approximating it as tightly as possible*. Lately, several methods have been proposed to train Lipschitzian networks, which fall into two main categories. Regularization approaches include double backpropagation [17] or applying penalization on the network Jacobian [18], which imposes local Lipschitz constraints, but do not enforce the constraint globally on the network. Another approach consists of imposing some constraints on the architecture of the network, so as to constrain the spectral norm of each layer [12] [19]. At the expense of computation complexity, these methods ensure a Lipschitzian network. In [20], novel results leading to accurate approximations to the Lipschitz constant of positive feed-forward neural networks were proposed. These preliminary results served as a starting point for proposing efficient methods for designing safe neural networks.

After establishing all the mathematical backbone, we next focus on *building new neural network architectures* based on the aforementioned philosophy. An important part of the work presented in this thesis consists in *developing efficient optimization methods for supervised learning of neural networks*. We look at the possible choices for the structure of the network, given the different classes of existing iterative optimization algorithms. To handle stability constraints, particular attention is paid to *proximal methods* which offer powerful tools for optimization in a large-scale context. We study how ensuring *robustness affects the overall performance* of the learning systems, and try to reach a good *robustness-accuracy trade-off*.

A very important aspect in all exploratory research is the *validation of the theoretical results in a real application context*. Some of the models trained with stability guarantees are tested in real-life contexts to show the versatility of the designed solutions. We then measure the influence on the system performance and *compare* the obtained results with those generated with classical architectures, as well as other defense strategies.

1.2 . Impact and applicability

Artificial neural networks have become the workhorse of many advances in artificial intelligence in the last years. Sophisticated inference tasks such as automatic medical diagnosis from 3D radiology images or monitoring social networks for preventing terrorist acts have become possible with a good accuracy thanks to deep learning. A large community of enthusiastic researchers and engineers is continuously working on building larger training databases and proposing more effective architectures to widen the scope of these powerful methods. However, it turns out that for domains where safety is a critical issue such as automatic driving, air-flight control, digital forensics, or surgery applications, neural networks offer

few guarantees in terms of certifiability. The main reason for this limitation is the potential lack of robustness to adversarial perturbations. This weakness reveals that neural networks are grounded on underlying mechanisms that are not fully understood from a mathematical viewpoint. They often appear as black boxes whose behaviour may be unpredictable. Analyzing neural networks is not an easy task since they constitute complex systems with intricate nonlinear behaviour.

This thesis contributes to the field of machine learning by trying to give an answer to the fundamental question:

How safe neural networks are?

The objective is to provide mathematically proven robustness guarantees, develop the associated software, and make it publicly available. Another important aspect of this thesis is the focus on applications based on audio and physiological signals which have direct use in the development of innovative technologies and can directly benefit a variety of consumer products.

More generally, by approaching the concept of Safe Neural Networks, this thesis contributes to the state of knowledge in artificial intelligence, leveraging on the latest research results in the field of optimization. Developing new methods that can be used to make learning systems more robust and explainable will open new perspectives in terms of safe and controlled technological progress.

1.3 . Main contributions

The first contributions of the thesis appear in Chapter 3:

(i) We propose a robust real-time Automatic Gesture Recognition system based on sEMG signals. The robustness is ensured by using a novel learning algorithm for training feedforward neural networks.

(ii) We show that a good accuracy-robustness balance can be reached. To do so, we train the system under carefully crafted spectral norm constraints, allowing us to finely control its Lipschitz constant. A tight Lipschitz constant is efficiently estimated by focusing on neural networks with nonnegative weights, as in [21].

(iii) We demonstrate the performance of the final architecture in real-life experiments, where we show that the proposed robust model outperforms those trained conventionally.

(iv) We analyze how our system behaves when the input is affected by different noise levels, simulating perturbations that may occur in real scenarios.

(v) We show the validity of our solution by experimenting on several publicly available sEMG gesture datasets.

Chapter 4 includes the following main contributions.

(i) Inspired by MIMO filters, we introduce a new class of neural networks, which can be seen as an intermediate solution between CNNs and FCNs.

(ii) We propose a constrained training strategy, which allows us to control the Lipschitz constant of the network in order to secure its robustness to adversarial noise.

(iii) We present a new architecture (RCFF-Net), which operates in the complex-valued domain, for which we derive tight Lipschitz constant bounds.

(iv) We develop a constrained learning strategy to train the proposed structure while controlling its global Lipschitz constant.

(v) Both architectures ACNN and RCFF are evaluated in audio signal denoising tasks, proving that our solution is not limited to classification problems.

The contributions from Chapter 5 are mentioned below.

(i) We introduce ABBA networks, a novel class of (almost) non-negative neural networks, which are shown to possess a series of appealing properties.

(ii) We show that we can put any arbitrary signed network in an ABBA form. We show that this property holds for fully connected as well as for convolutional neural networks.

(iii) Universal approximation theorems are derived for networks featuring non-negatively weighted layers.

(iv) We present a method for effectively controlling the Lipschitz constant of ABBA networks. This control strategy applies to both fully connected and convolutional cases.

(v) Numerical experiments conducted on standard image datasets showcase the excellent performance of ABBA networks for small models. Notably, they exhibit substantial improvements in both performance and robustness compared to networks with exclusively non-negative weights. Moreover, we demonstrate that ABBA networks are competitive with robust networks featuring arbitrarily signed weights, trained using state-of-the-art techniques.

1.4 . Publications

Submitted journal articles

- A. Neacșu, J.-C. Pesquet, V. Vasilescu and C.Burileanu, *"ABBA Neural Networks: Coping with Positivity, Expressivity, and Robustness"*, submitted to SIAM Journal on Mathematics of Data Science (SIMODS), 2023.

Accepted or published journal articles

- A. Neacșu, J.-C. Pesquet and C.Burileanu, *"EMG-Based Automatic Gesture Recognition Using Lipschitz-Regularized Neural Networks"*, accepted for publication in ACM Transactions on Intelligent Systems and Technology (TIST), 2023.

- N Lassau, S. Ammari, E. Chouzenoux, A. Neacșu et al. *"Integrating deep learning CT-scan model, biological and clinical variables to predict severity of COVID-19 patients"*,in Nature Communication 12, 634 (2021), https://doi.org/10.1038/s41467-020-20657-4

Conference Proceedings

- C. Andronache, M. Negru, I. Bădițoiu, G. Cioroiu, A. Neacsu and C. Burileanu, *"Automatic Gesture Recognition Framework Based on Forearm EMG Activity"*, in Proc. 45th International Conference on Telecommunications and Signal Processing (TSP), Prague, Czech Republic, 2022, pp. 284-288, doi: 10.1109/TSP55681.2022.9851314.

- A. Neacșu, R. Ciubotaru, J. -C. Pesquet and C. Burileanu, *"Design of Robust Complex-Valued Feed- Forward Neural Networks"*, in Proc. 30th European Signal Processing Conference (EUSIPCO), Belgrade, Serbia, 2022, pp. 1596-1600, doi: 10.23919/EUSIPCO55093.2022.9909696.

- A. Neacșu, K. Gupta, J. -C. Pesquet and C. Burileanu, *"Signal Denoising Using a New Class of Robust Neural Networks"* in Proc. of 28th European Signal Processing Conference (EUSIPCO), Amsterdam, Netherlands, 2021, pp. 1492-1496, doi: 10.23919/Eusipco47968.2020.9287630.

- V. Vasilescu, A. Neacșu, E. Chouzenoux, J. -C. Pesquet and C. Burileanu, *"A Deep Learning Approach For Improved Segmentation Of Lesions Related To Covid-19 Chest CT Scans"*, in Proc. IEEE 18th Int. Sym. on Biomedical Imaging (ISBI), Nice, France, 2021, pp. 635-639, doi: 10.1109/ISBI48211.2021.9434139.

- A. Neacșu, J.-C. Pesquet, and C. Burileanu, *"Accuracy-robustness trade-off for positively weighted neural networks"*, in Proc. IEEE International Conference on Acoustics and Speech Signal Process. (pp. 8389–8393). Barcelona, Spain, 2020, doi: 10.1109/ICASSP40776.2020.9053803.

- C. Andronache, M. Negru, A. Neacsu, G. Cioroiu, A. Radoi and C. Burileanu, *"Towards extending real-time EMG-based gesture recognition system"*, in Proc. 43rd International Conference on Telecommunications and Signal Processing (TSP), Milan, Italy, 2020, pp. 301-304, doi: 10.1109/TSP49548.2020.9163481.

1.5 . Co-tutelle thesis

Collaboration lies at the heart of scientific progress and innovation. In today's interconnected world, the significance of collaborative efforts cannot be overstated, particularly in the field of academic research. This thesis is the result of a co-tutelle collaboration, between University Politehnica of Bucharest and CentraleSupélec, Graduate School of Engineering Sciences of University Paris Saclay. This thesis has provided a remarkable opportunity to foster cooperation and exchange knowledge between these two distinguished institutions.

By bringing together the expertise and perspectives from both universities, this thesis embodies the spirit of collaboration within EU, transcending geographical boundaries and cultural differences. The shared intellectual environment, research projects, and joint supervision have facilitated a rich learning experience, allowing for the synthesis of diverse ideas and methodologies. This collaborative endeavor has not only broadened the horizons of the research presented in this work but has also cultivated a sense of global scientific community and mutual understanding.

During the unprecedented challenges posed by the COVID-19 pandemic, our work took on a dual significance. Beyond the purview of our academic pursuits, we understood the importance of contributing to the global endeavour to combat the pandemic. As part of our commitment, we developed an algorithm for the automatic segmentation of the lungs. This led to two publications.

1.6 . Outline

The rest of the thesis is organized as follows. In Chapter 2, we present an overview of existing attacks and defenses. In Section 2.1 we establish the concept of robustness in the context of neural networks, while in Section 2.2 we introduce the mathematical notation used throughout the chapter. We present the most used scenarios of threat models (Section 2.3) and then we describe both white-box and black-box attack mechanisms in Section 2.4. We end the chapter by emphasising different defense strategies in Section 2.5.

In Chapter 3 we present a robust mechanism for training non-negative neural networks in the context of automatic gesture recognition based on sEMG signals.

In Section 3.1 we lay the foundational understanding of electromyography and emphasize its relevance in the context of gesture recognition. Following this, in Section 3.2 we introduce innovative approaches to enhance the robustness of fully connected neural networks. Section 3.3 then details the optimization techniques crucial to our proposed methods, variants of which will be used in the rest of this work. Transitioning to practical implementation, Section 3.4 provides insights into the experimental framework considered for our task. The chapter culminates with Section 3.5 where we extensively validate the robustness of our proposed models. Finally, we conclude this chapter by summarizing our key findings and their implications in Section 3.6.

In Chapter 4 we embark on a journey to enhance signal denoising through innovative robust neural network architectures. Starting with Section 4.1 we introduce the first novel architecture we propose in this thesis. We then explore, in Section 4.1.1, a critical step in bridging the gap between these powerful neural network paradigms: the use of fully connected and convolutional layers. Section 4.1.2 delves into the optimization strategies employed for training our proposed models, shedding light on the core of our methodology. Our practical applications developed in Section 4.1.3 provide an in-depth examination of our model performance in signal denoising scenarios. The second part of the chapter, starting with Section 4.2 introduce a new class of networks (RCFF) operating in the complex domain. Theoretical foundations and insights are presented in Sections 4.2.1-4.2.3 where we elucidate the mathematical underpinnings of our robust training mechanisms, and then we detail its implementation in Section 4.2.4. Then, we showcase the empirical outcomes of applying our RCFF-Net to audio denoising problems in Section 4.2.5. Ultimately, we conclude this chapter by summarizing our key findings and their implications in Section 4.3.

In Chapter 5, we introduce a groundbreaking class of neural networks known as ABBA Neural Networks, engineered to grapple with issues of positivity, expressivity, and robustness. We start with Section 5.1 offering an overview of the challenges that our novel ABBA networks aim to address. We provide context in Section 5.2, examining the existing landscape of neural network solutions and underscoring the unique contributions of ABBA networks. The core of our chapter unfolds with Section 5.3 where we describe the architectural foundations and key attributes of this innovative neural network class. Subsequently, in Section 5.4, we extend the applicability of ABBA networks to the convolutional case, highlighting the adaptability of this approach across diverse network architectures. An in-depth look into the training methods and techniques ensuring Lipschitz stability is presented in Section 5.5. Section 5.6 serves as the empirical heart of this chapter, where we conduct comprehensive evaluations to validate the performance and effectiveness of ABBA networks across various classification scenarios. In Section 5.7, we sum

up our key findings, insights, and implications of our research.

Finally, in Chapter 6, we draw the final remarks of this thesis, followed by a brief description of some envisioned perspectives.

Chapter 2 – Overview of adversarial attacks and defenses

This chapter presents an overview of the current advancements in the domain of the robustness of neural networks against adversarial perturbations. We will define the concept of adversarial attacks and explain the insights of the most efficient attack strategies. Studying deliberately crafted attacks in machine learning is crucial because it allows to identify the vulnerabilities of models and enhances their robustness. These attacks involve intentionally perturbing input data in subtle ways that can lead machine learning models to misclassify or produce erroneous outputs. While these attacks are often designed on purpose, they sometimes mimic variations that can naturally occur in real-world data. By analyzing these attacks, researchers gain a deeper understanding in the weaknesses of their models and can develop more robust algorithms that are better able to handle unexpected variations in input data. Next, we will present the main mechanisms for defense, which constitutes a challenging research topic.

2.1 . Robustness of neural networks

Neural networks (NNs) have emerged as effective tools for a variety of applications, including image recognition, natural language processing, and decision-making systems. Despite these accomplishments, numerous fundamental features of deep neural networks remain unknown and have been the focus of significant research in recent years. Due to its importance when applied to visual data, the robustness of deep networks to various types of disturbances has garnered increasing attention. This line of research was largely inspired by the demonstration of the intriguing properties of deep networks, first reported by [9]. Concerns have been expressed

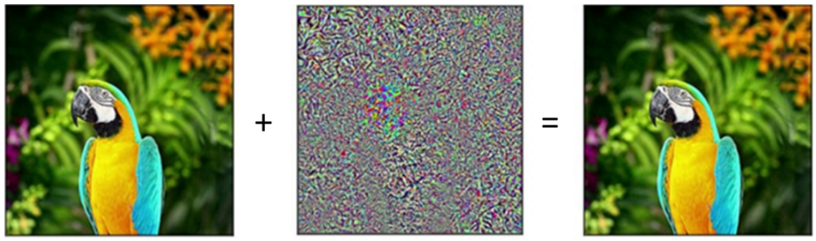


Figure 2.1: Example of adversarial example. It can be observed that by adding a small perturbation, which is imperceptible to the human eye, the model changes drastically its prediction.

regarding neural network dependability and security due to their susceptibility to adversarial inputs, also known as adversarial examples. Adversarial inputs are small perturbations, deliberately designed to trick neural networks, resulting in inaccurate predictions or misclassifications. Figure 2.1 shows the effect of adding a very small, carefully crafted perturbation field over the original image. Considering an image classification problem, Shamir *et al.*[22] presents a framework to monitor how the decision boundary between different classes evolves during the training process, showing some interesting insight into why adversarial perturbations are so effective. Adversarial examples are not limited to digital images, they can be present in the physical world as well [10, 23]. In [24] several real-life adversarial examples are presented to highlight the impact and potential risks of adversarial attacks in practical settings. These examples demonstrate the vulnerability of machine learning systems to subtle manipulations and the potential consequences in various domains. A few notable examples are:

(i) *Physical Adversarial Examples* – Eykholt *et al.*[25] demonstrate how physical modifications can deceive object detection systems. By placing stickers on a stop sign or altering traffic signs with carefully designed patterns, they show that the modified signs can be misclassified by the object detection models, potentially leading to dangerous consequences on the road.

(ii) *Textual Adversarial Examples* – In the context of natural language processing, the work in [26] presents adversarial examples for sentiment analysis. By making small changes to the text, such as replacing or adding specific words, they demonstrate that sentiment analysis models can be manipulated to produce incorrect classifications or alter the predicted sentiment.

(iii) *Printed Image Attacks* – The authors of [25] investigate attacks using printed images that are designed to deceive deep learning models. By placing specially crafted images in the environment, i.e. on posters or billboards, they demonstrate how these images can cause misclassifications or trigger specific target classifications, even from a distance.

(iv) *Adversarial Examples in Recommender Systems* – Xu *et al.*[27] discuss how adversarial examples can impact recommender systems. By carefully crafting user profiles or manipulating item features, attackers can influence the recommendations made by the system. This can have implications in personalized advertising or content curation, where malicious actors may try to exploit the system for their own benefit or to spread misinformation.

Recent years of research have been devoted to comprehending and enhancing the resilience of neural networks to such adversarial inputs. The rest of this chapter investigates the concept of robustness in neural networks, explains how such perturbations are created, and discusses techniques used to limit their effects.

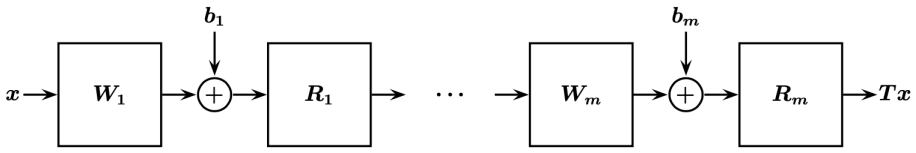


Figure 2.2: Representation of a NN as a composition of operators

2.2 . Definitions and notation

A neural network with m layers can be viewed as a function T, admitting an input $x \in \mathbb{R}^{N_0}$ and delivering an output $T(x) \in \mathbb{R}^{N_m}$, where $(N_i)_{0 \leqslant i \leqslant m} \in (\mathbb{N} \setminus \{0\})^{m+1}$ is the number of neurons on each layer. In classification scenarios, N_m is the total number of classes.

Model 2.2.1 Any feedforward neural network is obtained by cascading m layers associated with operators $(T_i)_{1 \leqslant i \leqslant m}$. The neural network can thus be expressed as the following composition of operators:

$$T = T_m \circ \cdots \circ T_1. \tag{2.1}$$

Each layer $i \in \{1, \ldots, m\}$ has a real-valued vector input x_i of dimension N_{i-1} which is mapped to

$$T_i(x_i) = R_i(W_i x_i + b_i), \tag{2.2}$$

where $W_i \in \mathbb{R}^{N_i \times N_{i-1}}$, $b_i \in \mathbb{R}^{N_i}$ are the weight matrix and bias parameter, respectively. $R_i \colon \mathbb{R}^{N_i} \to \mathbb{R}^{N_i}$ constitutes a non-linear activation operator which is applied component-wise (e.g., ReLU or Sigmoid) or globally (e.g, Softmax). Figure 2.2 shows a graphical representation of this concept.

Usually, for multiclass problems, the output of such systems is the result of a Softmax operator, which produces a vector that can be viewed as a discrete probability distribution. In other words, the output could be interpreted as the probability that the input x belongs to each class i. The inputs of the softmax operator are known as *logits* and represent the confidence of the classifier. To obtain the predicted label $\overline{T}(x)$, we select the maximum argument of the components of $T(x)$. Considering a small perturbation $z \in \mathbb{R}^{N_0}$, an *adversarial example* is defined as

$$\tilde{x} = x + z. \tag{2.3}$$

The perturbations can have a given form, e.g. in the case of images they can be the result of some geometric (affine) transforms or can be random additive perturbations, created using different algorithms which will be later explored. A key

insight to consider when generating adversarial perturbations is that the original input x and its adversarial $\tilde{x}$ should be as close as possible given a similarity measure. The minimal perturbation required to achieve an adversarial example is usually formulated as the minimization of a distance metric.

Consider the model T from Model 2.2.1, and $(x, y) \in \mathbb{R}^{N_0} \times \mathbb{N}$ a pair of input and the associated label examples. The minimal adversarial perturbation associated to this pair is:

$$z_{\mathsf{min}} = \operatorname{argmin}_z \|z\| \quad \text{s.t.} \quad \overline{T}(x + z) \neq y, \tag{2.4}$$

where $\|\cdot\|$ is usually chosen an ℓ_p norm. We recall that the ℓ_p norm with $p \in [1, +\infty[$ of a vector $\boldsymbol{\phi} = (\phi_i)_{1 \leqslant i \leqslant n} \in \mathbb{R}^n$ is

$$\|\boldsymbol{\phi}\|_p = \left(\sum_{i=1}^{n} |\phi_i|^p \right)^{1/p}, \tag{2.5}$$

and the associated ℓ_p distance between ϕ and a vector $\overline{\phi} = (\overline{\phi}_i)_{1 \leqslant i \leqslant n}$ is given by $\|\phi - \overline{\phi}\|_p$. Standard choices for p are 0, 2 and ∞, as briefly described below.

- ℓ_0 distance (which not associated with a norm but a pseudo-norm) counts the number of indices i for which $\phi_i \neq \overline{\phi}_i$. In the context of image processing tasks, where n corresponds to the image size, ℓ_0 distance is the number of altered pixels in the original image.
- ℓ_2 distance corresponds to the classic Euclidean distance and it is the most used distance in adversarial settings.
- ℓ_∞ distance measures the greatest variation in any of the components, defined as

  $$\|\phi - \overline{\phi}\|_{+\infty} = \max(|\ \phi_1 - \overline{\phi}_1\ |, \ldots, |\ \phi_n - \overline{\phi}_n\ |).$$

  If vector x corresponds to an image luminance, it is common to impose a maximum budget, which determines the maximum allowable perturbation for each pixel.

The minimal perturbation can identify the adversarial instance that most closely resembles the original input in a given vector space. However, it is important to note that no distance metric is a precise measure of human perceptual similarity, and there is no determination regarding the optimal distance metric. Constructing and evaluating an effective distance metric is a crucial research question that is the topic of ongoing investigation.

In this context, the *robustness* of input x given the model T can be formulated as

$$r(x, T) = \|z_{\mathsf{min}}\|, \tag{2.6}$$

where z_{min} is the minimum norm perturbation that have been identified.

We can further extend this definition to the input set $\mathcal{S}$, defining the *global robustness* of T as

$$\overline{r} = \mathop{\mathbb{E}}_{x \sim \mathcal{N}} r(x, T), \tag{2.7}$$

where $\mathbb{E}$ is the expected value, computed with respect to a probability distribution $\mathcal{N}$ supported on $\mathcal{S}$.

Naturally, as r and $\overline{r}$ increase, the adversary is required to sacrifice similarity in order to generate a more efficient attack. Consequently, we can infer that the model exhibits greater robustness in such scenarios. Thus, we can now formulate the concept of *most adversarial* example as the one which has the largest loss value in a ϵ-radius ball around $\tilde{x}$:

$$\tilde{x}_{\text{adv}} = \text{argmax}_{\tilde{x}} \quad \mathcal{L}(\eta, \tilde{x}, y, T) \quad \text{s.t.} \quad \|\tilde{x} - x\| \leqslant \epsilon. \tag{2.8}$$

Here-above, $\mathcal{L}$ is the considered loss, η is a vector including all the model parameters, and ϵ represents the norm of the highest perturbation allowed. The *adversarial loss* associated with this example is

$$\mathcal{L}_{\text{adv}}(x) = \mathcal{L}(\eta, \tilde{x}_{\text{adv}}, y, T). \tag{2.9}$$

The model is all the more robust as the loss $\mathcal{L}_{\text{adv}}(x)$ is small. Calculating the expectation over $\mathcal{N}$ yields the *global adversarial loss*

$$\overline{\mathcal{L}}_{\text{adv}} = \mathop{\mathbb{E}}_{x \sim \mathcal{N}} \mathcal{L}_{\text{adv}}(x). \tag{2.10}$$

Table 2.1 summarizes the notation used throughout the chapter.

2.3 . Threat models

Adversarial attacks are mainly studied in the context of classification tasks. In a recent work, Gupta *et al.* [28] proposed an attacker suited for regression problems, based on the properties of the network Jacobian, but the research in this particular direction is limited. Since the main focus in the literature is on studying attacks and defenses for neural models trained as classifiers, the remainder of this chapter will focus on this topic.

The optimization problem introduced in (2.4) is non-convex, since the constraint relies on the classification model T, which may be quite complex, depending on its architecture. Lately, several (mostly iterative) algorithms have been proposed in the literature to approximate adversarial perturbations. Depending on their objective and level of access to the original model, they can fall in several categories, as explained next.

Notation	Short description
x	: original input sample, $x \in \mathbb{R}^{N_0}$
$\tilde{x}$	: adversarial input sample, $\tilde{x} \in \mathbb{R}^{N_0}$
z	: perturbation $z \in \mathbb{R}^{N_0}$
$\mathcal{B}_p(\tilde{x}, \epsilon)$	: ℓ_p ball with center $\tilde{x}$ and radius ϵ
ϵ	: the maximum norm of the perturbation z
$\|\| \cdot \|\|_p$	: ℓ_p norm
N_0	: input data dimension
$\mathcal{N}$	: original data distribution
y	: ground truth label associated to x,
t	: target label
N_m	: the total number of classes
$\mathcal{L}$	: the loss function employed during training
m	: the total number of layers
T^*	: an m-layer neural-network model having logits as output (before the *softmax* operator)
T	: an m-layer neural-network model having a score-vector output *i.e.* $T(x) \in [0,1]^{N_m}$
$\overline{T}$	: an m-layer neural-network model having a label as output
R_i	: activation operator at layer i
η	: parameter vector associated to Model T
θ	: Lipschitz constant associated to Network T

Table 2.1: Mathematical notation

2.3.1 . Adversary's objective

Targeted and untargeted attacks.

We consider an input $x \in \mathbb{R}^{N_0}$, y its associated label, which is normally correctly classified by the model. Let t be a *target* such that $t \neq y$. The adversary strives to find a similar input $\tilde{x}$ (according to some distance metric), for which the classification is incorrect. In this case, $\tilde{x}$ is considered as a *targeted* adversarial example. There are different strategies for selecting the target class t. The main scenarios are:

- *Best-case scenario* – attack all incorrect classes and select the target class that was least challenging to attack.
- *Worst-case scenario* – attack all incorrect classes and select the target class that was the most difficult to attack.
- *Mean-case scenario* – choose the target class at random from among the misclassified labels.

It is also possible to create *untargeted* adversarial examples, where we do not impose a given target class, but we strive to find an adversarial input $\overline{x}$ such that $T(\overline{x}) \neq y$, while the similarity between x and $\overline{x}$ is maintained. Usually, targeted attacks require higher perturbations than untargeted ones.

Poison and evasion attacks.

Poisoning attacks imply that the attacker can modify the dataset on which the model T is trained on, by introducing fake samples. In real scenarios, a lot of web-based repositories actively collect data for training malware detection models,

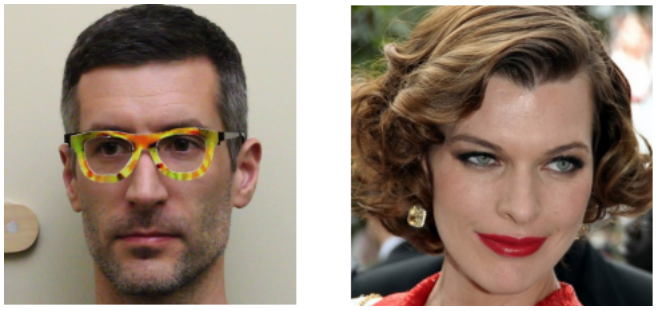


Figure 2.3: Adversary wearing adversarial glasses fools face recognition model. By creating fine-crafted perturbations the attacker is able to confuse the model to say with high probability that the input image contains the face of the actress Milla Jovovich. [1]

which offers an open door for adversaries to deliberately poison the data. Boggio *et. al.* [29] showed that, by employing an incremental learning scheme for SVMs, it is possible to create fake input samples which leads to poor classification results. Although this procedure was successful in fooling SVM models, it is quite challenging to find the influence of the training samples in the context of deep neural models. Some steps were made in this direction by [30], where the authors tried to answer a particularly difficult question, namely *How would the model's predictions change if a training sample was modified?*

Another notable work introduces *poison frogs* [31], a mechanism that implies inserting in the training set some adversarial images having a true label associated, which leads the trained model to misclassify the original test samples.

On the other hand, in the context of evasion attacks, the models are already trained and they usually have good performance on standard test examples. The adversaries strive to create some fake examples, to *evade* detection by the classifiers. For example, in [1] they show that it is possible to mislead state-of-the-art face recognition systems, by creating adversarial glasses. Figure 2.3 depicts such an example, where the model predicted that the adversary is a well-known actress, Milla Jovovich.

2.3.2 . Adversary's level of access

Depending on the adversary's knowledge, there are three main categories of attacks.

White-box attacks – In this environment, the adversary can access all the target neural network information, such as its weights, hyperparameters, and gradients. Thus, the adversary can aggregate all this information and use it to create custom adversarial samples. The exposure of the model architecture and parameters allows

users to explicitly understand the vulnerabilities of deep neural network (DNN) models. As a result, white-box attacks have been extensively researched Additionally, these attacks may be examined mathematically and may offer some clues about the representations learned by the model [32]. Hence, security against white-box attackers is of crucial importance.

Black-box attacks [33] – In this set-up, the adversary cannot access the inner configuration of the model. He can only study how the model behaves when facing different inputs and query the outputs, trying to figure out some of the model's weaknesses. Although black-box attacks are less powerful than white-box ones, they are more realistic from the attacker's viewpoint.

Gray-box attacks – Also known as semi-white attacks, gray-box attacks represent a nuanced approach, lying between white box and black box attacks. One possible approach found in the literature is the following one. Consider a victim model T, the adversary designs a generative model, consisting in a generator $\mathcal{G}$ and a discriminator $\mathcal{D}$. The generator aims to create a perturbation $\mathcal{G}(x)$ which will be added over the clean input x. $x + \mathcal{G}(x)$ is then fed to the discriminator, which has the task of encouraging $\mathcal{G}$ to create perturbations close to the original data. In order to achieve the objective of deceiving a learning model, the initial step involves running a white-box assault, on the target model. The victim model T accepts an input of $x + \mathcal{G}(x)$ and produces a loss value $\mathcal{L}_{\text{adv}}$, which quantifies the discrepancy between the predicted class and the target class t in the case of a targeted attack, or the inverse of the discrepancy between the predicted class and the true class in the case of an untargeted attack, and it is used as a part of the total loss function used to train $\mathcal{G}$. After this model is fully trained, the attacker no longer needs to access the victim model, so it can generate adversarial perturbations in a black-box setting. Some of the most efficient gray-box attackers are presented in [34, 35].

2.4 . Attack mechanisms

Next, we detail the main algorithms for generating adversarial samples in all three contexts. We consider mainly evasion methods since they are more common.

2.4.1 . White-box attacks

Consider the model T and a pair of input-output (x, y). In a white-box setting, the goal of the adversary can be formulated as

$$\text{find} \quad \tilde{x} \quad \text{satisfying} \quad \begin{cases} \|\tilde{x} - x\|_p \leqslant \epsilon \\ \overline{T}(\tilde{x}) \neq y, \end{cases} \tag{2.11}$$

where ϵ is a positive constant. Variants of this formulation will be considered hereafter. Biggio *et al.* [36] were the first to introduce such kind of examples to

fool image classification models based on SVMs and shallow convolutional networks.

Limited-Memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) – The first attack for deep learning models was introduced by Szegedy *et al.*[9], where the optimal adversarial example was sought by introducing the following optimization problem:

$$\min \|\tilde{x} - x\|_2^2 \quad \text{s.t.} \quad \overline{T}(\tilde{x}) = t \quad \text{and} \quad \tilde{x} \in [0,1]^{N_0}. \tag{2.12}$$

They reduced the complexity of the problem, by introducing the following penalized problem

$$\underset{\tilde{x}}{\text{minimize}} \;\; c\|\tilde{x} - x\|_2^2 + \mathcal{L}(\eta, \tilde{x}, t, T) \quad \text{s.t.} \quad \tilde{x} \in [0,1]^{N_0}, \tag{2.13}$$

where $c \geqslant 0$ is a constant that can be adapted. The solution they propose to find the optimal $\tilde{x}$ relies on an instance of L-BFGS algorithm [37].

Fast-gradient sign method (FGSM) – This method, introduced by Goodfellow *et al.* [14], shows a way to generate fast adversarial examples using a one-step method, for both targeted and untargeted settings. The proposed solution is based on the following formulation:

$$\tilde{x} = x + \epsilon \text{sgn}(\nabla_x \mathcal{L}(\eta, x, y, T)), \quad \text{non-target} \tag{2.14}$$

$$\tilde{x} = x - \epsilon \text{sgn}(\nabla_x \mathcal{L}(\eta, x, t, T)), \quad \text{target on } t. \tag{2.15}$$

In the case of untargeted attacks, the algorithm tries to maximize the loss between the predicted output and the ground truth label class. In a targeted attack context, this can be viewed as one-step of gradient descent aiming to solve the following optimization problem:

$$\underset{\tilde{x}}{\text{minimize}} \quad \mathcal{L}(\eta, \tilde{x}, t, T) \quad \text{s.t.} \quad || \tilde{x} - x ||_\infty \leqslant \epsilon. \tag{2.16}$$

This means that the algorithm searches the point that is most likely to fool the model, which is where the loss value of the target label t is minimum in the neighborhood of an ϵ-ball around $\tilde{x}$.

While time-efficient, this method yields only a coarse approximation to the optimization problem defined in Eq. (2.11) for the case when $p = +\infty$.

DeepFool [38] – DeepFool strategy investigates the decision boundary of T with respect to an input example x. It aims to find a path that enables moving outside the decision boundary. This is done iteratively, to find the optimal point where the classifier will assign a false class for x. This can be viewed as the linearization of the constraint defined in Eq. (2.11).

Suppose we want to attack a sample $\overline{x}$, to change its label from class k to class i. The decision boundary between the two classes can be mathematically modeled

as $\mathcal{F}_{k,i} = \{z \mid T_k(x) = T_i(x)\}$. At each iteration, the attack linearizes the decision boundary using a first-order Taylor series expansion, as follows:

$$\widetilde{\mathcal{F}}_{k,i} = \{x \mid \zeta(x) \sim \zeta(\overline{x}) + (\nabla_x \zeta(\overline{x}))^\top (x - \overline{x}) = 0\},$$

where $\zeta(x) = T_k(x) - T_i(x)$, and computes the orthogonal projection of $\overline{x}$ onto the hyperplane $\overline{\mathcal{F}}_{k,i}$. The adversarial example is then found along the projection error vector.

The results show that this method is very effective, especially for image classification tasks, where usually the test samples are close to the decision boundary [38].

Jacobian-based saliency map attack (JSMA) [39] – This algorithm implies creating the saliency map of the model T, based on its Jacobian matrix $J_T(\overline{x}) = \frac{\partial T(\overline{x})}{\partial x^\top}$. $J_T(x)$ is used to compute the change of the model T output in relation to changes of the input x. At a macroscopic level, the attack employs a greedy algorithm that sequentially selects individual components of the input vector to change, so incrementally enhancing the target classification with each repetition. The saliency map serves as a representation of the influence that each individual component exerts on the final classification outcome. A high numerical value suggests that modifying it will considerably enhance the probability of the model categorizing the input as the desired class t. Based on the saliency map, the algorithm selects the input component with the highest importance and adjusts its value in order to enhance the probability of belonging to t. This process is iteratively performed until either the number of modified components exceeds a predetermined threshold, rendering the attack identifiable, or the attack successfully changes the prediction of the victim model. A variant of this method, adapted for regression problems, was developed in [28].

Projected gradient descent (PGD) [10] – This method introduced by Kurakin *et al.* represents an iterative version of the FGSM attacker. At iteration $n \geqslant 0$ the algorithm reads

$$x_{n+1} = \mathsf{P}_{\mathcal{B}_\infty(x,\epsilon)}(x_n + \alpha \nabla_x \mathcal{L}(\eta, x_n, y, T)), \tag{2.17}$$

where $\mathsf{P}_{\mathcal{B}_\infty(x,\epsilon)}$ represents the projection onto the ball $\mathcal{B}_\infty(x,\epsilon) = \{\tilde{x} \mid \|\tilde{x} - x\|_\infty \leqslant \epsilon\}$ and $\alpha \in]0, \infty[$ is the step-size.

Carlini and Wagner (C&W) [40] – Carlini and Wagner's attack is also inspired by the L-BFGS strategy from [9], the main difference being that C&W cleverly define the margin loss $f(x,t)$, which allows them to minimize the distance between the input x and the associated adversarial sample $\tilde{x}$.

They reformulate the objective from Eq. (2.11) as follows:

$$\underset{\tilde{x}}{\text{minimize}} \;\; \|x - \tilde{x}\|_2^2 + c f(\tilde{x}, t) \quad \text{s.t.} \quad \tilde{x} \in [0,1]^{N_0}, \tag{2.18}$$

where $f(\tilde{x}, t) = [\max_{i \neq t} \quad T_i^*(\tilde{x}) - T_t^*(\tilde{x})]^+$, $[x]^+$ denotes $\max(x, 0)$ calculated componentwise, and c a positive constant that is determined via a line search algorithm. This method has proven to be effective against defensive distillation techniques [41].

Other attacks – Decoupling direction norm (**DDN**) [42] and Fast minimum norm (**FMN**) [43] attacks fall into the category of projected-gradient methods, using iterative updates of the perturbation vector towards the minimization of its magnitude. FMN extends the DDN attack by adapting to other distance metrics and providing several improvements from an optimization viewpoint. Augmented Lagrangian method for adversarial (**ALMA**) [44] propose the use of Augmented Lagrangian algorithms to create minimally perturbed adversarial samples. It combines the versatility of penalty methods with the computational efficiency of distance-customized algorithms. This approach can be easily applied to a variety of distance metrics.

2.4.2 . Black-box attacks

Substitute model – In the context where the adversary has no access to the parameters of the victim model, a pioneer work was introduced by Papernot *et al.* [45], where they showed an interesting property of adversarial examples, namely *transferability*. They show that they can efficiently fool DNN-based classifiers by training a substitute model T' that has a similar structure to the original one. They show that adversarial samples crafted on T' using some white-box mechanisms can be also efficient in fooling the original model T.

Query-based methods –Several research efforts have focused on making the process of producing black-box adversarial instances using a finite amount of queries more efficient. The authors of [46] presented a method that is more effective in estimating the gradient information based on the model outputs. They make use of natural evolutionary strategies proposed in [47], which involve taking a sample of the model output based on the queries that are performed around and estimating the expected gradient of T with respect to the input x.

By analyzing the decision boundary of the model, one might obtain a new understanding of the existence of adversarial cases. The adversarial examples are often quite close to the decision boundary of a naturally trained model. This may be due to the fact that the decision boundary is either inflexible [48], or too curved [49], or too flat [50]. Researching the rationale behind the existence of adversarial instances is crucial because it can guide us in the development of more robust models and assist us in better comprehending those that already exist in the field of deep learning. Even though, up to this moment, there is no unified

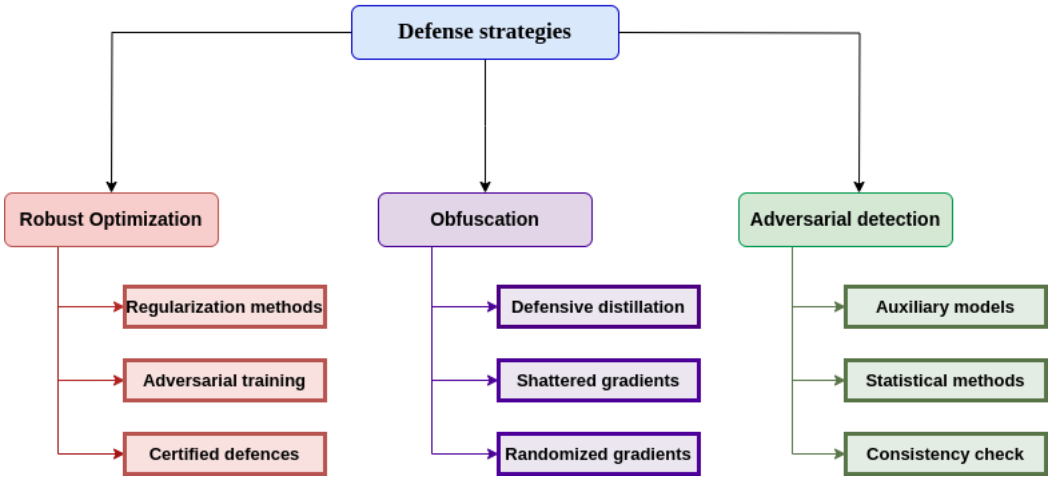


Figure 2.4: The taxonomy of defense against adversarial attacks strategies.

view on this problem, there are several defense mechanisms proposed in the literature to limit the effect of adversarial attacks, which will be discussed in the following.

2.5 . Defense strategies

Since there are many ways an adversary can exploit the model's weaknesses, defensive strategies have been developed to alleviate this robustness issue.

Silva *et al.* [51] divide adversarial defense methods into three categories, as follows.

(i) *Robust Optimization* techniques protect the model by imposing some robustness constraints during the training phase, or by augmenting the dataset. These techniques are proven to upgrade the model stability with respect to adversarial attacks, but they may hinder the performance of the model.

(ii) *Obfuscation*, also known as gradient masking, focuses on building models with gradients that cannot be used by adversaries. In general, obfuscated models generate smooth loss functions in the neighborhood of the input samples, which can confound adversaries and improve robustness.

(iii) *Adversarial example detection* strategies concentrate on detecting perturbed inputs using an additional module (usually another neural network specialized in adversarial detection), or by performing some statistical tests on the input data.

A taxonomy of existing defense strategies is presented in Figure 2.4.

2.5.1 . Robust Optimization

The main idea behind robust optimization methods is to study how to change the way of learning the model so that representations will be obtained that are stable against adversarial inputs. In this context, the vast majority of research focuses mainly on learning a new set of parameters η^* which minimize the average adversarial loss defined in Eq. (2.10):

$$\eta^* = \text{argmin}_\eta \mathop{\mathbb{E}}_{x\sim\mathcal{N}} \max_{||\tilde{x}-x||\leqslant\epsilon} \mathcal{L}(\eta, \tilde{x}, y, T), \tag{2.19}$$

where $\mathcal{N}$ denotes the distribution of the input data.

Robust optimization mechanisms usually aim to create classifiers that are robust against small ℓ_p perturbations [52]. Although they do not provide stability certificates against non-ℓ_p attacks (e.g., spatial attacks [53]), it is fundamental to understand how to assess and control the robustness in the case of ℓ_p attacks, which can stand as a baseline for further generalizations. The rest of this section will focus on various defense strategies in the case of ℓ_p norm attacks.

Regularization methods – An early work of Rifai *et al.*[54] introduced contractive auto-encoders, as a solution for extracting better feature representations from images. They show that introducing a penalization factor corresponding to the Frobenius norm of the activation's Jacobian matrix w.r.t. its input helps improve the model's generalization capabilities while reducing overfitting. Inspired by their work, Gu *et al.*[55] extended this idea and proposed an algorithm for building deep contractive networks. By adding a penalty term on the Jacobian matrix of each layer during the back-propagation phase, they limit the effect that a slight perturbation of the input data will have on the final prediction, hence improving the model's stability.

Control of the Lipschitz constant A similar defense strategy aims to control the Lipschitz constant of the network. As highlighted in [14], the Lipschitz behaviour of the network is tightly correlated with its robustness against adversarial attacks. The Lipschitz constant allows to upper bound of the output perturbation knowing the magnitude of the input one, for a given metric [16]. Given the model T, for every $(x, z) \in (\mathbb{R}^{N_0})^2$, then

$$\|T(x+z) - T(x)\|_p \leqslant \theta\|z\|_p, \tag{2.20}$$

where $\theta \geqslant 0$ denotes the Lipschitz constant of the network. Different norms can also be used for the input and output spaces. Controlling this constant thus represents a feasible approach to limit the effect of adversarial attacks. Computing the exact Lipschitz constant of a neural network is however an NP-hard problem [16, 20], so the main challenge is to find efficient ways to approximate this constant effectively and to control it during the training phase without hindering the model performance. A recent study [56] shows how the Lipschitz behaviour of a network is influenced

by the architectural choices as well as the initialization values of the weights.

Computing the Lipschitz bound

Several studies have been devoted to the complex problem of estimating the Lipschitz constant of DNNs. Goodfellow *et al.* estimate the global Lipschitz constant by multiplying the constants of each individual layer in the network. (see [14] for more information). Although this method is simple and general, since it is valid for convolutional layers as well [57], the upper bound it produces is rough and usually over-pessimistic.

In [16], the authors introduce a generic technique known as AutoLip to compute the upper bound for the Lipschitz constant of any differentiable function, which can be applied for feed-forward neural networks. Moreover, the authors introduce a variant that is suitable for sequential neural networks, named SeqLip. Another study [58] introduces a convex programming framework capable of deriving tight Lipschitz bounds in the context of feed-forward neural networks. They take advantage of the fact that the usual non-linear operators used in neural network-based models are gradients of convex functions. Using this observation, they formulate the Lipschitz estimation problem in the form of a semi-definite program (SDP), referred to as LipSDP. Despite the fact that there exist quite effective SDP solvers, the approach itself is still very computationally demanding.

Combettes and Pesquet [20] derived several global Lipschitz bounds for forward neural networks, by making the assumption that for every layer $i \in \{1, \dots, m\}$, the associated activation function R_i is an α_i-averaged operator where $\alpha_i \in]0, 1]$. This means that there is a 1-Lipschitz (non-expansive) operator Q_i such that $R_i = (1 - \alpha_i)\,\mathrm{Id} + \alpha_i Q_i$. Then, the following holds, $(\forall (x, x') \in (\mathbb{R}^{N_i})^2)$,

$$|| R_i x - R_i x' ||_2^2 \leqslant || x - x' ||_2^2 - \frac{1 - \alpha_i}{\alpha_i} || (\mathrm{Id} - R_i)x - (\mathrm{Id} - R_i)x' ||_2^2 . \quad (2.21)$$

We can say that the stability of the operator R_i increases as α_i gets smaller. We recover some special cases: if $\alpha_i = 1$, then R_i is non-expansive and if $\alpha_i = 1/2$, R_i is firmly non-expansive. An example of a firmly non-expansive operator is the proximity operator of a function which is *proper*, *convex*, and *lower-semicontinuous* (l.s.c.). We recall below some definitions.

Let $f : \mathbb{R}^N \to]-\infty, +\infty[$.

(i) Function f is proper if

$$\mathrm{dom}(f) \neq \varnothing,$$

where $\mathrm{dom}(f) = \{x \in \mathbb{R}^N \mid f(x) < +\infty\}$.

(ii) Function f is lower-semicontinous (l.s.c.) if

$$(\forall x_0 \in \mathbb{R}^N) \quad \lim_{x \to x_0} \inf f(x) \geqslant f(x_0).$$

(iii) Function f is convex if the following holds for every $\alpha \in]0,1]$:

$$\big(\forall (x,x') \in (\mathrm{dom}(f))^2\big) \quad f(\alpha x + (1-\alpha)x') \leqslant \alpha f(x) + (1-\alpha) f(x').$$

The proximity operator was introduced by Moreau in 1965 [59], and can be viewed as an extension of the projection onto a convex set. The proximity operator of the afore-mentioned function f at a point $x \in \mathbb{R}^N$ is

$$(\forall x \in \mathbb{R}^N) \quad \mathrm{prox}_f(x) = \mathrm{argmin}_t \, f(t) + \frac{1}{2}\|t-x\|^2. \tag{2.22}$$

In [60] is shown that the activation operators commonly used in deep learning models, e.g. Sigmnoid, Tanh, ReLU and its variants (SeLU, ELU, Leaky ReLU) are proximity operators of some proper, convex, and l.s.c. functions. A fundamental result established in [20] states that, when such activation functions are used, tight Lipschitz bounds can be derived and these bounds are easy to compute when the weight matrices of the network are non-negative.

In the context of convolutional neural networks, a first naive method for estimating the Lipschitz bound has been proposed in [61], and later refined in [61], where a linear program is introduced to compute an upper bound.

Lipschitz-based defenses

Recently, several techniques to ensure the Lipschitz stability of neural networks have been explored. The work in [19] formalized the idea of ensuring robustness by controlling the Lipschitz constant of the network. This study shows that the network adversarial risk is directly related to its Lipschitz constant θ:

$$\mathop{\mathbb{E}}_{x \sim \mathcal{N}} [\max_{||\tilde{x}-x|| \leqslant \epsilon} \mid \mathcal{L}(T(\eta, \tilde{x}, y, T)) \mid] \leqslant \mathop{\mathbb{E}}_{x \sim \mathcal{N}} \mathcal{L}(\eta, x, y, T) + \lambda \theta \epsilon, \tag{2.23}$$

where λ is a positive constant quantifying the smoothness of the loss. The authors introduce Parseval networks, another approach for designing networks that are intrinsically robust to adversarial noise, by imposing the Lipschitz constant of each layer of the system to be less than 1. This approach was further extended for unsupervised learning in [62].

In another study, Miyato *et al.* [12] propose a novel weight spectral normalization technique applied to stabilize the training of the discriminator in Generative Adversarial Networks (GANs). The Lipschitz constant of the network is viewed as a hyper-parameter that can be tuned in the training process of the generator. By

doing so, the model exhibits improved generalization capabilities.

In [63] norm-constraint GroupSort-based architectures are proposed, and it is shown that they can be used as universal Lipschitz function approximators. The authors apply gradient norm preservation to create Lipschitzian networks that offer adversarial robustness guarantees.

Serrurier *et al.*[64] propose DeelLip, a mechanism to train 1-Lipschitz networks using a hinge regularized formulation of Kantorovich-Rubinstein loss. They use Björck normalization to control the Lipschitz constant of feed-forward layers. Another method that accounts for a local Lipschitz bound during the training phase is presented in [65].

Although training a neural network under Lipschitz constraints guarantees its robustness against possible attacks, it may affect its performance. Finding a good accuracy-robustness balance is a challenging task, which will be one of the main topics of this thesis.

Adversarial training – This is one of the most popular defenses against adversarial attacks. First introduced by Goodfellow *et al.* [14], it implies introducing adversarial examples in the training process to help the model improve generalization capabilities. The process implies augmenting the dataset by adding pairs of adversarial examples together with their associated true label $(\tilde{x}, y)$, to increase the model robustness against future adversarial samples.

In [14], the authors use a non-targeted version of FGSM attack (Eq. (2.14)) to create the adversarial samples $\tilde{x}$, which are later used to augment the training set. This strategy was later reformulated in [10], by introducing scaled adversarial training, which facilitates the integration of this methodology for larger datasets. The adversarial training algorithm includes an additional step in the training process, which augments the original batch of inputs with $n \in \mathbb{N}^*$ adversarial examples. This method has been proven efficient against FGSM attacks, but it is still vulnerable against more powerful iterative-based adversaries [32].

Another popular approach, introduced in [66] proposes PGD attack as an alternative to one-step FGSM for adversarial training. When T is trained on the most-adversarial cases in an ℓ_∞-ball around x, the optimizer actually minimizes the adversarial loss from Eq (2.9). A small adversarial loss ensures that the model is stable in all points around the ball $\mathcal{B}_\infty(x, \epsilon)$.

Concerning the training strategy, Madry *et al.* [66] propose to train the model only on adversarial examples, instead of concatenating the original dataset with

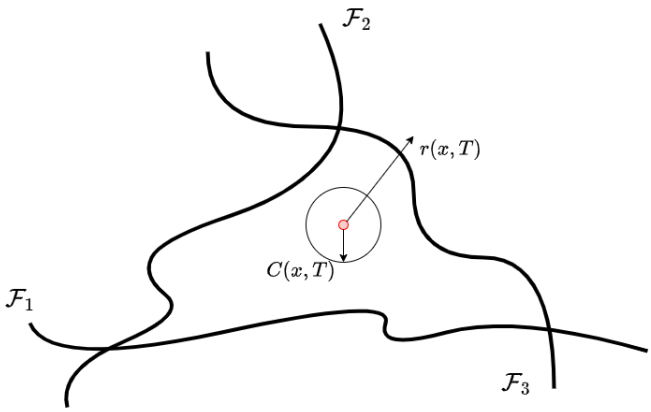


Figure 2.5: $\mathcal{F}_i$ are the decision boundaries. The certificate $C(x,F)$ lower bounds the minimal perturbation distance, guaranteeing the robustness of the model inside the ball.

some adversarial samples. This involves introducing an additional loop into the training process, thus increasing the training time. Due to this additional overhead, this method is not scalable to large datasets.

Tramer *et al.* introduced ensemble adversarial training [32], a technique that implies augmenting the training set with adversarials designed by using other pre-trained classifiers. In their study, they proved that this method is more time-efficient than the previous two strategies since it decouples the process of crafting adversarial samples from the actual training. Hence, this method is also suited for large datasets.

Lately, the techniques used for adversarial training were refined (see for example [67, 68, 69, 70]) and several optimizations techniques have been proposed to reduce the additional overhead during training. The main disadvantage of adversarial training is that, although it can be efficient against some adversaries, it remains quite an empirical strategy that does not offer any theoretical robustness guarantees [71].

Certifiable defences – To circumvent the disadvantage of adversarial training, methods that can offer some robustness guarantees were proposed. Carlini *et al.*[72] were the first to propose a Reluplex [73] algorithm to check the stability properties of neural network (NN) models. Considering a model T with ReLU activation functions, their method computes the minimal perturbation distance $r(x,T)$ w.r.t. the input x. Extending the Reluplex to the entire test data, it is possible to compute the percentage of safe examples against perturbations with norms less than r, thus, certifying the robustness of the model on x. Other studies refining this idea [74, 75] propose methods to assess trainable certificates $C(x,T)$

in order to ensure the robustness of the trained network.

For example, Hein *et al.* [75] propose a trainable certificate mechanism, which implies computing a lower bound of the minimal perturbation distance of F on x, i.e. $C(x,T) \leqslant r(x,T)$, guaranteeing the model robustness inside the $C(x,T)$ ball. A visual representation of this concept is depicted in Figure 2.5. Their method is based on the Cross-Lipschitz constant, resulting from the following maximization problem:

$$\max_{\epsilon} \min \left\{ \min_{i \neq k} \frac{T_k^*(x) - T_i^*(x)}{\max\limits_{\tilde{x} \in \mathcal{B}(x,\epsilon)} || \nabla T_k^*(\tilde{x}) - \nabla T_i^*(\tilde{x}) ||}, \epsilon \right\}, \tag{2.24}$$

where $T_k(x) = \max_i T_i(x)$.

Alternatively, works like [76, 77, 78, 79] propose different strategies to solve the same problem. They derive an upper bound $U(x,T)$ for the adversarial loss defined in Eq. (2.9), based on the margin loss.

$$\mathcal{L}(x) = \max_{\tilde{x}} \{ \max_{i \neq y} \quad T_i^*(\tilde{x}) - T_k^*(\tilde{x}) \} \quad \text{s.t.} \quad \tilde{x} \in \mathcal{B}(x,\epsilon). \tag{2.25}$$

The certificate $U(x,T)$ operates as follows. If the value of $U(x,T)$ is negative, it implies that the value of the adversarial loss is also negative, and thus, the model will always award the highest score to the true label y inside the region delimited by $\mathcal{B}(x,\epsilon)$. Wong *et al.*[76] solved Problem 2.25 as a linear programming problem, by training an alternative version of the neural network, while Raghunathan [77] solved the certificate via semi-definite programming (SDP). Due to the computational overhead, these early solutions only considered shallow neural networks, containing one hidden layer. In more recent works, Leiono *et al.* improved the scalability of these techniques, introducing Globally-Robust neural networks (GloRo-Nets) [78].

2.5.2 . Obfuscation

Since the strategies employed by adversarial attackers heavily rely on exploiting the gradient information of the classifier, the concept of hiding or obscuring this information has emerged as a defense mechanism to thwart adversarial attacks. These techniques are also known as *gradient masking* include *defensive distillation*, *shattered gradients*, and *randomized gradients*, briefly described in the following.

Defensive distillation – The concept was first introduced by Hinition *et al.* [80] and implies compressing the model's architecture, by training a reduced version of the network on the logits. The work was later refined in [41], which proposed a distillation technique that uses a temperature parameter τ for the final softmax operator, defined as

$$\text{softmax}(x,\tau) = \left(\frac{e^{x_i/\tau}}{\sum_j e^{x_j/\tau}} \right)_{1 \leqslant i \leqslant N_m}. \tag{2.26}$$

Although this technique proved to be effective against some attacks like FGSM and L-BFGS, it provides poor results against more sophisticated adversaries like C&W, DDN, and FMN.

Shattered gradients – Another strategy is to use some preprocessing module for the input data to ensure the robustness of the classifier [81, 82]. The main idea is to include a non-differentiable or non-smooth operator $f(\cdot)$, apply it to the dataset, and then train the model T on this new dataset. Since the trained model $T(f(\cdot))$ is usually not differentiable w.r.t x, it may make the adversary fail in crafting efficient adversarial samples.

Randomized gradients – Another way to confuse the adversary is to find a way to randomize the model T. This can be done by randomly omitting some neurons of the trained classifier, as proposed by [83], or by randomly resizing the input image and then introducing zero-padding, as suggested in [84].

Although obfuscation methods are usually easy to use and implement, they are still vulnerable to adversarial samples, as shown in [40], because their main objective is to *confuse* the adversaries, not to completely mitigate their effect.

2.5.3 . Adversarial example detection

Adversarial example detection techniques constitute another popular defense strategy, They strive to find if the input is benign or adversarial [85].

Auxiliary models – some efforts concentrate on the development of auxiliary models that are intended to differentiate between adversarial and benign instances. Grosse *et al.* [86] introduce an additional label $m+1$ to the classification problem, which accounts for adversarial examples. A variation of this method was proposed in [87, 88], where a binary adversarial detection model is trained and used before the classification module. The auxiliary model takes as input an intermediate feature representation of the original classifier. This binary model has the task to distinguish adversarial examples from genuine inputs.

Statistical models – statistics can be used to distinguish adversarial examples. The work in [89] claims that principal component analysis (PCA) is a useful tool for detecting adversarial inputs. The authors compute the covariance matrix C of the training date. Then, they perform a singular value decomposition (SVD) on $C = U\Sigma U^\top$. The PCA whitened input associated with an input x is computed as $\Sigma^{-1/2}U^\top x$. The resulting coefficients can be sorted in the ascending order of the diagonal components of Σ. It is shown that the coefficients for the later principal components associated with adversarial inputs show consistently a greater value than the ones of the clean inputs.

Other statistical detection techniques imply computing some measure to tell if two datasets were drawn from the same distribution. For example, Grosse *et al.*[86] used the maximum mean discrepancy (MMD) [90] on random groups of data, in an effort to detect possible adversarial attacks.

Consistency check – this direction of research concentrates on determining whether, given an input sample, the model prediction is feasible or not. In most cases, they adjust the values of the model parameters or modify the instances that are being fed into the classifier in order to see if this makes a substantial difference in the results. These are predicted on the assumption that the classifier will continue to make accurate predictions when applied to the original data after some changes [52]. For example, [91] randomizes the model, by employing dropout technique. If these classifiers come up with highly different prediction results following randomization, then this sample is quite likely to be an adversarial one. However, Carlini *et al.* [85] showed that their attack is capable of evading 10 distinct detection mechanisms, proving that the intrinsic properties of adversarial examples are not easy to spot and elude.

2.6 . Conclusion

This chapter has presented an overview of the state-of-the-art in the field of adversarial attacks and defenses of neural networks. The robustness of deep learning models is a hot topic that has attracted increasing attention from the research community, since it represents an important aspect to consider in the development and integration of future trust-worthy AI solutions in real-life applications. The next chapters will present new contributions in this domain.

Chapter 3 – EMG-based automatic gesture recognition using robust neural networks

This chapter introduces a novel approach for building a robust Automatic Gesture Recognition system based on Surface Electromyographic (sEMG) signals, acquired at the forearm level. Our main contribution is to propose new constrained learning strategies that ensure robustness against adversarial perturbations by controlling the Lipschitz constant of the classifier. We focus on nonnegative neural networks for which accurate Lipschitz bounds can be derived, and we propose different spectral norm constraints offering robustness guarantees from a theoretical viewpoint. Experimental results on four publicly available datasets highlight that a good trade-off in terms of accuracy and performance is achieved. We then demonstrate the robustness of our models, compared to standard trained classifiers in three scenarios, considering both white-box and black-box attacks.

3.1 . EMG and automatic gesture recognition

sEMG stands for surface electromyography and represents the electrical manifestation of the neuromuscular activation related to the contraction of the muscles [92]. This technology may be used by physically impaired persons to control rehabilitation and assisting devices. EMG is also used in many types of research domains, including those involved in biomechanics, motor control, neuromuscular physiology, movement disorders, postural control, and physical therapy [93].

The first recording of EMG activity was made by Marey in 1890, who introduced the term electromyography. Clinical use of surface EMG for the treatment of different disorders began in the 1960s. Hardyck was the first practitioner who used EMG [94]. Cram and Steger introduced a clinical method for scanning a variety of muscles, in 1980, using an EMG sensing device [95]. Progress in understanding these signals has been made during the past 15 years. Still, there are some limitations in characterizing the properties of surface EMG signals (estimation of the phase, acquiring exact information) due to derivation from normality. Traditional system reconstruction algorithms have various limitations and considerable computational complexity, and many show high variance [93].

The work carried out by researchers focused on sEMG signals resulted in developing better algorithms, upgrading existing methodologies, improving detection techniques to reduce noise, and acquiring accurate EMG signals. This section will

further present the performance of various technologies developed with sEMG signals.

3.1.1 . Challenges and limitations

In recent years, the concept of human-computer interaction (HCI) has been at the core of many scientific and sociological developments. Combined with the power of machine learning algorithms, it has led to some of the most outstanding achievements in nowadays technology which are used successfully in an ever-increasing number of areas impacting our lives, e.g. medicine [3], autonomous driving [4], natural language processing [5], etc. Researchers all around the world focus on providing new intuitive and accurate ways of interacting with devices around, based on gesture, voice, or vision analysis [26]. Gestures constitute a universal and intuitive way of communication, with the potential of bringing the Internet of Things (IoT) experience to a different, more organic level [6]. Automatic gesture recognition (AGR) algorithms can be successfully used in various applications, from sign language recognition (SLR) [96] to Virtual Reality (VR) games [97].

Various solutions for AGRs based on image or video stream analysis, leveraging computer vision algorithms, have been proposed; see for example [98, 99, 100]. A multi-stream solution for dynamic hand-gesture recognition is described in [101]. Multi-modal approaches for gesture classification have also been studied [102]. A novel method showing a fully neuromorphic implementation [103] achieves good results (96% accuracy while reducing the inference time by 30%). Although a good performance is achieved on synthetic data, in real-life scenarios these systems may be sensitive to environmental conditions, e.g. light conditions, background, etc. Additionally, these systems are often computationally demanding and consequently not always suited for real-time applications. Accelerometers and electromyography (EMG) sensors provide an alternative low-cost technology for gesture sensing [104]. In [105] the authors propose a method combining feature selection with ensemble learning, achieving around 78% classification accuracy for 52 gestures. The applications of sEMG-based classification systems are focused on, but are not limited to, assertive devices and rehabilitation or postural control therapy for physically impaired persons [106]. With the continuous development of more versatile signal processing techniques, the applications of EMG signal classification expanded to a wide range of domains including augmented reality, the gaming industry, military applications, etc.[107, 108].

Two critical issues need to be addressed when developing AGR algorithms: fast enough inference to ensure real-time feeling for the end-user, and accurate and robust classification to guarantee that the gesture is correctly identified no matter the environmental conditions. Machine learning methods have become the main tools for AGR systems, on account of their ability to solve a great variety of problems, from simple regressions to complex multi-modal classification.

However, as emphasized in Chapter 2, deep neural networks, which are probably the most powerful methods, may appear as black boxes whose robustness is not always well-controlled. For real-life applications, it is mandatory to guarantee the reliability of such techniques. Nowadays, the main difficulty to overcome consists in developing high-performance systems that are also trustable and safe. An additional challenge is to avoid implementation heaviness during the learning phase.

It must be emphasized that adversarial inputs are not necessarily artificially created with the intention to sabotage the system. As other physiological signals, e.g. EEG or EKG, EMG signals have low frequency components (usually between 10 – 150Hz), and low amplitudes ($\leqslant 10$ mV Peak to Peak). This makes them very sensitive to noise and outside perturbations that can occur innately, in the form of noise stemming from acquisition devices, imperfect sensor contact, etc. Those can seriously flaw the performance of real-life applications based on pre-trained models [15].

The Lipschitz behaviour of the network is intimately connected with its resilience against adversarial attacks, as was mentioned in Section 2.5. For a given metric, the Lipschitz constant enables an upper bound to be placed on the output perturbation. Consequently, exercising control over this constant is a workable answer to the problem of reducing the impact of perturbations.

3.2 . Robustness solutions in the context of non-negative neural networks

3.2.1 . Problem formulation

Consider the Model 2.2.1 introduced in the previous chapter. Even though the choice of the activation R_i may differ depending on the task at hand, it has been shown in [20, 60] that most of them are actually α_i-averaged operators with $\alpha_i \in]0, 1]$. Recall that R_i is an α_i-averaged operator if, for every pair $(x_i, y_i) \in (\mathbb{R}^{N_i})^2$, the following inequality holds:

$$\|R_i(x_i) - R_i(y_i) - (1 - \alpha_i)(x_i - y_i)\| \leqslant \alpha_i \|x_i - y_i\|. \tag{3.1}$$

When $\alpha_i = 1/2$, R_i is said to be firmly nonexpansive. For standard choices of activation operators, R_i is firmly nonexpansive since it is the proximity operator of a proper, lower-semicontinuous function (see [60] for more details). Note that, in [58], it is assumed that R_i operates component-wise and is slope-bounded. The authors emphasize that the most common case corresponds to lower and upper slope values equal to 0 and 1, respectively. It follows from [109, Proposition 2.4] that a function satisfies this property if and only if it is the proximity operator of some proper lower-semicontinuous convex function so that similar assumptions to

those made in [60] are recovered.

As explained in [20], examples of activation operators R_i which are α_i-averaged with $\alpha_i > 1/2$ can be encountered. They basically correspond to over-relaxations of firmly nonexpansive operator. An example of such operator is the Swish activation function [110]. Another famous example is the group-sort operator:

$$\left(\forall x_i = \begin{bmatrix} x_{i,1} \\ \vdots \\ x_{i,M} \end{bmatrix} \in \mathbb{R}^{N_i} \right) \qquad R_i(x_i) = \begin{bmatrix} x_{i,1}^{\uparrow} \\ \vdots \\ x_{i,M}^{\uparrow} \end{bmatrix}, \tag{3.2}$$

where the vector x_i has been decomposed in M subvectors $x_{i,j}$ with $j \in \{1, \dots, M\}$, of dimension B ($N_i = BM$) and $x_{i,j}^{\uparrow}$ designates the vector of components of $x_{i,j}$ sorted in ascending order. R_i is then purely nonexpansive, i.e. $\alpha_i = 1$. Note that max-pooling can be achieved by composing this group sort operation with a linear operator. Indeed, if $i < m$, $M = N_{i+1}$, and W_{i+1} is the matrix extracted from the $N_i \times N_i$ identity matrix Id_{N_i} by selecting the matrix rows with indices multiple of B, then $W_{i+1} \circ R_i$ corresponds to a max-pooling.

3.2.2 . Lipschitz robustness certificate

Consider a neural network T as described in Figure 2.2. let $x \in \mathbb{R}^{N_0}$ be the input of the network and let $T(x) \in \mathbb{R}^{N_m}$ be its associated output. By adding some small perturbation $z \in \mathbb{R}^0$ to the input, the perturbed input is

$$\tilde{x} = x + z.$$

The effect of the perturbation on the output of the system can be quantified by the following inequality:

$$\|T(\tilde{x}) - T(x)\| \leqslant \theta_m \|z\|, \tag{3.3}$$

where $\theta_m \geqslant 0$ denotes a Lipschitz constant of the network. θ_m represents thus an important parameter that allows us to assess and control the sensitivity of a neural network to various perturbations. It needs however to be accurately estimated to provide valuable information. A standard approximation to the Lipschitz constant [14] is given by

$$\theta_m = \prod_{i=1}^{m} \|W_i\|_{\mathrm{S}}, \tag{3.4}$$

where $\|\cdot\|_{\mathrm{S}}$ denotes the *spectral norm* of a matrix. Although simple to compute, this approximate bound is over-pessimistic. Different methods for obtaining tighter estimates of the Lipschitz constant have been presented in the recent literature; see for example [16, 20, 58, 111, 112]. Local estimates of the Lipschitz constant can

also be performed, which may appear more relevant. But they are more complex to compute and, as we will see, controlling the global Lipschitz constant is usually sufficient to get a good performance. Estimating the global Lipschitz constant of the network is an NP (non-deterministic polynomial-time)-hard problem [16]. Although there exist efficient approaches to approximate an accurate bound [58, 111, 112], computing these estimates may be expensive for wide or deep networks. In addition, using these bounds within a training procedure is a difficult task [113].

In this work, we will make the following assumption.

Assumption 3.2.1 Let a neural network be given by (2.1) where the i-th layer with $i \in \{1, \dots, m\}$ is given by (2.2). We assume that

(i) all the activation layers, except possibly the last one, consist of separable averaged operators, that is, for every $i \in \{1, \dots, m-1\}$, there exist averaged functions $(\rho_{i,k})_{1 \leqslant k \leqslant N_i}$ from $\mathbb{R}$ to $\mathbb{R}$ such that $R_i \colon (\xi_{i,k})_{1 \leqslant k \leqslant N_i} \mapsto (\rho_{i,k}(\xi_{i,k}))_{1 \leqslant i \leqslant k}$;

(ii) at the last activation layer, R_m is an averaged operator.

Our approach will be grounded on the following result.

Proposition 3.2.2 [20] *Suppose that Assumption 3.2.1 holds. For every $i \in \{1, \dots, m\}$, let A_i be the matrix whose elements are the absolute values of those of W_i. Then,*

$$\vartheta_m = \|A_m \cdots A_1\|_{\mathrm{S}} \tag{3.5}$$

is a Lipschitz constant of T. In addition

$$\|W_m \cdots W_1\| \leqslant \vartheta_m. \tag{3.6}$$

In particular if, for every $i \in \{1, \dots, m\}$, $W_i \in [0, +\infty[^{N_i \times N_{i-1}}$, ϑ_m is equal to the lower bound in (3.6).

Based on this proposition, the best estimate for the Lipschitz constant of a given feedforward neural network having nonnegative weights simplifies to the spectral norm of the product of all the weight matrices composing the network. More precisely, the obtained Lipschitz constant

$$\vartheta_m = \|W_m \cdots W_1\|_{\mathrm{S}}$$

is the Lipschitz constant of a purely linear network, where all the non-linear activation operators have been replaced with the identity operator.

The above result is guaranteed to be valid only in the case when all the weights are nonnegative. In the general case of networks with weights having arbitrary

signs, it can be proved that $\|W_m \cdots W_1\|_{\mathrm{S}}$ represents only a lower bound of the Lipschitz constant established in [20]. It is also worth mentioning that the proposed results hold for any algebraic structure of the weight matrices $(W_i)_{1\leqslant i\leqslant m}$. Using the above-defined bound, in the following we will propose an algorithm for training models with theoretical robustness guarantees, and validate it in the context of gesture classification. By focusing on gesture recognition, we aim to showcase the effectiveness of our methodology in a challenging domain having multiple applications and for which real experiments can be made. Moreover, gesture recognition tasks often involve complex and dynamic data, making them suitable testbeds for evaluating the robustness and adaptability of our proposed approach.

3.3 . Optimization methods for training robust feed-forward neural networks

Standard training in neural networks consists in the minimization of a nonconvex cost function with respect to the model parameters by means of an iterative strategy. Let $\mathcal{L}$ be the cost function defined as follows:

$$\mathcal{L}(\eta) = \sum_{k=1}^{K} \ell(z_k, \eta), \tag{3.7}$$

where $\eta = (\eta_i)_{1\leqslant i\leqslant m}$ is a vector encompassing all the model parameters. For each layer $i \in \{1, \ldots, m\}$, η_i denotes a vector of dimension $N_i(N_{i-1}+1)$ that contains the scalar variables associated with the weight matrices W_i and the corresponding bias components b_i. The data information is represented by $(z_k)_{1\leqslant k\leqslant K}$. For every $k \in \{1, \ldots, K\}$, z_k is a pair consisting of an input of the system and the associated desired output (ground truth). Also, ℓ represents the loss function assumed to be differentiable (almost everywhere) with respect to η.

To ensure robustness, we shall impose spectral norm constraints on the weight matrices. In other words, the vector of parameters η is constrained to belong to a closed set $\mathcal{S}$ that will be described in the next section. We propose to use an extension of a standard optimization technique for training neural networks [114]. More specifically, we will implement a *projected stochastic gradient* algorithm. A momentum parameter is introduced in this algorithm to accelerate the convergence process.

Algorithm 1 describes the iterations performed at each epoch $n > 0$. We see that there are two nested loops: the outer loop operates on the batch index q and the second one on the layer index i. In this algorithm, $\gamma_n \in]0, +\infty[$ is the learning rate, while $\zeta_n \in [0, +\infty[$ denotes the inertia parameter for momentum. The algorithm is very similar to block-iterative techniques used in convex optimization [114]. The parameters of each layer are indeed updated successively by performing a gradient step on the data in the current mini-batch (which can be epoch-dependent).

Algorithm 1: Projected SGD Algorithm

Partition $\{1,\dots,K\}$ *into mini-batches* $(\mathbb{L}_{q,n})_{1\leqslant q\leqslant Q}$
foreach $q \in \{1,\dots,Q\}$ **do**
  foreach $i \in \{1,\dots,m\}$ **do**
    $\Delta_{i,n} = (1+\zeta_n)\eta_{i,n} - \zeta_n\eta_{i,n-1}$
    $\widetilde{\eta}_{i,n} = [(\eta_{j,n+1}^\top)_{j<i} \;\; \Delta_{i,n}^\top \;\; (\eta_{j,n}^\top)_{j>i}]^\top$
    $\eta_{i,n+1} = \mathrm{P}_{\mathcal{S}_{i,n}}\Big(\Delta_{i,n} - \gamma_n \sum_{k\in\mathbb{L}_{q,n}} \nabla_i \ell(z_k, \widetilde{\eta}_{i,n})\Big)$

where $\mathcal{S}_{i,n} = \big\{\eta_i \mid [(\eta_{j,n+1}^\top)_{j<i} \;\; \eta_i^\top \;\; (\eta_{j,n}^\top)_{j>i}]^\top \in \mathcal{S}\big\}$.

∇_i represents the gradient, computed by standard backpropagation mechanism, with respect to η_i for each $i \in \{1,\dots,m\}$. This stochastic gradient step is followed by a projection $\mathrm{P}_{\mathcal{S}_{i,n}}$ onto the constraint set $\mathcal{S}_{i,n}$. The definition of this set as well as the way of handling this projection are detailed in the following.

3.3.1 . Constraints sets

As mentioned before, this thesis revolves around feed-forward networks with positive weights. Thus, the first condition that we impose is nonnegativity for each layer $i \in \{1,\dots,m\}$, which is modelled by the constraint set

$$\mathcal{D}_i = \{W_i \in \mathbb{R}^{N_i \times N_{i-1}} \mid W_i \geqslant 0\} \tag{3.8}$$

Moreover, based on our standing assumptions and Proposition 3.2.2, we must impose a spectral norm constraint on the weight matrices to control the robustness of the system. This translates mathematically as the following upper bound constraint:

$$\|W_m \cdots W_1\|_{\mathrm{S}} \leqslant \overline{\vartheta}, \tag{3.9}$$

where $\overline{\vartheta}$ represents the target maximum Lipschitz constant of the network. This bound constitutes a direct measure of the system's level of robustness against adversarial inputs. We need to handle these two constraints simultaneously during the training process. Imposing nonnegativity is fairly easy since (4.2) defines a simple convex constraint. By contrast, constraint (3.9) does not satisfy the convexity property. Since (3.9) corresponds to a closed set in the underlying space of weight matrices and this set has a nonempty intersection with $\mathcal{D}_1 \times \cdots \times \mathcal{D}_m$, the projection onto the intersection of the two sets can be defined but it is not guaranteed to be unique. To circumvent this difficulty, it can be noticed that (3.9) actually defines a multi-convex constraint in the sense that if, for every $i \in \{1,\dots,m\}$, $(W_j)_{1\leqslant j\leqslant m, j\neq i}$ are given, then (3.9) imposes a convex constraint on W_i. This suggests introducing the following closed and convex set:

$$\mathcal{C}_{i,n} = \{W_i \in \mathbb{R}^{N_i \times N_{i-1}} \mid \|A_{i,n} W_i B_{i,n}\|_{\mathrm{S}} \leqslant \overline{\vartheta}\} \tag{3.10}$$

in order to control the Lipschitz constant. Hereabove, the matrices $A_{i,n}$ and $B_{i,n}$ represent the product of the weight matrices for the previous and the posterior layers, respectively. By adopting the convention that $A_{i,n} = \mathrm{Id}$ if $i = m$ and $B_{i,n} = \mathrm{Id}$ if $i = 1$, we define these matrix products as

$$A_{i,n} = W_{m,n} \cdots W_{i+1,n}, \qquad B_{i,n} = W_{i-1,n+1} \cdots W_{1,n+1}, \tag{3.11}$$

where $(W_{j,n})_{1 \leqslant j \leqslant m}$ denote the estimates of the weight matrices at each iteration n, as it appears in Algorithm 1.

Thus, our objective will be to perform the projection onto the set $\mathcal{S}_{i,n} = \mathcal{D}_i \cap \mathcal{C}_{i,n}$, for each layer $i \in \{1, \ldots, m\}$ and at each iteration n. Several algorithms can be envisaged to solve this convex optimization problem.

Before describing our proposed algorithmic solution, let us recall the expressions of the required elementary projections. For every $W \in \mathbb{R}^{S \times T}$, the projection of W onto $[0, +\infty[^{S \times T}$ is

$$\mathsf{P}_{[0,+\infty[^{S \times T}}(W) = (\widetilde{W}_{s,t})_{1 \leqslant s \leqslant S, 1 \leqslant t \leqslant T}, \tag{3.12}$$

where, for every $s \in \{1, \ldots, S\}$ and $t \in \{1, \ldots, T\}$,

$$\widetilde{W}_{s,t} = \begin{cases} W_{s,t} & \text{if } W_{s,t} \geqslant 0 \\ 0 & \text{otherwise.} \end{cases} \tag{3.13}$$

Let $\mathcal{B}(0, \overline{\vartheta})$ be the closed spectral ball of centre 0 and radius $\overline{\vartheta}$ defined as[1]

$$\mathcal{B}(0, \overline{\vartheta}) = \{W \in \mathbb{R}^{S \times T} \mid \|W\|_{\mathrm{S}} \leqslant \overline{\vartheta}\}. \tag{3.14}$$

For every $W = (W_{s,t})_{1 \leqslant s \leqslant S, 1 \leqslant t \leqslant T} \in \mathbb{R}^{S \times T}$, let $U \Lambda V^\top$ be the singular value decomposition of W, where $U \in \mathbb{R}^{S \times R}$ and $V \in \mathbb{R}^{T \times R}$ are matrices such that $U^\top U = \mathrm{Id}$ and $V^\top V = \mathrm{Id}$, $R = \min\{S, T\}$, and $\Lambda = \mathrm{Diag}(\lambda_1, \ldots, \lambda_R)$, $(\lambda_r)_{1 \leqslant r \leqslant R} \in [0, +\infty[^R$ being the singular values of W. Then the projection of W onto $\mathcal{B}(0, \overline{\vartheta})$ is expressed as

$$\mathsf{P}_{\mathcal{B}(0,\overline{\vartheta})}(W) = U \widetilde{\Lambda} V^\top \tag{3.15}$$

where $\widetilde{\Lambda} = \mathrm{Diag}(\widetilde{\lambda}_1, \ldots, \widetilde{\lambda}_r)$ and

$$(\forall i \in \{1, \ldots, r\}) \quad \widetilde{\lambda}_i = \begin{cases} \lambda_i & \text{if } \lambda_i \leqslant \overline{\vartheta} \\ \overline{\vartheta} & \text{otherwise.} \end{cases} \tag{3.16}$$

To compute the projection onto $\mathcal{S}_{i,n}$ of a matrix $\overline{W}_i \in \mathbb{R}^{N_i \times N_{i-1}}$, we propose to employ the accelerated iterative dual forward-backward method in Algorithm 2. This algorithm is based on a dual proximal approach [115] and constitutes an extension of the optimization method originally proposed in [116]. The rationale for this algorithm is given in Appendix 3.7.

[1] To simplify our notation, $\mathcal{B}(0, \overline{\vartheta})$ will designate any spectral ball of this kind whatever the dimensions of the involved matrices.

Algorithm 2: FISTA-like accelerated version of DFB algorithm

Let $Y_0 \in \mathbb{R}^{N_m \times N_0}$
Set $\gamma = 1/(\|A_{i,n}\|_{\mathrm{S}} \|B_{i,n}\|_{\mathrm{S}})^2$
Set $\alpha \in]2, +\infty[$
for $l = 0, 1, \ldots$ **do**
$\eta_l = \frac{l}{l+1+\alpha}$
$Z_l = Y_l + \eta_l (Y_l - Y_{l-1})$
$V_l = \mathsf{P}_{D_i}(\overline{W}_i - A_{i,n}^\top Z_l B_{i,n}^\top)$
$\widetilde{Y}_l = Z_l + \gamma A_{i,n} V_l B_{i,n}$
$Y_{l+1} = \widetilde{Y}_l - \gamma \mathsf{P}_{\mathcal{B}(0,\overline{\vartheta})}(\gamma^{-1} \widetilde{Y}_l)$
return V_l

3.3.2 . Handling looser constraints

The Lipschitz constant of the network can be controlled in multiple ways. Besides the solution formulated in Section 3.3.1, a more standard approach to control it [9] consists in imposing

$$\prod_{i=1}^{m} \|W_i\|_{\mathrm{S}} \leqslant \overline{\vartheta}. \tag{3.17}$$

Two strategies have been implemented to enforce this constraint.

(i) The first one consists in imposing a uniform bound on the spectral norm of each weight matrix $(W_i)_{1 \leqslant i \leqslant m}$, which leads to the following convex constraint sets:

$$(\forall i \in \{1, \ldots, m\}) \quad \widetilde{\mathcal{C}}_i = \{W_i \in \mathbb{R}^{N_i \times N_{i-1}} \mid \|W_i\|_{\mathrm{S}} \leqslant \overline{\vartheta}^{1/m}\}. \tag{3.18}$$

(ii) The second strategy aims at introducing more flexible bounds on the spectral norms of each layer. It is based on the following choice for the individual convex constraint sets:

$$(\forall n \in \mathbb{N} \setminus \{0\})(\forall i \in \{1, \ldots, m\}) \quad \check{\mathcal{C}}_{i,n} = \Big\{W_i \in \mathbb{R}^{N_i \times N_{i-1}} \mid \|W_i\|_{\mathrm{S}} \leqslant \|W_{i,n}\|_{\mathrm{S}} \Big(\frac{\vartheta}{\prod_{j=1}^{m} \|W_{j,n}\|_{\mathrm{S}}}\Big)^{1/m}\Big\}. \tag{3.19}$$

For every $i \in \{1, \ldots, m\}$, projecting onto $\widetilde{\mathcal{C}}_i$ or $\check{\mathcal{C}}_{i,n}$ is performed by truncating a singular value decomposition, similar to the technique described at the end of Section 3.3.1. The projections onto $\widetilde{\mathcal{C}}_i \cap \mathcal{D}_i$ and $\check{\mathcal{C}}_{i,n} \cap \mathcal{D}_i$ can then be computed

by using the same iterative method as in Algorithm 2 with $A_{i,n} = B_{i,n} = \mathrm{Id}$.

In all the proposed constrained optimization methods, the projection $\mathsf{P}_{\mathcal{B}(0,\widetilde{\vartheta})}$ onto a spectral ball with radius $\widetilde{\vartheta} > 0$ plays a prominent role. The ball radius depends on the handled constraint (3.9), (3.18), or (3.19). A complex operation such as a singular value decomposition may be very demanding in terms of computational resources when dealing with large-size matrices. In that case, we propose to use an approximate projection [12] defined as

$$(\forall W \in \mathbb{R}^{S\times T}) \quad \mathsf{P}_{\mathcal{B}(0,\widetilde{\vartheta})}(W) \simeq \begin{cases} W & \text{if } \|W\|_{\mathrm{S}} \leqslant \widetilde{\vartheta} \\ \dfrac{\widetilde{\vartheta}}{\|W\|_{\mathrm{S}}} W & \text{otherwise.} \end{cases} \tag{3.20}$$

Using this approximation in Algorithm 2 yields approximate projections $(\widetilde{\mathsf{P}}_{\mathcal{C}_{i,n}\cap\mathcal{D}_i})_{1\leqslant i\leqslant m, n>0}$. Note however that we then lose the theoretical guarantees of convergence Algorithm 2, even if this issue was not observed in our implementation.

An additional advantage of Formula (3.20) is that it allows the nonnegativity of the elements of the input matrix to be kept. This allows us to derive cheap approximate versions of the projection onto $\widetilde{\mathcal{C}}_i \cap \mathcal{D}_i$ with $i \in \{1, \ldots, m\}$ by first projecting onto $\mathcal{D}_i$ and then applying the approximate projection onto $\widetilde{\mathcal{C}}_i$. The resulting approximate projection is denoted by $(\widetilde{\mathsf{P}}_{\widetilde{\mathcal{C}}_i\cap\mathcal{D}_i})_{1\leqslant i\leqslant m}$. A similar procedure can be followed to compute approximate projections $(\widetilde{\mathsf{P}}_{\check{\mathcal{C}}_{i,n}\cap\mathcal{D}_i})_{1\leqslant i\leqslant m, n>0}$ onto $(\check{\mathcal{C}}_{i,n} \cap \mathcal{D}_i)_{1\leqslant i\leqslant m, n>0}$.

3.4 . AGR experimental setup

3.4.1 . sEMG datasets

We test our proposed training scheme on four online datasets containing EMG information on different hand gestures. The first three were acquired using Myo armband, a device developed by Thalmic Labs, equipped with eight sEMG sensors displayed circularly, while the last one was acquired using 10 active double-differential `OttoBockMy-oBock13E200` sEMG electrodes [2].

Myo-sEMG. The first dataset, detailed in [117] contains EMG signals characterizing 7 hand gestures correlated to the primary movements of the hand. There are four mobility gestures (i.e., wrist flexion and extension, ulnar, and radial deviation) and two gestures used for grasping and releasing objects (i.e., spread fingers and close fist). The 7$^{\text{th}}$ gesture characterizes the neutral position, corresponding to the

[2] https://www.ottobock.com/en-gb/home-uk

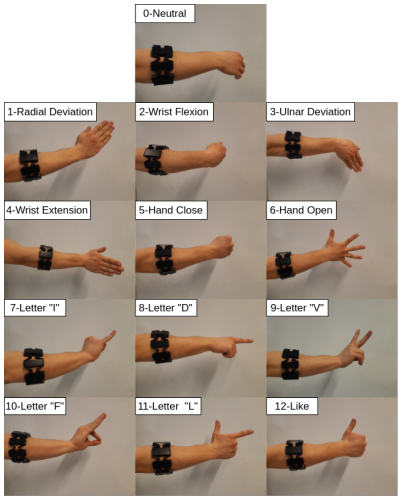


Figure 3.1: 13-gestures Dataset [2]

relaxation of the muscles.

13Myo-sEMG. The second dataset includes 13 gestures: the same 7 gestures described above, plus 6 additional classes. It contains gestures from 50 different subjects and two sets of trials per user. All 13 gestures are depicted in Figure 3.1. More details about the dataset can be found in [2].

NinaPro DB5.C. The third dataset is a subset of NinaPro DB5 dataset, detailed in [118]. The dataset is acquired using two Myo armbands, one positioned just below the elbow and the other one closer to the arm. For our experiments, we considered the subset C, which contains sEMG data associated with 24 gestures.

NinaPro DB1. The fourth dataset was introduced in [119], and encompasses physiological data acquired from 27 able-bodied subjects, performing a total of 53 different gestures. The sEMG data is recorded using 10 electrodes, positioned as follows. The first eight electrodes are evenly distributed around the forearm using an elastic band, maintaining a consistent distance from the radio-humeral joint located directly below the elbow. Two more electrodes are strategically positioned on the major flexor and extensor muscles located in the forearm.

We also validate our models in a real-context scenario. For the real-life predictions, we recorded the EMG activity associated with each gesture at the forearm level using the Myo armband. The information collected from each channel is transmitted to a computer via Bluetooth protocol, where it is processed to extract relevant time domain features that will be used by the classifier to determine which gesture has been performed.

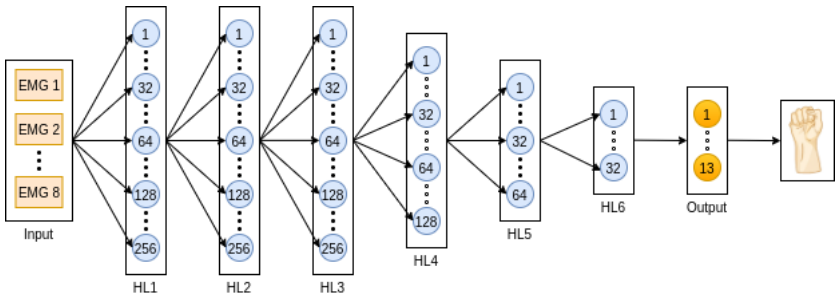


Figure 3.2: *Proposed neural network architecture for AGR. All the layers except the last one use ReLU activation functions; the last layer uses Softmax. The number of neurons considered for each layer is: 128, 128, 128, 64, 32, 16, in the case of the 7-gestures dataset, 256, 256, 256, 128, 64, 32 in the case of the 13-gesture dataset, and 512, 512, 256, 128, 64 in the case of the 24-gesture and the 53-gesture dataset. The last layer has 7, 13, 24, or 53 neurons representing the gesture number being recognized. Each EMG box represents a column vector containing* 8 *time descriptors.*

3.4.2 . Proposed Architecture

The raw 8 /10 channels EMG signal is split using a 250 ms sliding window, with 50% overlap. A 250 ms window is long enough to cover the most common gesture durations, ensuring that the essential temporal aspects of each gesture are captured within this window. Overlap ensures that important signal characteristics, such as abrupt changes or transient patterns, are not missed due to window boundaries. By using overlapping windows, the feature extraction process also becomes more robust, as multiple windows contribute to representing the same temporal information from the EMG signal. From each window of each channel, a series of 8 time descriptors are extracted. The information from all the channels is then concatenated, forming a 64 (80 for the fourth dataset)-dimensional vector. The 7-gestures dataset contains around 200k vector samples, the 13-gestures dataset has around 59k vector samples, the 24-gestures dataset has around 20k vector samples, and the 53-gestures dataset has 250k vector samples. Those are split into training, validation, and test sets at the user level according to the ratio: 70%, 20%, 10%. These vectors are fed to the network in mini-batches of size 2048. For our experiments, we used as the loss function ℓ the categorical cross-entropy, with a learning rate $\gamma = 10^{-3}$ and momentum parameter $\zeta = 0.02$. The considered architectures consist of a 6-hidden layer ($m = 6$) fully connected neural networks, with different parameters depending on the considered datasets, but the same core structure, as displayed in Figure 3.2. Let $x = (x_k)_{0 \leqslant k \leqslant K-1}$ be the vector of EMG samples acquired on a window from one channel. We considered some of the most relevant features to describe sEMG data, as follows.

(i) **Mean Absolute Value (MAV)** – represents the average muscle activation level within a specific time window. As different gestures involve varying

degrees of muscle activation, MAV can capture the overall muscle activity pattern, helping to distinguish between gestures with low and high muscle involvement.

$$\mathrm{MAV}(x) = \frac{1}{K} \sum_{k=0}^{K-1} |x_k|. \tag{3.21}$$

(ii) **Zero Crossing Rate (ZCR)** – indicates how frequently the EMG signal crosses zero within a time window. Rapid changes in muscle activation lead to higher ZCR values, making it relevant for identifying gestures involving quick and repetitive movements. A threshold $\alpha \geqslant 0$ is used in order to lessen the noise effect. This feature can be computed in an incremental manner, and it is defined as

$$\mathrm{ZCR}(x) = \Big|\big\{k \in \{1, \ldots, K-1\} \mid |x_k - \ x_{k-1}| \geqslant \alpha \text{ and } x_k x_{k-1} < 0\big\}\Big|. \tag{3.22}$$

(iii) **Waveform Length (WL)** – quantifies the amplitude variations within a time window. Longer WL values may correspond to gestures involving sustained muscle activity or complex patterns. It corresponds to the following total variation seminorm:

$$\mathrm{WL}(x) = \sum_{k=1}^{K-1} |x_k \ - x_{k-1}|. \tag{3.23}$$

(iv) **Slope Sign Changes (SSC)** – counts the number of times the slope of the EMG signal changes its sign within a window. It is effective in detecting abrupt changes in muscle activation, which is crucial for recognizing gestures with distinct start and stop points. It amounts to checking a condition on three consecutive samples x_k, x_{k-1}, x_{k+1} with $k \in \{2, \ldots, K-2\}$:

$$\mathrm{SSC}(x) = \Big|\{k \in \{2, \ldots, K-2\} \mid (x_k - \ x_{k-1})(x_k \ - \ x_{k+1}) \geqslant \alpha\}\Big|, \tag{3.24}$$

where the threshold $\alpha > 0$ is employed to reduce the influence of the noise.

(v) **Root Mean Square (RMS)** – provides information about the overall energy of the EMG signal within a time window. High energy levels may correspond to forceful gestures, while lower energy levels may indicate more subtle movements. RMS helps in recognizing gestures with varying intensity levels, and it is given by

$$\mathrm{RMS}(x) = \sqrt{\frac{1}{K} \sum_{k=0}^{K-1} x_k^2}\,. \tag{3.25}$$

Table 3.1: Comparison to other sEMG-based AGR systems

Myo-sEMG		13Myo-sEMG		NinaPro DB5 Ex. C		NinaPro DB 1	
Method	Acc.[%]	Method	Acc.[%]	Method	Acc.[%]	Method	Acc.[%]
7-DNN (ours)	99.67	13-DNN (ours)	99.31	24-DNN (ours)	86.20	53-DNN (ours)	88.50
DL-TL [117]	98.12	EMG-CNN [120]	99.28	EELM [105]	83.60	IRDC-Net[121]	89.82

(vi) **Hjorth parameters** – are a set of three features originally developed for characterizing electroencephalography signals and then successfully applied to sEMG signal recognition. The most relevant Hjorth activity parameter can be thought of as the integrated power spectrum and basically corresponds to the variance of the signal, calculated as follows:

$$\sigma^2(x) \;=\; \frac{1}{K}\sum_{k=0}^{K-1}(x_k - \;\mu(x))^2, \tag{3.26}$$

where $\mu(x)$ represents the mean value of the signal. The standard deviation and $\mathrm{RMS}(x)$ are equal when the mean of the signal is zero.

(vii) **Skewness** – measures the asymmetry of the EMG signal amplitude distribution within a time window. Positive skewness indicates a longer tail on the right side, while negative skewness indicates a longer tail on the left side, and can be useful in identifying gestures with asymmetric muscle activations.

$$\mathrm{Skew}(x) = \frac{1}{K}\sum_{k=0}^{K-1}\left(\frac{x_k - \;\mu(x)}{\sigma(x)}\right)^3. \tag{3.27}$$

(viii) **Integrated Square-root EMG (ISEMG)** – It provides a measure of the total muscular activity and is particularly useful for capturing the overall muscle involvement over time. iEMG is commonly used to quantify muscle fatigue and effort during movements. In the context of gesture recognition, iEMG can help differentiate between gestures with varying levels of sustained muscle activation and can be indicative of the gesture intensity and duration.

$$\mathrm{ISEMG}(x) = \sum_{k=0}^{K-1}\sqrt{|\;x_k\;|}. \tag{3.28}$$

3.4.3 . Performance analysis in terms of accuracy and robustness

Since in this case the weights are not guaranteed to be positive, the lower bound introduced in Proposition 3.2.2 does not constitute a valid Lipschitz constant. Computing the exact Lipschitz constant θ_m of the system is a very difficult task [20], but we can easily bound θ_m between the estimate given by (3.4) and the spectral norm of the product of all the weight matrices from the network. We found

Table 3.2: Performance and robustness results for 7-gestures dataset baseline models. The training time is computed for an epoch, with a batch size=2048. All models were trained for 1000 iterations. All experiments were performed using 2 $\times$ A100 40 Gb Nvidia GPUs.

Regularization method	Param.	Accuracy [%]			Lipschitz constant	Training time [ms]
		Train	Validation	Test		
None	–	99.98	79.63	80.35	5.3×10^{12}	150
Dropout	rate=0.1	98.31	97.76	97.66	5.1×10^{11}	160
	rate=0.15	98.00	97.44	97.34	5.6×10^{11}	160
	rate=0.2	97.55	97.03	96.98	6.1×10^{12}	162
Batch Norm.	–	99.96	99.65	99.78	6.3×10^{12}	160
$\ell_1$regularization	reg. factor=10^{-4}	99.28	97.35	97.59	7.2×10^{9}	135
	reg. factor=10^{-3}	95.87	95.53	95.48	9.2×10^{9}	162
	reg. factor=10^{-2}	84.35	84.24	83.34	5.8×10^{10}	160
$\ell_2$regularization	reg. factor=10^{-4}	99.71	98.36	98.02	5.7×10^{11}	160
	reg. factor=10^{-3}	98.66	97.99	97.25	3.2×10^{10}	160
	reg. factor=10^{-2}	91.97	91.86	91.78	5.5×10^{8}	160
Non-negativity	–	97.23	96.82	96.92	9.69×10^{10}	162

that the Lipschitz constant upper bound θ_m is greater than 10^{12} for all our baseline models. Also, while training our model, we faced the problem of overfitting, which is a challenging issue in the classification of physiological signals.

This suggests that despite the high performance of the classifiers, their robustness is poorly controlled, leaving the systems vulnerable to adversarial perturbations. A first step towards controlling the Lipschitz constant of the classification algorithm and implicitly its robustness is to impose the nonnegativity condition associated with the constraint $\mathcal{D}$. Training under such a nonnegativity constraint is shown to improve the network operation interpretability [21] and acts as a regularization, reducing overfitting. On the other hand, it can affect its approximation capability and potentially lead to a performance decay. To further study the effect of other regularization techniques from a dual performance-robustness perspective, we trained several models for 1000 iterations using common regularization methods, such as *Dropout*, ℓ_1/ℓ_2 *Regularization*, and *Batch Normalization*. Such comparisons were also featured in other works like [122]. The results for the 7-gesture dataset are summarized in Table 3.2. It can be observed that Batch Normalization is the most efficient technique from the accuracy viewpoint, but it comes with an increase in the overall Lipschitz constant of the classifier. Training the proposed system subject to the nonnegativity constraint ($\mathcal{D}$) results in an overall accuracy of 96.92%, 95.87%, 84.75%, and 85.65% for the case of 7, 13, 24, and 53 classes, respectively. The performance decay was balanced by an increase in the robustness, since the Lipschitz constant, computed as indicated in Proposition 3.2.2, equals $\theta_m = 9.69 \times 10^{10}$ for 7 classes, $\theta_m = 9.73 \times 10^{10}$ for 13 classes, $\theta_m = 1.03 \times 10^{11}$ for 24 classes, and $\theta_m = 8.4 \times 10^{10}$ for 53 classes. We observed that the accuracy

Table 3.3: Lipschitz constant obtained with various constrained optimization strategies for different accuracies

		Accuracy	75%	80%	85%	90%	95%
Lipschitz constant 7-gestures Myo-sEMG	$\widetilde{\mathcal{C}}_i \cap \mathcal{D}_i$	$\widetilde{\mathrm{P}}_{\widetilde{\mathcal{C}}_i \cap \mathcal{D}_i}$	19.5	37.5	68.3	3.5×10^4	3.5×10^8
		$\mathrm{P}_{\widetilde{\mathcal{C}}_i \cap \mathcal{D}_i}$	0.66	13.47	74.16	1.04×10^3	1.39×10^5
	$\check{\mathcal{C}}_{i,n} \cap \mathcal{D}_i$	$\widetilde{\mathrm{P}}_{\check{\mathcal{C}}_{i,n} \cap \mathcal{D}_i}$	0.71	1.84	3.42	6.87	11.60
		$\mathrm{P}_{\check{\mathcal{C}}_i \cap \mathcal{D}_i}$	0.70	1.35	3.41	6.79	11.20
	$\mathcal{C}_{i,n} \cap \mathcal{D}_i$	$\widetilde{\mathrm{P}}_{\mathcal{C}_{i,n} \cap \mathcal{D}_i}$	0.44	1.79	2.93	4.85	5.68
		$\mathrm{P}_{\mathcal{C}_{i,n} \cap \mathcal{D}_i}$	0.35	0.46	0.65	0.82	0.95
Lipschitz constant 13-gestures 13Myo-sEMG	$\widetilde{\mathcal{C}}_i \cap \mathcal{D}_i$	$\widetilde{\mathrm{P}}_{\widetilde{\mathcal{C}}_i \cap \mathcal{D}_i}$	20.2	41.8	145.2	2.2×10^5	1.21×10^{11}
		$\mathrm{P}_{\widetilde{\mathcal{C}}_i \cap \mathcal{D}_i}$	0.85	20.47	112.3	1.62×10^4	2.31×10^8
	$\check{\mathcal{C}}_{i,n} \cap \mathcal{D}_i$	$\widetilde{\mathrm{P}}_{\check{\mathcal{C}}_{i,n} \cap \mathcal{D}_i}$	0.84	2.08	4.23	7.54	12.02
		$\mathrm{P}_{\check{\mathcal{C}}_i \cap \mathcal{D}_i}$	0.81	2.01	4.12	7.50	11.92
	$\mathcal{C}_{i,n} \cap \mathcal{D}_i$	$\widetilde{\mathrm{P}}_{\mathcal{C}_{i,n} \cap \mathcal{D}_i}$	0.54	1.87	3.38	4.20	5.78
		$\mathrm{P}_{\mathcal{C}_{i,n} \cap \mathcal{D}_i}$	0.49	0.53	0.75	0.92	1.25
		Accuracy	65%	70%	75%	80%	85%
Lipschitz constant 24-gestures NinaPro DB5 Ex C.	$\widetilde{\mathcal{C}}_i \cap \mathcal{D}_i$	$\widetilde{\mathrm{P}}_{\widetilde{\mathcal{C}}_i \cap \mathcal{D}_i}$	25.13	57.16	188.26	2.5×10^6	2.14×10^{11}
		$\mathrm{P}_{\widetilde{\mathcal{C}}_i \cap \mathcal{D}_i}$	1.85	31.12	112.3	1.82×10^4	4.63×10^8
	$\check{\mathcal{C}}_{i,n} \cap \mathcal{D}_i$	$\widetilde{\mathrm{P}}_{\check{\mathcal{C}}_{i,n} \cap \mathcal{D}_i}$	1.74	2.41	6.02	10.17	20.14
		$\mathrm{P}_{\check{\mathcal{C}}_i \cap \mathcal{D}_i}$	1.57	2.18	5.94	10.58	19.69
	$\mathcal{C}_{i,n} \cap \mathcal{D}_i$	$\widetilde{\mathrm{P}}_{\mathcal{C}_{i,n} \cap \mathcal{D}_i}$	0.88	2.05	4.28	5.74	6.84
		$\mathrm{P}_{\mathcal{C}_{i,n} \cap \mathcal{D}_i}$	0.77	0.96	1.27	1.44	1.96
Lipschitz constant 53-gestures NinaPro DB 1	$\widetilde{\mathcal{C}}_i \cap \mathcal{D}_i$	$\widetilde{\mathrm{P}}_{\widetilde{\mathcal{C}}_i \cap \mathcal{D}_i}$	26.26	86.17	200.45	4.10×10^6	4.45×10^{11}
		$\mathrm{P}_{\widetilde{\mathcal{C}}_i \cap \mathcal{D}_i}$	2.60	50.12	163.14	2.8×10^4	2.9×10^9
	$\check{\mathcal{C}}_{i,n} \cap \mathcal{D}_i$	$\widetilde{\mathrm{P}}_{\check{\mathcal{C}}_{i,n} \cap \mathcal{D}_i}$	2.94	4.43	6.88	14.25	22.16
		$\mathrm{P}_{\check{\mathcal{C}}_i \cap \mathcal{D}_i}$	2.83	2.18	5.56	16.48	20.16
	$\mathcal{C}_{i,n} \cap \mathcal{D}_i$	$\widetilde{\mathrm{P}}_{\mathcal{C}_{i,n} \cap \mathcal{D}_i}$	1.22	1.80	6.83	7.40	8.23
		$\mathrm{P}_{\mathcal{C}_{i,n} \cap \mathcal{D}_i}$	1.56	2.08	2.53	2.74	3.88

reduction can be overcome by adding additional layers to the architecture. Indeed, we were able to obtain a similar accuracy to the baseline by adding an extra layer to the existing architecture and retraining both systems subject to $\mathcal{D}$, i.e. 98.68%, 97.21%, 85.12%, and 87.03% for the 7-gesture, 13-gesture, 24-gesture, and 53-gesture datasets, respectively. Furthermore, compared to the unconstrained models, we managed to maintain high performance while improving the robustness with respect to unconstrained training, i.e. $\theta_m = 1.02 \times 10^{11}$ for the 7-classes dataset, $\theta_m = 9.96 \times 10^{10}$ for the 13-classes dataset, $\theta_m = 4.24 \times 10^{11}$ for the 24-classes dataset, and $\theta_m = 3.15 \times 10^{11}$ for the 53-classes dataset. We can however conclude from these tests that imposing the nonnegativity of the weight coefficients is not sufficient to reach satisfactory robustness.

To further control the robustness of the systems, we have to manage the Lipschitz constant of the networks by training them under additional spectral norm constraints, as described by (3.9). Searching for the optimal accuracy robustness trade-off, we trained several models considering each of the four aforementioned con-

straints, namely $(\mathcal{C}_{i,n})_{1\leqslant i\leqslant m, n\in\mathbb{N}}$ in (3.10), $(\widetilde{\mathcal{C}}_i)_{1\leqslant i\leqslant m}$ in (3.18), and $(\check{\mathcal{C}}_{i,n})_{1\leqslant i\leqslant m, n\in\mathbb{N}}$ in (4.2).

By adjusting the upper bound $\overline{\vartheta}$, we were able to assess the effect of a robustness constraint on the overall performance of the neural network-based classifiers, and finally to achieve the optimal trade-off. All our models were trained using Algorithm 1 as the optimizer. The obtained results are summarized in Table 3.3. As expected, obtaining a good robustness-accuracy trade-off requires paying attention to the way we design our constrained networks. In all the cases, we show that using tight constraints during the training phase to approximate the Lipschitz bound improves the overall performance of the classifier, proving the generalization properties of our solution.

For comparison, for each of the proposed constraints, we also evaluated the use of an inexact projection, designated by $\widetilde{\mathrm{P}}$ (see Section 3.3.2). It can be observed that using an exact projection yields significantly better results. By combining tight constraints and exact projection techniques, we observe that the robustness of the network can be properly ensured while keeping a good accuracy in both cases. Indeed, we succeeded in ensuring a Lipschitz constant of around 1 for a 95% accuracy for the first two datasets. The observed loss in accuracy with respect to standard training is consistent with the "no free lunch theorem" [123].

Training neural networks subject to tight spectral norm constraints can be challenging, and the cost of obtaining a good performance is the training time. We used a learning rate scheduler strategy during training, reducing the learning rate by a factor of 2 if the performance does not improve for 1000 epochs. Figure 3.3 shows the training curves for both validation and training sets in the context of the unconstrained baseline model (yellow and green lines), and in the case of training a constrained version (red and blue lines) using the optimal projection $\mathrm{P}_{\mathcal{C}_{i,n}\cap\mathcal{D}_i}$, with $\overline{\vartheta}_m = 0.95$. Even though it requires more iterations, the constrained model is capable of reaching an accuracy comparable to the baseline, while providing a robustness certificate.

Since the training curves may show some slight variations, we measured the accuracy variations in two ways: by computing the classical *standard deviation* (std), and by employing *median absolute deviation* (mad). For a vector $(x_i)_{1\leqslant i\leqslant I}$, it is expressed as $\mathrm{MAD} = \mathrm{median}\left((|x_i - \zeta(x)|)_{1\leqslant i\leqslant I}\right)$, where $\zeta(x)$ represents the median of the vector components. From this quantity, we can derive an empirical estimate of the standard deviation by multiplying MAD with a factor equal to 1.4826. The latter estimate is known to be more robust to outliers for Gaussian distributed data, especially in the case of small populations. The results are summarized in Table 3.4. It can be observed that the empirical standard deviation is

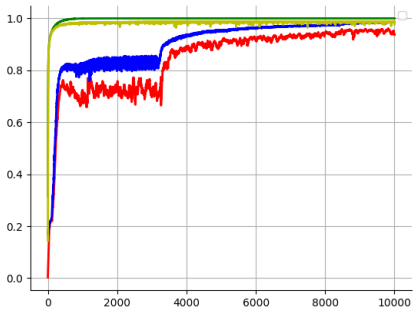


Figure 3.3: *Accuracy vs. Iterations – constrained and unconstrained models in the context of 7-gesture dataset. The training and validation curves are displayed in green and yellow, respectively, for the unconstrained model. The training and validation curves are displayed in blue and red, respectively, in the case of constrained training, with the bound* $\overline{\vartheta} = 0.95$.

below 1.6% and the robust estimate of it is below 1.1% for all four datasets. These deviation values are normal considering the size of the dataset and show that the presented results are relevant and consistent. Next, we have also evaluated how the positivity constraint impacts the overall accuracy of our system. We trained a robust network by allowing the weights to have arbitrary signs. For this purpose, we control individually the Lipschitz constant of each layer $i \in \{1, \dots, m\}$ to be less than a given value $\overline{\vartheta}^{1/m}$. The exact projection onto $\widetilde{\mathcal{C}}_i$, $\mathrm{P}_{\widetilde{\mathcal{C}}_i}$, as well as the approximate one $\widetilde{\mathrm{P}}_{\widetilde{\mathcal{C}}_i}$ were computed as described previously. In this case $\overline{\vartheta}$ represents an upper bound on the Lipschitz constant of the system. Table 3.5 summarizes the results for different values of $\overline{\vartheta}$, for two datasets. We compare our method for dealing with Lipschitz constraints with the approach proposed in [64]. This approach, which is implemented in the *deel-lip* library, allows the user to train robust networks in a convenient manner, offering a robustness certificate by performing a spectral normalization for each layer. It can be observed on these datasets that our method yields similar results when using the approximate projection, but better ones when using the exact projection. These results underline again the importance of carefully managing the projections and the effect it has on the accuracy of the system.

3.5 . Robustness validation

In this section, we investigate to what extent the theoretical concepts described in the previous sections help in improving the robustness of the classifier in different settings. To this goal, we consider the following three scenarios. In the first one, we examine the impact of adversarial attacks on the performance of the classifier. The second scenario takes into account the effect of noise in the acquisition process. In the case of sEMG signals, this noise may come from imperfect skin-sensor contact

Table 3.4: Standard deviation of accuracy computed on 15 epochs, after convergence, on the test set for constrained models.

	Accuracy	75%	80%	85%	90%	95%
7-gestures dataset (Myo-sEMG) Model Variation	empirical std	0.65	1.22	0.56	1.35	1.10
	robust std	1.02	0.94	0.53	0.87	1.07
13-gestures dataset (13Myo-sEMG) Model Variation	empirical std	0.65	1.05	0.75	0.75	0.72
	robust std	0.77	0.81	0.72	0.97	0.59
	Accuracy	65%	70%	75%	80%	85%
24-gestures dataset (NinaPro DB5.C) Model Variation	empirical std	0.68	0.95	0.87	0.77	0.76
	robust std	0.89	0.74	0.79	0.89	0.64
53-gestures dataset (NinaPro DB1) Model Variation	empirical std	0.72	0.93	0.95	0.87	0.78
	robust std	0.94	0.77	0.78	0.92	0.84

Table 3.5: Lipschitz constant for networks trained with arbitrary signs – 7-gestures / 13-gestures datasets.

	Accuracy	75%	80%	85%	90%	95%
7-gestures dataset Lipschitz constant	$\mathcal{C}_i$ $\widetilde{\mathrm{P}}_{\tilde{\mathcal{C}}_i}$	72.03	127.5	1296	8.75×10^4	5.43×10^9
	$\mathcal{C}_i$ $\mathrm{P}_{\tilde{\mathcal{C}}_i}$	52.06	102.49	905.45	7.23×10^4	8.14×10^8
	Deel-lip [64]	75.81	126.9	1283.6	8.70×10^4	5.43×10^9
13-gestures dataset Lipschitz constant	$\mathcal{C}_i$ $\widetilde{\mathrm{P}}_{\tilde{\mathcal{C}}_i}$	76.59	125.20	1016	2.03×10^4	4.3×10^8
	$\mathcal{C}_i$ $\mathrm{P}_{\tilde{\mathcal{C}}_i}$	61.22	99.74	740	1.26×10^4	6.7×10^7
	Deel-lip [64]	77.21	125.63	1120	2.04×10^4	4.5×10^8

caused by hairs or drops of sweat. In the last scenario, we perform a real-life experiment using 10 able-bodied volunteers.

3.5.1 . Sensitivity to adversarial attacks

We evaluate our robust model on purposely designed perturbations, by studying their influence on the overall performance of the system. We lead attacks on our best robust model in terms of accuracy and robustness, achieving 92.95% accuracy and a Lipschitz constant $\overline{\vartheta} = 0.87$ for the 7-gesture dataset. We compare the results with two conventionally trained models: the best one in terms of performance, which achieves 99.78% prediction accuracy on non-adversarial data, and another one trained to have similar performance as our robust model, reaching 92.99% accuracy on the original test set.

To create the adversarial samples we used some of the most popular white-box attackers, namely:

- *Fast gradient sign method (FGSM)* [14] – generates adversarial data based on the gradient of the cost function with respect to the input data;

Table 3.6: Adversarial attack results. The first four lines correspond to white-box attacks, whereas the last line shows a black-box attack. We consider out best constrained model, having a Lipschitz constant $\theta = 0.97$, two models trained conventionally: the best baseline and another one having similar performance as the constrained one. On the last columns we feature an adversarial trained model using PGD-generated perturbations.

	Accuracy [%]							
	robust model		baseline model				adv. trained model – PGD	
Attack	adv.	non-adv.	adv.	non-adv.	adv.	non-adv.	adv.	non-adv.
FGSM [14]	91.75	92.95	76.48	99.78	71.21	92.99	80.43	97.25
C&W ℓ_2 [40]	90.09		48.03		45.85		60.17	
PGD [66]	91.92		59.36		56.38		97.25	
JSMA [39]	91.10		89.37		81.27		83.31	
GM [124]	92.13		98.25		89.04		95.38	

- *Jacobian Saliency Map Attacker (JSMA)* [39] – computes a perturbation based on ℓ_2 distance metric by iteratively selecting the input sample that will increase the chance of miss-classification;
- *Projected gradient descent (PGD)* [66] – uses local first-order information about the network to create adversarial examples;
- *Carlini and Wagner (C&W)* [40] – utilizes ℓ_2 distance to compute the optimal adversarial perturbation.
- *Gradient Matching (GM) [124]* – this is a data-poisoning black-box attack. In this case, the attacker does not have access to the victim model parameters but instead is trying to match the gradient direction for adversarial examples.

We also show a comparison with another popular technique of ensuring the robustness of neural network-based models, namely *Adversarial training*. This implies training an extended version of the dataset, containing the original training data together with a perturbed version of the samples, in an effort to increase the system stability against adversarial inputs. Note that this method is purely empirical and gives no theoretical robustness certificates. We implemented an adversarial training strategy detailed in [66], training the system using an augmented version of the dataset which was updated every 25 epochs. The adversarial samples were created using PGD attack, and then the model was validated using data containing perturbations computed with various attacks.

The results summarized in Table 3.6 show the performance obtained for the 7-gesture test set. Note that the robust model performance is barely affected by the adversarial perturbations, whereas the baseline models show a huge drop in accuracy. It can be observed that adversarial training helps to increase the robustness, but

our method of controlling the Lipschitz constant of the network provides better results when facing data perturbed with other attackers than PGD. As expected, the poisoning attack is less effective than the white-box ones against the baseline models, but still, our robust model showcases better performance. This shows that our method is more versatile since its performance remains stable whatever the attacker.

3.5.2 . Noisy input behaviour

To simulate the effect of underlying noise generated during the acquisition process, we added synthetic noise directly to the raw sEMG data, prior to the feature extraction step. The noise is chosen independent and identically distributed according to a Gaussian mixture law $(1-p)\mathcal{N}(0,\sigma_0^2)+p\mathcal{N}(0,\sigma_1^2)$. The mixture comprises a background component, corresponding to the intrinsic electronic noise in the armband, such as thermal or quantization noise, and an impulsive component accounting for outliers. Those may be related to imperfect wiring that can generate impulse-like artefacts. In our experiments, we consider background and impulse noises with standard deviations $\sigma_0 = \alpha$ and $\sigma_1 = 10\alpha$ with $\alpha \in [0, +\infty[$. We generate different levels of noise, by varying the parameter α. The probability of peaks $p \in [0, 1]$ is also adjusted to simulate more or less severe scenarios in terms of outliers.

From the resulting noisy signals, we extract the features described in Section 3.4 and pass them to the classifier, using our robust models reaching an accuracy of 92.95% ($\overline{\vartheta} = 0.87$) for the 7-gestures dataset, and 93.05% ($\overline{\vartheta} = 0.98$) in the case of the 13-gestures dataset, trained with non-altered data. We compared the results achieved with our robust training with those obtained with i) classical training and ii) adversarial training. In this case, the adversarial training was performed by generating an extended dataset, containing the original data and corrupted versions of them by additive noise following the Gaussian mixture law described above, where the parameters p and α were drawn randomly in a uniform manner on $[0.15, 0.45]$ and $[0, 2]$, respectively. In the absence of noise, a similar performance in terms of accuracy was obtained: 7-gestures dataset – 92.99%, and 92.97%, 13-gestures dataset – 93.03% and 92.98% for baseline and adversarial training, respectively.

The experimental results obtained on two datasets are depicted in Figure 3.4. The red, blue, and green lines correspond to the unconstrained, constrained, and adversarial models, respectively. We observe that the constrained model is significantly less affected by the presence of noise in the inputs than the one trained without robustness guarantees. It is also worth noting that training with adversarial inputs also leads to satisfactory results, although usually slightly less accurate. The Lipschitz lower and upper bounds computed for the networks trained in an adversarial manner are indeed much lower than those with standard training, but they remain quite large ((1845.23, 79534.2) for 7-gestures dataset and (1754.74, 64595.8) for 13-gestures dataset).

Table 3.7: Real-scenario experiment results

Movement	User #1		User#2		User#3		User#4		User#5		User#6		User#7		User#8		User#9		User#10	
	C	U	C	U	C	U	C	U	C	U	C	U	C	U	C	U	C	U	C	U
up	2	2	1	3	0	0	0	1	0	0	0	2	0	2	1	2	0	2	1	3
down	1	1	0	2	2	3	0	0	2	4	1	0	2	3	1	1	0	1	0	1
right	0	4	0	0	0	1	0	1	1	1	0	2	0	0	0	0	0	1	1	2
left	3	5	1	4	0	1	0	1	2	5	0	0	0	1	2	3	1	2	0	1
fist	0	2	2	4	0	0	1	0	0	3	0	1	1	1	0	2	1	1	1	3
spread	0	3	2	5	3	4	2	4	1	0	0	0	1	2	1	0	0	1	0	3
Sum	6	17	6	18	5	9	3	7	6	13	1	5	4	9	6	7	2	8	3	3
Error rate (%)	5	14	5	15	4.1	7.5	2.5	5.7	5	10.7	0.7	4.1	3.3	7.5	5	5.7	1.6	6.6	2.5	10.7

This experiment emphasizes that controlling the Lipschitz constant of a network improves its robustness not only against targeted adversarial attacks, as shown previously, but also in the case of black-box attacks, where no prior information about the model is used.

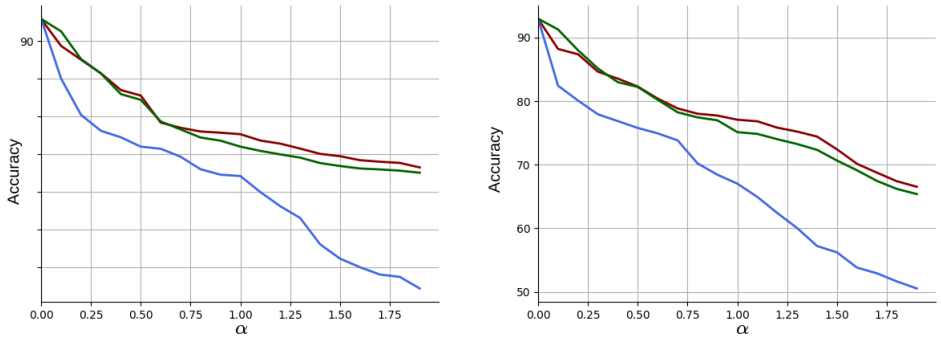
3.5.3 . Real-life scenario validation

To illustrate the practical applicability of our findings, we proceed to validate our model in a real-life context. For this purpose, we designed an experiment to compare a conventionally trained model with the constrained one. We integrated both models in a real-time application that controls a 3D hand on a screen, as well as a game that can be controlled by gestures, to give the user tangible feedback. We used the `Unity`[3] platform to design and control a 3D hand and then encapsulated our models in an application that performed real-time inference and the hand was moving on the screen in accordance with the predicted gesture. We asked 10 volunteers (males and females) to test both models by performing each gesture 20 times. We emphasize that the user had no prior knowledge about what model was implemented since it was randomly selected at the beginning of each new trial. Pictures of the experimental setup are provided in Figure 3.5. Table 3.7 details on a user level, how many (out of the 20) trials were erroneously classified. `U` and `C` denote the Unconstrained and the Constrained models, respectively. Note that, despite obtaining very good results on the test set, the unconstrained model loses a lot in terms of performance (up to 15%) when facing real-life data. We can observe that training a positive neural network subject to Lipschitz constraints improves the overall robustness of the classifier against adversarial perturbations, not only from a theoretical viewpoint but also practically by leading to more reliable systems with greater generalization power.

As for the other application, we asked the volunteers to play 2 rounds of a gesture-controlled game, one with each model. The game was inspired by the famous `Temple Run` [4], and consists of a moving cube which the user controls via gestures.

[3] https://unity.com/

[4] https://play.google.com/store/apps/details?id=com.imangi.templerun&



(a) $\sigma_0 = \alpha$; $\sigma_1 = 10\alpha$; $p = 0.15$

(b) $\sigma_0 = \alpha$; $\sigma_1 = 10\alpha$; $p = 0.3$

(c) $\sigma_0 = \alpha$; $\sigma_1 = 10\alpha$; $p = 0.45$

(d) $\sigma_0 = \alpha$; $\sigma_1 = 10\alpha$; $p = 0.15$

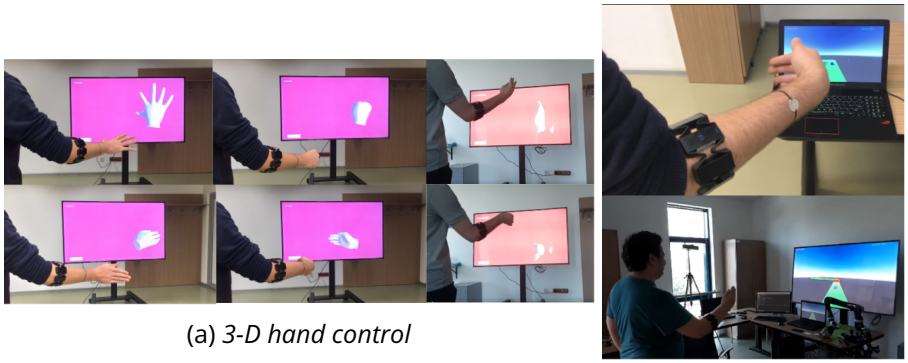
(e) $\sigma_0 = \alpha$; $\sigma_1 = 10\alpha$; $p = 0.3$

(f) $\sigma_0 = \alpha$; $\sigma_1 = 10\alpha$; $p = 0.45$

Figure 3.4: *Accuracy vs. α in the context of Noisy Inputs training. (a), (b), (c): 7-gesture dataset; (d), (e), (f): 13-gesture dataset. Red line: robust model; Blue line: baseline model; Green line: adversarial trained model*

The player can move his/her hand left or right to move the character to either side

`hl=en&gl=US&pli=1`



(a) *3-D hand control*

(b) *Gesture-control game*

Figure 3.5: *Real-life experimental setup*

Table 3.8: Training time for different constraints in the case of an $m = 6$-layer network. The training time is computed on a batch of 2048 examples.

Constraint	None	$\widetilde{\mathrm{P}}_{\widetilde{\mathcal{C}}_i \cap \mathcal{D}_i}$	$\mathrm{P}_{\widetilde{\mathcal{C}}_i \cap \mathcal{D}_i}$	$\widetilde{\mathrm{P}}_{\check{\mathcal{C}}_{i,n} \cap \mathcal{D}_i}$	$\mathrm{P}_{\check{\mathcal{C}}_i \cap \mathcal{D}_i}$	$\widetilde{\mathrm{P}}_{\mathcal{C}_{i,n} \cap \mathcal{D}_i}$	$\mathrm{P}_{\mathcal{C}_{i,n} \cap \mathcal{D}_i}$
time/step [ms]	7	9	11	9	11	18	28

of the screen to avoid obstacles. The player can also move the hand up to jump or spread its fingers to shoot and clear the obstacles ahead. The game is over when the player fails to take a turn or to jump/ clear an obstacle. We observed that 70% of the users were able to obtain higher scores when they used the constrained model, showing again that our solution is more stable when it comes to real-life applications.

3.5.4 . Limitations

Increased training time is one of the main limitations of our proposed approach. Indeed, to compute the true projection, the proposed method uses an iterative algorithm that performs singular value decomposition at each iteration, which is a resource-consuming operation, especially when performed on large matrices. We propose several lower complexity solutions, which have proved to offer a good trade-off between training time, robustness, and performance. Table 3.8 shows the training time for all the proposed constraint algorithms The time is measured per step, which consists of a batch of 2048 examples. Nevertheless, it is worth noting that the additional time overhead is applicable only during the training phase. The inference is the same for all the models, around 7 ms per step.

Another limitation is related to the fact that our method for controlling the Lipschitz constant of the system is currently applicable in the context of nonnegative weighted fully connected feed-forward neural networks. Although the performance remains good for the considered AGR systems, the nonnegativity constraint might

lead to a loss of expressivity of the neural networks in other inference tasks. In future work, we plan to extend our method towards more general neural network architectures, including convolutional layers, skip connections, etc.

3.6 . Conclusion

This chapter has shown the usefulness of designing robust feed-forward neural networks for automatic gesture recognition based on sEMG physiological signals. More precisely, we proposed to finely control the Lipschitz constant of these non-linear systems by considering positively weighted neural architectures. To offer robustness certificates, we also developed new optimization techniques for training classifiers subject to spectral norm constraints on the weights. We studied various constrained formulations and showed that robustness can be secured without sacrificing accuracy when using a combination of tight constraints and exact projections. We also provide several lower-complexity solutions, which reduce the training time significantly.

Experiments on four distinct datasets illustrated the good performance of our approach. We further demonstrated the effectiveness of our robust classifier, compared to classically trained ones, when facing white-box and black-box attacks. We also want to highlight that one of the key advantages of our research was the ability to conduct real-life experiments. This was made possible because we had access to a specialized acquisition module tailored for capturing gesture data. The availability of this acquisition module allowed us to gather real-world gesture data in a controlled setting, which closely mimics practical scenarios. By conducting experiments with real users and their gestures, we could thoroughly evaluate the performance and accuracy of our proposed methodology. This real-life experimentation not only provided us with invaluable insights into the effectiveness of our approach but also demonstrated its feasibility and potential for implementation in real-world applications. In future works, it would be interesting to apply such a robust training procedure to other applications in pattern recognition involving data acquired in real-time.

3.7 . Appendix – accelerated DFB algorithm

Let $n \in \mathbb{N}\backslash\{0\}$ and $i \in \{1, \dots, m\}$. Computing the projection of a matrix $\overline{W}_i \in \mathbb{R}^{N_i \times N_{i-1}}$ onto $\mathcal{D}_i \cap \mathcal{C}_{i,n}$ is equivalent to solve the following matrix optimization problem:

$$\underset{W_i \in \mathbb{R}^{N_i \times N_{i-1}}}{\text{minimize}} \quad \iota_{\mathcal{D}_i}(W_i) + \iota_{\mathcal{B}(0,\overline{\vartheta})}(A_{i,n} W_i B_{i,n}) + \frac{1}{2}\|W_i - \overline{W}_i\|_{\mathrm{F}}^2 \tag{3.29}$$

where $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm and $\iota_{\mathcal{S}}$ denotes the indicator of a set $\mathcal{S}$ (this function is equal to 0 on this set and $+\infty$ otherwise.) The dual optimization problem associated to this strongly convex minimization problem reads

$$\underset{Y \in \mathbb{R}^{N_m \times N_0}}{\text{minimize}} \quad f^*(-A_{i,n}^\top Y B_{i,n}^\top) + \iota^*_{\mathcal{B}(0,\overline{\vartheta})}(Y), \tag{3.30}$$

where, for a given function g, g^* denotes its Fenchel-Legendre conjugate. In our case $f = \iota_{\mathcal{D}_i} + \frac{1}{2}\|\cdot - \overline{W}_i\|_{\mathrm{F}}^2$. From standard conjugation rules [115], f^* is equal to

$$(\forall W_i \in \mathbb{R}^{N_i \times N_{i-1}}) \quad f^*(W_i) = \widetilde{\iota_{\mathcal{D}_i}}(W_i + \overline{W}_i), \tag{3.31}$$

where $\widetilde{\iota_{\mathcal{D}_i}}$ is the Moreau envelope of $\iota^*_{\mathcal{D}_i}$ given by

$$\widetilde{\iota_{\mathcal{D}_i}}(W_i) = \inf_{W_i' \in \mathbb{R}^{N_i \times N_{i-1}}} \iota^*_{\mathcal{D}_i}(W_i') + \frac{1}{2}\|W_i' - W_i\|_{\mathrm{F}}^2. \tag{3.32}$$

The Moreau envelope of a proper lower-semincontinuous convex function is differentiable. Thus f^* is differentiable, and its gradient is [125, Example 17.33]

$$\nabla f^*(W_i) = \mathsf{P}_{\mathcal{D}_i}(W_i + \overline{W}_i). \tag{3.33}$$

We deduce that the gradient of $Y \mapsto f^*(-A_{i,n}^\top Y B_{i,n}^\top)$ is

$$-A_{i,n}\mathsf{P}_{\mathcal{D}_i}(\overline{W}_i - A_{i,n}^\top Y B_{i,n}^\top)B_{i,n}.$$

Since $\mathsf{P}_{\mathcal{D}_i}$ is a nonexpansive operator, the latter function has a Lipschitz gradient with constant $\beta = \|A_{i,n}\|_{\mathrm{S}}^2 \|B_{i,n}\|_{\mathrm{S}}^2$. The dual problem (3.30) thus corresponds to the minimization of the sum of a smooth convex function and a proper lower-semicontinuous function. Consequently, it can be minimized by a proximal algorithm. Such a strategy will require calculating the proximity operator of $\gamma\iota^*_{\mathcal{B}(0,\overline{\vartheta})}$ for some scaling parameter $\gamma \in]0, +\infty[$. By using Moreau's formula [125], this proximity operator is expressed as

$$(\forall Y \in \mathbb{R}^{N_m \times N_0}) \ \ \mathrm{prox}_{\gamma\iota^*_{\mathcal{B}(0,\overline{\vartheta})}}(Y) = Y - \gamma \mathsf{P}_{\mathcal{B}(0,\overline{\vartheta})}(\gamma^{-1} Y). \tag{3.34}$$

A classical solution for solving the dual problem consists in using the standard forward-backward algorithm [126, 109]. This leads to Algorithm 3 [116]. Another solution consists in using the FISTA-like algorithm in [127], which leads to the

accelerated version in Algorithm 2. The sequences $(Y_\ell)_{\ell\in\mathbb{N}}$ generated by these two algorithms are guaranteed to converge to a solution $\widehat{Y}$ to the dual problem. In addition, from Kuhn-Tucker conditions, the solution to the primal problem $\widehat{W}_i = \mathsf{P}_{\mathcal{S}_{i,n}}(\overline{W}_i)$ is equal to $\nabla f^*(-A_{i,n}^\top \widehat{Y} B_{i,n}^\top)$. It follows from (3.33) and the continuity of $\mathsf{P}_{\mathcal{D}_i}$ that the sequence $(V_\ell)_{\ell\in\mathbb{N}}$ converges to $\widehat{W}_i$.

Algorithm 3: Dual Forward-Backward algorithm

Let $Y_0 \in \mathbb{R}^{N_m \times N_0}$
Set $\epsilon \in]0, 1/(\|A_{i,n}\|_{\mathrm{S}}\|B_{i,n}\|_{\mathrm{S}})^2[$
for $l = 0, 1, \ldots$ **do**
  Set $\gamma_\ell \in [\epsilon, 2/(\|A_{i,n}\|_{\mathrm{S}}\|B_{i,n}\|_{\mathrm{S}})^2 - \epsilon]$
  $V_\ell = \mathsf{P}_{\mathcal{D}_i}(\overline{W}_i - A_{i,n}^\top Y_\ell B_{i,n}^\top)$
  $\widetilde{Y}_\ell = Y_\ell + \gamma_\ell A_{i,n} V_\ell B_{i,n}$
  $Y_{\ell+1} = \widetilde{Y}_\ell - \gamma_\ell \mathsf{P}_{\mathcal{B}(0,\overline{\nu})}(\gamma_\ell^{-1}\widetilde{Y}_\ell)$
return V_l

Chapter 4 – Signal denoising using new classes of robust neural networks

In this chapter, we focus on robust solutions for a regression problem, namely audio signal denoising. We address the task at hand from two perspectives. First, we only concentrate on denoising the magnitude elements resulting from a Fourier analysis of the audio signal. To this end, we design a fully connected network, called Adaptive Convolutional Neural Network (ACNN), whose layers have a special structure that exhibits some similarity with a 1D convolutional one. In the second part of the chapter, we extend our approach to denoising the whole complex spectrum of the audio signals, using complex-valued neural networks (CVNN). For both solutions, we derive tight Lipschitz bounds and propose robust training mechanisms which are later validated on denoising piano music clips corrupted by various levels of additive white noise.

4.1 . Adaptive convolutional networks

Convolutive Neural Networks (CNNs) and their variants have been applied successfully to a variety of learning problems in various domains such as image classification [128, 129], face recognition [130], object tracking [131], natural language processing [5], speech enhancement [132, 133] etc. Despite their massive success with respect to fully connected neural networks, CNNs are difficult to analyze theoretically and thus raise some robustness issues. However, as Deep Neural Networks (DNNs) have highly complex and non-linear structures, an accurate and efficient estimation of the Lipschitz constant remains a challenge. In the recent literature [20, 58], different techniques have been proposed to derive Lipschitz bounds for deep feed-forward neural networks using non-expansive activation operators. As we will show in Chapter 5, these techniques may, however, be challenging to apply to CNNs.

This section introduces a new class of neural networks, called Adaptive Convolutional Neural Networks (ACNN), which can be seen as an intermediate between CNNs and Fully Connected Networks (FCNs). Learning capabilities of CNNs being well-investigated and proven, we take advantage of this potential by structuring the weights of our network in a similar manner. A significant difference is that the network makes use of filters that are no longer time/space invariant, similar to what is done in adaptive filtering.

Thus, such network architecture appears more flexible. In addition, we show that we can obtain tight Lipschitz certificates to ensure its robustness. More precisely, we control the Lipschitz constant of the network to reach a good performance-

robustness balance by training the system under suitable spectral norm constraints. Our approach focuses on neural networks with nonnegative weights for which we can efficiently compute an optimal Lipschitz constant, as shown in the previous chapter.

Many recent developments in the field of deep learning emphasize the importance of convolutions in neural network architectures [134] and the improvement in their performance [135]. To the best of our knowledge, there are few works that propose ways to improve FCN architectures. In [136], the performance increase is achieved by reducing the sparsity of the network and improving the gradient flow in it. In our work, we also aim at improving the performance of an FCN, but we follow a different approach. The linear layers of our network have a structure similar to convolutive ones, in the sense that each neuron input is a weighted sum of the outputs of a given limited number of neurons in the preceding layer and the processing is split into multiple channels operating in a parallel manner. In contrast to convolutive layers, ACNN uses multiple kernels at each channel, leading to more flexibility in the choice of weights. Not only do we reach a good performance, but we also keep it robust to input perturbations by providing Lipschitz certificates.

4.1.1 . Making the bridge between CNNs and FCNs

In this section we aim at filling the gap between FCNs and CNNs. In terms of signal processing concepts, a convolutive layer is a Multiple-Input Multiple-Output (MIMO) filter. For one-dimensional signals, each of these filters can be viewed as a Tœplitz matrix generated by the impulse response of the filter, which is applied to the vector of signal samples. If the filter length is short, large upper and lower triangular parts of this matrix are null. In our proposed approach, we will keep this band structure for the weight matrix, which is equivalent to performing local processing at each time within a sliding window. However, in order to add more flexibility to this architecture, we will allow all the nonzero coefficients of this matrix to be fully optimized.

The proposed architecture is depicted in Figure 4.1a. Figure 4.1.b is the graphical representation of the presented concept. As it can be observed, the weights are split to emulate a MIMO system. Each kernel is associated with a specific shape of the matrix, which is depicted in Figure 4.1.c. The lower and upper triangular null parts are displayed in dark gray, while the light gray central part contains overlap and may use different weight values.

4.1.2 . Learning algorithm

For training the proposed ACNN, we use a stochastic gradient-like optimization based on the popular ADAM method [137]. Consider the vector of parameters of the network, $\eta = (\eta_i)_{1 \leqslant i \leqslant m}$, such that, for each layer $i \in \{1, \ldots, m\}$, η_i represents a vector of dimension $N_i(N_{i-1}+1)$, composed of the elements of the weight matrix

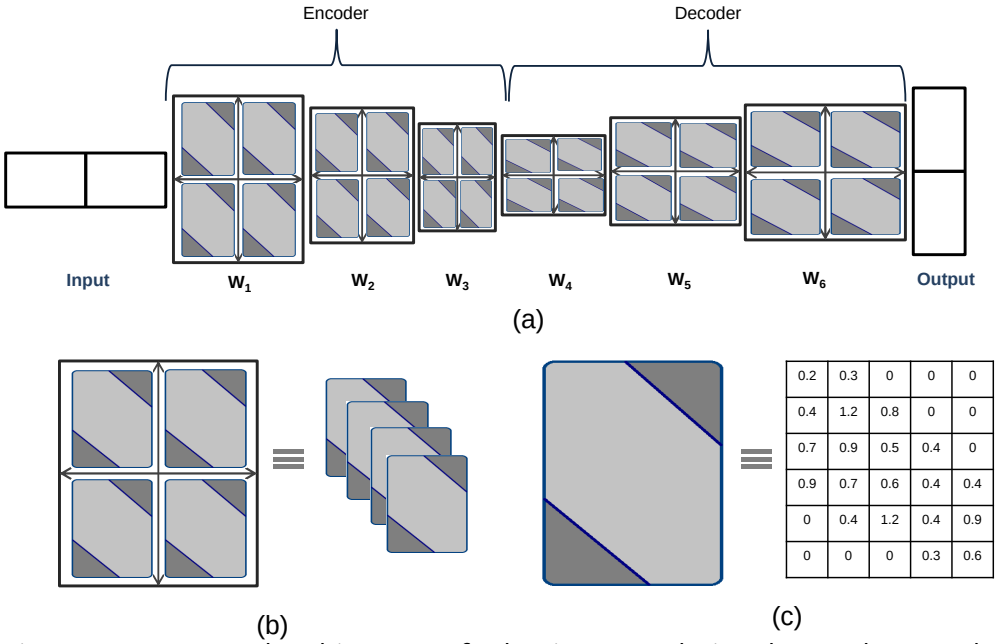


Figure 4.1: Proposed architecture of Adaptive Convolutional Neural Network (ACNN). a) An encoder-decoder architecture composed of a 6-layer FCN followed by ReLU activation function. b) Relation between proposed FCNs and CNNs; the weights are split into sub-matrices simulating convolutive filters in CNNs c) Each of the sub-matrices is constrained to have a band structure as shown in this example. The dark grey area marks the zero-entries, while the light-grey colour corresponds to the ones that are allowed to be non-zero.

W_i and the components of the bias vector b_i.

To secure the conditions of robustness while imposing the desired structure for our network, the parameter vector η is projected onto a closed set $\mathcal{S}$ that expresses all these constraints. The parameter update at epoch $n > 0$ is performed for mini-batches $(\mathbb{M}_{q,n})_{1\leqslant q\leqslant Q}$. If the training data are denoted by $(z_k)_{1\leqslant k\leqslant K}$, where z_k is the k-th pair of inputs and their associated outputs, the operations performed during the n-th epoch are summarized in Algorithm 4, where the square, the square root, and the division are performed component-wise, and

$$\mathcal{S}_{i,t} = \big\{\eta_i \mid [(\eta_{j,t+1}^\top)_{j<i} \;\; \eta_i^\top \;\; (\eta_{j,t}^\top)_{j>i}]^\top \in \mathcal{S}\big\}. \tag{4.1}$$

Here-above, ℓ denotes the loss function, ∇_i represents the gradients with respect to η_i; $\mu_{i,t}$ and $\nu_{i,t}$ represent the first and second momentum estimates at the iteration t, initialized with $\mu_{i,0} = \nu_{i,0} = 0$. $\mathrm{P}_{\mathcal{S}_{i,t}}$ designates the projection onto the constraint set $\mathcal{S}_{i,t}$. Although the set $\mathcal{S}$ is non-convex, the sets $\mathcal{S}_{i,t}$ will be defined as the intersection of three closed and convex constraint sets, as detailed next. The parameters used for learning are set to $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\gamma = 0.001$, and

Algorithm 4: Projected ADAM Algorithm

Partition $\{1,\dots,K\}$ *into mini-batches* $(\mathbb{M}_{q,n})_{1\leqslant q\leqslant Q}$
foreach $q \in \{1,\dots,Q\}$ **do**
  $t = (n-1)Q + q$
  foreach $i \in \{1,\dots,m\}$ **do**
    $g_{i,t} = \sum_{k\in\mathbb{M}_{q,n}} \nabla_i \ell\big(z_k, (\eta_{i,t})_{1\leqslant i\leqslant m}\big)$
    $\mu_{i,t} = \beta_1\mu_{i,t-1} + (1-\beta_1)g_{i,t}$
    $\nu_{i,t} = \beta_2\nu_{i,t-1} + (1-\beta_2)g_{i,t}^2$
    $\gamma_t = \gamma\sqrt{1-\beta_2^t}/(1-\beta_1^t)$
    $\eta_{i,t+1} = \mathsf{P}_{\mathcal{S}_{i,t}}\Big(\eta_{i,t} - \gamma_t\mu_{i,t}/(\sqrt{\nu_{i,t}}+\epsilon)\Big)$,

$\epsilon = 10^{-12}$.

To ensure the computation of a tight robustness bound, as explained in Section 3.2.2, we impose non-negative weights for every $i \in \{1,\dots,m\}$ by considering the constraint set:

$$\mathcal{D}_i = \{W_i \in \mathbb{R}^{N_i\times N_{i-1}} \mid W_i \geqslant 0\} \tag{4.2}$$

Let R_i (resp. Q_i) be the number of output (resp. input) channels used in layer $i \in \{1,\dots,m\}$. The proposed algebraic structure of each weight operator is guaranteed by splitting the corresponding matrix W_i into $R_i \times Q_i$ sub-matrices of dimension $N_i' \times M_i'$ (with $N_i' = N_i/R_i$ and $M_i' = N_{i-1}/Q_i$), denoted by $(W_i^{(r,q)})_{1\leqslant r\leqslant R_i, 1\leqslant q\leqslant Q_i}$. The desired band structure of the sub-matrix $W_i^{(r,q)}$ is ensured by imposing that it belongs to the following vector space:

$$\mathcal{E}_i = \{(V_{u,v})_{1\leqslant u\leqslant N_i', 1\leqslant v\leqslant M_i'} \in \mathbb{R}^{N_i'\times M_i'} \mid \forall (u,v) \text{ s.t. } |(M_i'-1)(u-1) - (N_i'-1)(v-1)| \geqslant d_i, V_{u,v} = 0\}.$$

Herein, d_i is an integer and, if $N_i' = M_i' > 1$, $2\lfloor d_i/(N_i'-1)\rfloor - 1$ plays a role similar to a kernel length in standard CNNs.

Finally, to control the robustness, we need to limit the Lipschitz constant of the network to a given value $\overline{\vartheta} > 0$. The related constraint can be expressed as:

$$\mathcal{C}_{i,t} = \{W_i \in \mathbb{R}^{N_i\times N_{i-1}} \mid \|A_{i,t}W_iB_{i,t}\|_{\mathrm{S}} \leqslant \overline{\vartheta}\} \tag{4.3}$$

$$A_{i,t} = W_{m,t}\cdots W_{i+1,t}, \tag{4.4}$$

$$B_{i,t} = W_{i-1,t+1}\cdots W_{1,t+1} \tag{4.5}$$

with the convention that for the first layer ($i = 1$), $B_{i,t}$ is the $N_i \times N_i$ identity matrix Id_{N_i} and for the last layer ($i = m$), $A_{i,m} = \mathrm{Id}_{N_i}$. Hereinabove, $(W_{j,t})_{1\leqslant j\leqslant m}$

				PSNR	MSE	CC
Noisy Signal				18.25	1.18×10^{-2}	0.76
Denoised Signal	Baseline - Wavelet-based denoiser			20.66	$1.00 \times 10{-}3$	0.80
	ACNN denoiser	Scenario (i)	$\overline{\vartheta} = 1$	24.27	3.73×10^{-3}	0.96
			$\overline{\vartheta} = 5$	29.03	1.25×10^{-3}	0.97
			$\overline{\vartheta} = 10$	33.76	6.53×10^{-4}	0.98
		Scenario (ii)	$\overline{\vartheta} = 1$	25.87	3.12×10^{-3}	0.96
			$\overline{\vartheta} = 5$	30.63	8.63×10^{-4}	0.98
			$\overline{\vartheta} = 10$	36.02	2.23×10^{-4}	0.99
	Standard FCN denoiser		$\overline{\vartheta} = 1$	23.38	4.59×10^{-3}	0.90

Table 4.1: Comparison of different variants of the proposed method with baselines.

designates the estimates of the weight matrices at iteration t of the projected ADAM optimizer.

The projection onto the intersection of the above three closed convex sets has no closed-form expression. The intersection $\mathcal{D}_i \cap \mathcal{E}_i^{R_i \times Q_i}$ is however quite simple to handle since the projection onto this set reduces to $\mathrm{P}_{\mathcal{D}_i} \circ \mathrm{P}_{\mathcal{E}_i^{R_i \times Q_i}}$. To compute the final projection onto $\mathcal{C}_{i,t} \cap (\mathcal{D}_i \cap \mathcal{E}_i^{R_i \times Q_i})$ of a weight matrix W_i, we use the dual forward-backward algorithm, as presented in the previous chapter.

4.1.3 . Experimental Evaluation

The proposed network has been evaluated for denoising music signals.

4.1.3.1 . Dataset Description

We train our proposed ACNN on a dataset consisting of musical exercises and songs performed on a Ronald organ. The organ covers 5 octaves (range `C2-C7`), each octave having 12 semitones, generating a total of 61 different possible notes. For the recordings, the whole range of notes is used. The songs are recorded in MIDI format using MidiEditor, in the following manner: the recording mode from MidiEditor is activated before each song is played and is stopped after the song is finished (so there is silence at the beginning and end of each recording). In total, the dataset contains 100 MIDI recordings, with a sampling frequency $F_s = 44100$ Hz, constituting 1 h and 17 min of audio. The data set is available online[1].

The dataset is divided into training, validation, and test sets. The training set contains 90 clips with variable lengths, ranging from 6 s up to 150 s, having in total over an hour (67 min) of audio recordings. Some songs are repeated on different octaves to obtain a minimum number of occurrences for all notes. The validation

[1] https://speed.pub.ro/downloads/

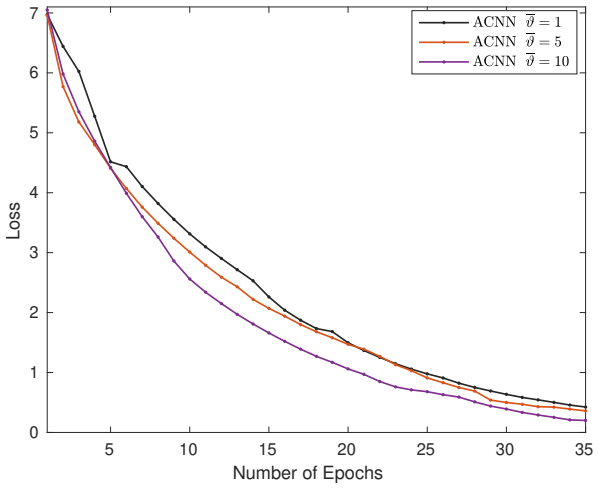


Figure 4.2: Convergence profile of the proposed method.

dataset contains 9 songs with lengths between 12 and 120 s, forming around 10 minutes of audio signals. The test set has one 2-minute-long clip.

4.1.3.2 . Experimental setup

The noisy data for training, validating, and testing is generated by adding white Gaussian noise to the original samples. The noise has zero mean and its standard deviation is randomly chosen so that the resulting signal-to-noise ratio (SNR) varies between 5 and 30 dB. The dataset samples are normalized between 0 and 1. We extract the frequency features from the audio signal using a *Short-Time Fourier Transform* (STFT). The network estimates the STFT coefficients of the samples, and an *Inverse Short-Time Fourier Transform (ISTFT)* is performed as the post-processing step. We consider a `Hanning` sliding analysis window of length $T = 23$ ms, with an overlap between two consecutive windows of 50%. The STFT is performed on 1024 points. In total, from each audio segment, a vector of length $L = 513$ frequency coefficients is obtained, constituting the input of our ACNN.

The denoising is performed using a 6-layer ACNN architecture, as presented in Figure 4.1. The network has an encoder-decoder structure. Each layer employs *ReLU* as activation function followed by a *batch normalization* step that acts as a regularizer and prevents the model from overfitting.

4.1.3.3 . Simulations and results

In order to measure the performance of our proposed ACNN architecture, we perform two sets of experiments. In the first set, we control the Lipschitz constant

of the architecture for three values $\overline{\vartheta}$ equal to 1, 5, and 10. In the second experiment, we test our architecture by varying the number of channels, i.e. the way we split each weight matrix. Note that for both scenarios we consider a network with $m = 6$ layers and $\forall i \in \{1, \ldots, 6\}$, $d_i = d'_i(\max\{N'_i, M'_i\} - 1)$, having the following characteristics:

(i) $R_1 = 2 = Q_6$, $Q_1 = 1 = R_6$, $\forall i \in \{2, \ldots, 5\}$ $R_i = Q_i = 2$, $(N_i)_{1 \leqslant i \leqslant 6} = (400, 200, 100, 200, 400, 513)$, $(d'_i)_{1 \leqslant i \leqslant 6} = (237, 80, 30, 30, 80, 237)$;

(ii) $R_1 = 3 = Q_6$, $Q_1 = 1 = R_6$, $\forall i \in \{2, \ldots, 5\}$ $R_i = 3$, $Q_i = 3$, $(N_i)_{1 \leqslant i \leqslant 6} = (630, 510, 420, 510, 630, 513)$, $(d'_i)_{1 \leqslant i \leqslant 6} = (237, 85, 65, 65, 85, 237)$.

We evaluate the performance on 3 standard metrics: *Peak to Signal Noise Ratio (PSNR)*, *Mean squared error (MSE)*, and *Cross-correlation (CC)*, as shown in the Table 4.1. We compare our method with a standard denoising technique, based on a *Wavelet* decomposition. We employ a 5-level decomposition using $Symlet8$ filters combined with *SureShrink* thresholding. We also compare ACNN with an FCN implementation, for which we ensured the Lipschitz bound $\overline{\vartheta} = 1$. Here, the FCN has no structural constraints but, in addition to the imposed Lipschitz property, it has positive weights.

Table 4.1 reports the quantitative results obtained by all three approaches, evaluated over the test set. Our method outperforms the baseline on all measures by a significant margin. ACNN also outperforms classical FCN, inferring that the structure imposed on the weight matrix for ACNN leads to a model with better generalization power, whereas FCN implementation may be prone to overfitting. We observed that, without any Lipschitz constraint, FCNs tend to have a very high Lipschitz constant of the order $10^5 - 10^6$, which emphasizes the importance of controlling the Lipschitz behaviour of the system. The Lipschitz constant of the network is also closely related to the expressiveness of the trained model. Architectures with tight robustness constraints have fewer degrees of freedom and are thus expected to be less accurate and lead to a slower convergence. The convergence profile of our proposed method is shown in Figure 4.2.

This section is a step towards bridging the gap between FCNs and CNNs. We split the weight matrix at each layer of an FCN in such a way that it acts similarly to multichannel convolutive filters in a CNN. We verified the effectiveness of the proposed architecture by showing its denoising capabilities on music clips. It is worth noting that the method is not only limited to such a regression problem. In the next section, the idea of controlling the Lipschitz constant for complex-valued architectures will be tackled.

4.2 . Design of robust complex-valued feed-forward neural networks

Complex valued neural networks (CVNNs) [138, 139, 140] have been a topic of ongoing interest in the data science community. Indeed, being able to deal with complex-valued data is of high interest in the signal processing domain. Even when analyzing real-valued signals (e.g., audio or images), one of the most commonly used approaches is based on frequency analysis, which immediately leads us to the complex plane. Magnitude and phase carry different information types, which offer insight into the information content of the signals [141]. Handling complex-valued data may be difficult since it requires more computational power, but it has also been shown that CVNNs may be more expressive than classical neural networks [142]. For example, [143] shows that an end-to-end complex-valued neural network outperforms a 2-channel real-valued network in the context of accelerated MRI reconstruction. In [144], CVNNs are employed for non-stationary RADAR data analysis. Also, it was shown that smaller CVNNs outperform larger real networks when dealing with physical data [145].

In this section, we introduce a new class of neural networks operating in the complex domain, called *Robust Complex Feed-Forward Network (RCFF-Net)*. The structure of the network is inspired by CapsNets [146, 147]. The weight matrices process the real and imaginary parts distinctly. The activation functions handle the correlation between the real and imaginary parts of complex pairs. We demonstrate that, for the proposed structure, the Lipschitz constant can be efficiently computed. In addition, we develop a learning strategy to control this constant and ensure the robustness of the network against adversarial perturbations. We show that this approach offers better performance than real-valued neural networks in the context of audio denoising.

4.2.1 . Theoretical background

A complex-valued feedforward neural network is defined as follows.

Model 4.2.1 Let $m \in \mathbb{N} \setminus \{0\}$. T is an m-layer complex-valued feedforward neural network if there exists $(N_i)_{0 \leqslant i \leqslant m} \in (\mathbb{N} \setminus \{0\})^{m+1}$ such that

$$T = T_m \circ \cdots \circ T_1 \tag{4.6}$$

where, for every $i \in \{1, \ldots, m\}$, $T_i = R_i(W_i \cdot + b_i)$, $W_i \in \mathbb{C}^{N_i \times N_{i-1}}$, $b_i \in \mathbb{C}^{N_i}$, and $R_i \colon \mathbb{C}^{N_i} \to \mathbb{C}^{N_i}$.

In the following, we will make the assumption that the activation operators $(R_i)_{1 \leqslant i \leqslant m}$ satisfy some nonexpansiveness properties and that all of them, except possibly for the last layer, are separable.

Assumption 4.2.2

(i) R_m is a nonexpansive (i.e., 1-Lipschitz) operator and, for every $i \in \{1, \dots, m-1\}$, R_i is separable, that is

$$\big(\forall z = (\zeta_k)_{1 \leqslant k \leqslant N_i} \in \mathbb{C}^{N_i}\big) \quad R_i z = \big(\varrho_{i,k}(\zeta_k)\big)_{1 \leqslant k \leqslant N_i}, \tag{4.7}$$

where, for every $k \in \{1, \dots, N_i\}$, $\varrho_{i,k} \colon \mathbb{C} \to \mathbb{C}$.

(ii) For every $i \in \{1, \dots, m-1\}$ and $k \in \{1, \dots, N_i\}$, $\varrho_{i,k}$ is α_i-averaged where $\alpha_i \in]0, 1]$, i.e.

$$(\forall (\zeta, \zeta') \in \mathbb{C}^2) \quad |\varrho_{i,k}(\zeta) - \varrho_{i,k}(\zeta')|^2 + \frac{1-\alpha_i}{\alpha_i} |\zeta - \varrho_{i,k}(\zeta) - \zeta' + \varrho_{i,k}(\zeta')|^2 \leqslant |\zeta - \zeta'|^2. \tag{4.8}$$

Note that, when $\alpha_i = 1$ in (4.8), we recover the definition of a nonexpansive function. More generally, nonexpansive functions form a superset of α_i-averaged functions. It is also worth to note that (4.8) is equivalent to the fact that the function defined as

$$(\forall \zeta \in \mathbb{C}) \quad \vartheta_{i,k}(\zeta) = \frac{\rho_{i,k}(\zeta) - (1-\alpha_i)\zeta}{\alpha_i} \tag{4.9}$$

is nonexpansive. When $\alpha_i = 1/2$, $\rho_{i,k}$ is a firmly nonexpansive function. It can be shown that if $\alpha_i > 1/2$, then $\rho_{i,k}$ is α_i averaged if and only if it is an overrelaxation of a firmly nonexpansive function [148], i.e. there exists $\lambda_{i,k} \in]1, 2]$ and a firmly nonexpansive function $\sigma_{i,k} \colon \mathbb{C} \to \mathbb{C}$ such that

$$(\forall \zeta \in \mathbb{C}) \quad \rho_{i,k}(\zeta) = (1 - \lambda_{i,k})\zeta + \lambda_{i,k}\sigma_{i,k}(\zeta). \tag{4.10}$$

4.2.2 . Nonexpansive complex-valued activation functions

There exist two main recipes for building activation functions, satisfying (4.8). The first one is to use split-complex activation functions of the form

$$(\forall z \in \mathbb{C}) \quad \varrho_{i,k}(\zeta) = \varrho_{i,k}^{\mathrm{R}}(\mathrm{Re}\zeta) + \imath \varrho_{i,k}^{\mathrm{I}}(\mathrm{Im}\zeta) \tag{4.11}$$

where $\varrho_{i,k}^{\mathrm{R}} \colon \mathbb{R} \to \mathbb{R}$ and $\varrho_{i,k}^{\mathrm{I}} \colon \mathbb{R} \to \mathbb{R}$ are α_i-averaged activation functions. It is shown in [20] that most existing real-valued activation functions are averaged. The majority of them are proximity operators of some proper lower-semicontinuous functions and are thus firmly nonexpansive. Examples of functions within this class are:

- the split-complex ReLU function

$$(\forall \zeta \in \mathbb{C}) \quad \varrho_{i,k}(\zeta) = \mathbb{C}\mathrm{ReLU}(\zeta) = \mathrm{ReLU}(\mathrm{Re}\zeta) + \imath\, \mathrm{ReLU}(\mathrm{Im}\zeta) \tag{4.12}$$

where

$$(\forall \xi \in \mathbb{R}) \quad \mathrm{ReLU}(\xi) = \begin{cases} \xi & \text{if } \xi \geqslant 0 \\ 0 & \text{otherwise.} \end{cases} \tag{4.13}$$

- the split-complex hyperbolic tangent function

$$(\forall \zeta \in \mathbb{C}) \quad \varrho_{i,k}(\zeta) = \mathbb{C}\tanh(\zeta) = \tanh(\mathrm{Re}\zeta) + \imath \tanh(\mathrm{Im}\zeta). \tag{4.14}$$

The second recipe we propose is based on the following property.

Proposition 4.2.3 *Let $\omega\colon [0,+\infty[\to \mathbb{R}$ be α-averaged with $\alpha \in]0,1]$ and such that $\omega(0) = 0$. Let ρ be defined as*

$$(\forall \zeta \in \mathbb{C}) \quad \rho(\zeta) = \begin{cases} \dfrac{\omega(|\zeta|)}{|\zeta|}\zeta & \text{if } \zeta \neq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{4.15}$$

Then the following hold

(i) *ρ is α-averaged.*

(ii) *if ω is the restriction to $[0,+\infty[$ of the proximity operator of a convex function $\varphi\colon \mathbb{R} \to \mathbb{R}$ which is even and differentiable on $\mathbb{R} \setminus \{0\}$, then ρ is the proximity operator of*

$$\begin{aligned} \Phi\colon \mathbb{C} &\to \mathbb{R}\colon \\ \zeta &\mapsto \varphi(|\zeta|). \end{aligned} \tag{4.16}$$

Proof.

(i) Let us first show that the property is satisfied when $\alpha = 1$. Then, for every $(\zeta, \zeta') \in (\mathbb{C} \setminus \{0\})^2$,

$$\begin{aligned} &|\rho(\zeta) - \rho(\zeta')|^2 \\ =&|\rho(\zeta)|^2 + |\rho(\zeta')|^2 - 2\mathrm{Re}\{\rho(\zeta)\rho(\zeta')^*\} \\ =&\omega(|\zeta|)^2 + \omega(|\zeta'|)^2 - 2\frac{\omega(|\zeta|)\omega(|\zeta'|)}{|\zeta||\zeta'|}\mathrm{Re}\{\zeta\zeta'^*\}. \end{aligned} \tag{4.17}$$

Since ω is nonexpansive and vanishes in 0,

$$|\omega(|\zeta|)| = |\omega(|\zeta|) - \omega(0)| \leqslant ||\zeta| - 0| = |\zeta|, \tag{4.18}$$

which implies that

$$\frac{\omega(|\zeta|)\omega(|\zeta'|)}{|\zeta||\zeta'|} \leqslant 1. \tag{4.19}$$

From the latter inequality, we deduce that

$$\begin{aligned} &|\zeta - \zeta'|^2 - |\rho(\zeta) - \rho(\zeta')|^2 \\ =&|\zeta|^2 - \omega(|\zeta|)^2 + |\zeta'|^2 - \omega(|\zeta'|)^2 - 2\Big(1 - \frac{\omega(|\zeta|)\omega(|\zeta'|)}{|\zeta||\zeta'|}\Big)\mathrm{Re}\{\zeta\zeta'^*\} \\ \geqslant&|\zeta|^2 - \omega(|\zeta|)^2 + |\zeta'|^2 - \omega(|\zeta'|)^2 - 2\Big(1 - \frac{\omega(|\zeta|)\omega(|\zeta'|)}{|\zeta||\zeta'|}\Big)|\zeta||\zeta'| \\ \geqslant&(|\zeta| - |\zeta'|)^2 - \big(|\omega(|\zeta|) - \omega(|\zeta'|)\big)^2 \\ \geqslant&0, \end{aligned} \tag{4.20}$$

where the last inequality follows again from the nonexpansiveness of ω. In summary, we have proved that, for every $(\zeta, \zeta') \in (\mathbb{C} \setminus \{0\})^2$

$$|\rho(\zeta) - \rho(\zeta')| \leqslant |\zeta - \zeta'|. \tag{4.21}$$

On the other hand if $\zeta \in \mathbb{C} \setminus \{0\}$ and $\zeta' = 0$, the inequality is also satisfied since, according to (4.18),

$$|\rho(\zeta) - \rho(0)| = |\rho(\zeta)| = \frac{|\omega(|\zeta|)}{|\zeta|}|\zeta| \leqslant |\zeta - 0|. \tag{4.22}$$

(4.21) also trivially holds if $\zeta = \zeta' = 0$. This shows that the property holds when $\alpha = 1$.

Let us now look at the case when $\alpha \neq 1$. Since ω is assumed to be α-averaged, there exists a nonexpansive operator ϖ such

$$(\forall \xi \in [0, +\infty[) \quad \omega(\xi) = (1 - \alpha)\xi + \alpha\varpi(\xi). \tag{4.23}$$

In addition, $\varpi(0) = 0$. Let

$$(\forall \zeta \in \mathbb{C}) \quad \vartheta(\zeta) = \begin{cases} \frac{\varpi(|\zeta|)}{|\zeta|}\zeta & \text{if } \zeta \neq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{4.24}$$

It follows from the first part of the proof that ϑ is nonexpansive. We have then, for every $\zeta \in \mathbb{C} \setminus \{0\}$,

$$\begin{aligned} \rho(\zeta) =& \frac{(1 - \alpha)|\zeta| + \alpha\varpi(|\zeta|)}{|\zeta|}\zeta \\ =& (1 - \alpha)\zeta + \alpha\vartheta(\zeta). \end{aligned} \tag{4.25}$$

This relation remains valid if $\zeta = 0$, which shows that ρ is α-averaged.

(ii) This is a direct consequence of [149, Proposition 24.27].

□

Proposition 4.2.3 allows us to show that many complex-valued activation functions $\rho_{i,k}$ operating jointly on the real and imaginary parts of their input satisfy (4.8). Let us give a few examples.

(i) The Hirose function [139] is defined as

$$(\forall \zeta \in \mathbb{C}) \quad \rho_{i,k}(\zeta) = \begin{cases} \dfrac{\tanh(|\zeta|)}{|\zeta|}\zeta & \text{if } \zeta \neq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{4.26}$$

Based on [60, Example 2.12], it is the proximity operator of function

$$\begin{aligned}&\Phi\colon \mathbb{C} \to \left]-\infty, +\infty\right]\\ &\qquad \zeta \mapsto \begin{cases} \dfrac{(1+|\zeta|)\ln(1+|\zeta|) + (1-|\zeta|)\ln(1-|\zeta|) - |\zeta|^2}{2} & \text{if } |\zeta| < 1;\\ \ln(2) - 1/2 & \text{if } |\zeta| = 1;\\ +\infty, & \text{if } |\zeta| > 1. \end{cases}\end{aligned} \tag{4.27}$$

(ii) The Georgiou-Katsageras function [150] defined as

$$(\forall \zeta \in \mathbb{C}) \quad \rho_{i,k}(\zeta) = \mathrm{GK}(\zeta) = \frac{\zeta}{1+|\zeta|} \tag{4.28}$$

is the proximity operator of

$$\begin{aligned}&\Phi\colon \mathbb{C} \to \left]-\infty, +\infty\right]\\ &\qquad \zeta \mapsto \begin{cases} -|\zeta| - \ln(1-|\zeta|) - \dfrac{|\zeta|^2}{2}, & \text{if } |\zeta| < 1;\\ +\infty, & \text{if } |\zeta| \geqslant 1 \end{cases}\end{aligned} \tag{4.29}$$

(see [60, Example 2.15]).

(iii) The squashing function used in CapsNets [147] defined as

$$(\forall \zeta \in \mathbb{C}) \quad \rho_{i,k}(\zeta) = \frac{8}{3\sqrt{3}} \frac{\zeta}{1+|\zeta|^2}. \tag{4.30}$$

is the proximity operator of a convex function (see [20, Example 3.2(ii)]).

(iv) The modulus-based inverse square root unit activation function

$$(\forall \zeta \in \mathbb{C}) \quad \rho_{i,k}(\zeta) = \frac{\zeta}{\sqrt{1+|\zeta|^2}} \tag{4.31}$$

is the proximity operator of

$$\Phi\colon \mathbb{R} \to \left]-\infty, +\infty\right] : \zeta \mapsto \begin{cases} -|\zeta|^2/2 - \sqrt{1-|\zeta|^2}, & \text{if } |\zeta| \leqslant 1;\\ +\infty, & \text{if } |\zeta| > 1 \end{cases} \tag{4.32}$$

(see [60, Example 2.9]).

(v) The modulus-based swish activation defined as

$$(\forall \zeta \in \mathbb{C}) \quad \rho_{i,k}(\zeta) = \frac{10\zeta}{11(1+\exp(-|\zeta|))}, \tag{4.33}$$

is α-averaged with $\alpha \approx 0.546$ (see [20, Example 3.2(iv)]).

Note that, unlike split-complex activation functions, any function ρ defined by (4.15) is phase-preserving, in the sense that, for every $\zeta \in \mathbb{C}$, $\arg \rho(\zeta) = \arg \zeta$. Another example of an activation function which is not phase-preserving is the group sort one, expressed as

$$(\forall \zeta \in \mathbb{C}) \quad \rho_{i,k}(\zeta) = \zeta^{\uparrow} = \min\{\mathrm{Re}\zeta, \mathrm{Im}\zeta\} + \iota \max\{\mathrm{Re}\zeta, \mathrm{Im}\zeta\}. \tag{4.34}$$

This activation function is nonexpansive [20, Example 3.5]. We can also perform convex combinations of nonexpansive activation functions to create new ones. For example, let $(\mu_1, \mu_2, \mu_3) \in [0,1]^3$ be such that $\mu_1 + \mu_2 + \mu_3 = 1$ and define

$$(\forall \zeta \in \mathbb{C}) \quad \rho_{i,k}(\zeta) = \mu_1 \mathbb{C}\mathrm{ReLU}(\zeta) + \mu_2 \,\mathrm{GK}(\zeta) + \mu_3 \zeta^{\uparrow}. \tag{4.35}$$

The above function is not-phase preserving (if $\mu_1 + \mu_3 \neq 0$) and, as consequence of [149, Proposition 4.42], it is α-averaged with

$$\alpha = \frac{1}{2}(\mu_1 + \mu_2) + \mu_3 = \frac{1 + \mu_3}{2}. \tag{4.36}$$

4.2.3 . Robustness results

In the following, for every $M \in \mathbb{N} \setminus \{0\}$ and $\alpha \in]0,1]$, we define the following set

$$\mathscr{B}_{\alpha}^{M} = \left\{ \mathrm{Diag}(\lambda_1, \ldots, \lambda_M) \;\middle|\; (\forall i \in \{1, \ldots, M\}) \;\; \lambda_i \in \mathbb{C} \;\text{ and }\; |\lambda_i - 1 + \alpha| = \alpha \right\}. \tag{4.37}$$

The convex envelope of this set is

$$\mathscr{D}_{\alpha}^{M} = \left\{ \mathrm{Diag}(\lambda_1, \ldots, \lambda_M) \;\middle|\; (\forall i \in \{1, \ldots, M\}) \;\; \lambda_i \in \mathbb{C} \;\text{ and }\; |\lambda_i - 1 + \alpha| \leqslant \alpha \right\}. \tag{4.38}$$

In the complex plane, each of the diagonal elements of a matrix in $\mathscr{B}_{\alpha}^{M}$ (resp. $\mathscr{D}_{\alpha}^{M}$) lies on a circle (resp. in a closed disk) with center $1 - \alpha$ and radius α.
In addition, for every complex-valued matrix (or vector) A, $|A|$ denotes the matrix (or vector) formed with the moduli of the elements of A.

Proposition 4.2.4 *Consider Model 4.2.1 where Assumption 4.2.2 holds. Define*

$$\theta_m = \sup_{\substack{\Lambda_1 \in \mathscr{B}_{\alpha_1}^{N_1} \\ \vdots \\ \Lambda_{m-1} \in \mathscr{B}_{\alpha_{m-1}}^{N_{m-1}}}} \|W_m \Lambda_{m-1} \cdots \Lambda_1 W_1\|. \tag{4.39}$$

Then θ_m is a Lipschitz constant of T. In addition,

$$\|W_m \cdots W_1\| \leqslant \theta_m \leqslant \||W_m| \cdots |W_1|\|. \tag{4.40}$$

Proof. For every $i \in \{1, \dots, m\}$, let $P_i = R_i(\cdot + b_i)$. Let $(z, z') \in (\mathbb{C}^{N_0})^2$. For every $i \in \{1, \dots, m-1\}$, let

$$y_i = (y_{i,k})_{1 \leqslant k \leqslant N_i} = P_i \circ W_i \circ \cdots \circ P_1 \ \circ W_1 z - P_i \circ W_i \circ \cdots \circ P_1 \ \circ W_1 z'. \quad (4.41)$$

Since R_m (hence, P_m) is assumed to be nonexpansive,

$$\|Tz - Tz'\| \leqslant \|W_m y_{m-1}\|. \quad (4.42)$$

For every $k \in \{1, \dots, N_{m-1}\}$,

$$\begin{aligned} & y_{m-1,k} \\ & = \rho_{m-1,k}([W_{m-1} \circ \cdots \circ P_1 \ \circ W_1 z]_k) - \rho_{m-1,k}([W_{m-1} \circ \cdots \circ P_1 \ \circ W_1 z']_k) \\ & = (1 - \alpha_k)([W_{m-1} \circ \cdots \circ P_1 \ \circ W_1 z]_k - [W_{m-1} \circ \cdots \circ P_1 \ \circ W_1 z']_k) \\ & \quad + \alpha_k \big(\vartheta_{m-1,k}([W_{m-1} \circ \cdots \circ P_1 \ \circ W_1 z]_k) - \vartheta_{m-1,k}([W_{m-1} \circ \cdots \circ P_1 \ \circ W_1 z']_k)\big), \end{aligned} \quad (4.43)$$

where $[\cdot]_k$ designate the k-th component of the vector in argument and $\vartheta_{m-1,k}$ is a nonexpansive operator. Because of the nonexpansiveness property, there exists $\omega_{m-1,k} \in \mathbb{C}$ (generally function of (z, z')) such that $|\omega_{m-1,k}| \leqslant 1$ and

$$\begin{aligned} & \vartheta_{m-1,k}([W_{m-1} \circ \cdots \circ P_1 \ \circ W_1 z]_k) - \vartheta_{m-1,k}([W_{m-1} \circ \cdots \circ P_1 \ \circ W_1 z']_k) \\ & \quad = \omega_{m-1,k}([W_{m-1} \circ \cdots \circ P_1 \ \circ W_1 z]_k - [W_{m-1} \circ \cdots \circ P_1 \ \circ W_1 z']_k). \end{aligned} \quad (4.44)$$

Thus, by setting $\lambda_{m-1,k} = 1 - \alpha_k + \alpha_k \omega_{m-1,k}$, we deduce that

$$y_{m-1,k} = \lambda_{m-1,k}([W_{m-1} \circ \cdots \circ P_1 \ \circ W_1 z]_k - [W_{m-1} \circ \cdots \circ P_1 \ \circ W_1 z']_k) \quad (4.45)$$

and $|\lambda_{m-1,k} - 1 + \alpha_k| \leqslant \alpha_k$. In summary, we have shown that there exists $\Lambda_{m-1} \in \mathscr{D}_{\alpha_{m-1}}^{N_{m-1}}$ (depending on (z, z')) such that

$$y_{m-1} = \Lambda_{m-1} W_{m-1} y_{m-2}. \quad (4.46)$$

By applying the same procedure iteratively, there exists $\Lambda_{m-2} \in \mathscr{D}_{\alpha_{m-2}}^{N_{m-2}}, \dots,$ $\Lambda_1 \in \mathscr{D}_{\alpha_1}^{N_1}$ such that

$$y_{m-1} = \Lambda_{m-1} W_{m-1} \Lambda_{m-2} W_{m-2} \cdots \Lambda_1 W_1 (z - z'). \quad (4.47)$$

It follows from (4.42) that

$$\|Tz - Tz'\| \leqslant \|W_m \Lambda_{m-1} \cdots \Lambda_1 W_1\| \|z - z'\|, \quad (4.48)$$

which implies that

$$\|Tz - Tz'\| \leqslant \theta_m \|z - z'\|, \quad (4.49)$$

where

$$\theta_m = \sup_{\substack{\Lambda_1 \in \mathscr{D}_{\alpha_1}^{N_1} \\ \vdots \\ \Lambda_{m-1} \in \mathscr{D}_{\alpha_{m-1}}^{N_{m-1}}}} \|W_m \Lambda_{m-1} \cdots \Lambda_1 W_1\|. \tag{4.50}$$

For every $i \in \{1, \dots, m-1\}$, let $(\lambda_{i,k})_{1 \leqslant k \leqslant N_i}$ be the diagonal elements of Λ_i. The function

$$((\mathrm{Re}\{\lambda_{i,k}\}, \mathrm{Im}\{\lambda_{i,k}\})_{1 \leqslant k \leqslant N_i})_{1 \leqslant i \leqslant m-1} \mapsto \|W_m \Lambda_{m-1} \cdots \Lambda_1 W_1\| \tag{4.51}$$

is a multi-convex function (i.e., convex with respect to each of its arguments). It follows from [20, Lemma A.2] that the supremum in (4.50) can be restricted to diagonal elements $(\lambda_{i,k})_{1 \leqslant i \leqslant m-1, 1 \leqslant k \leqslant N_i}$ of matrices $(\Lambda_i)_{1 \leqslant i \leqslant m-1}$ reaching the bounds of the interval imposed on $(|\lambda_{i,k} - 1 + \alpha_i|)_{1 \leqslant i \leqslant m-1, 1 \leqslant k \leqslant N_i}$. This means that the expression of θ_m reduces to (4.39).

The lower bound in (4.40) is simply obtained by noticing that, for every $i \in \{1, \dots, m-1\}$, the identity matrix of dimension N_i belongs to $\mathscr{B}_{\alpha_i}^{N_i}$.

For every $i \in \{1, \dots, m-1\}$, let $\Lambda_i = \mathrm{Diag}\left((\lambda_{i,k})_{1 \leqslant k \leqslant N_i}\right) \in \mathscr{B}_{\alpha_i}^{N_i}$. Note that, for every $k \in \{1, \dots, N_i\}$,

$$|\lambda_{i,k}| \leqslant |\lambda_{i,k} - 1 + \alpha_i| + 1 - \alpha_i = 1. \tag{4.52}$$

Let $y_0 = (y_{0,k_0})_{1 \leqslant k_0 \leqslant N_0} \in \mathbb{C}^{N_0}$ and let

$$y_m = (y_{m,k})_{1 \leqslant k \leqslant N_m} = W_m \Lambda_{m-1} \cdots \Lambda_1 W_1 y_0. \tag{4.53}$$

Thus, we deduce from (4.47) that, for every $k_m \in \{1, \dots, N_m\}$,

$$y_{m,k_m} = \sum_{k_0=1}^{N_0} \cdots \sum_{k_{m-1}=1}^{N_{m-1}} \lambda_{m-1,k_{m-1}} \cdots \lambda_{1,k_1} [W_m]_{k_m,k_{m-1}} \dots [W_1]_{k_1,k_0} \, y_{0,k_0}. \tag{4.54}$$

Because of (4.52),

$$|y_{m,k_m}| \leqslant \sum_{k_0=1}^{N_0} \cdots \sum_{k_{m-1}=1}^{N_{m-1}} |[W_m]_{k_m,k_{m-1}}| \dots |[W_1]_{k_1,k_0}| |y_{0,k_0}|. \tag{4.55}$$

Thus

$$\|y_m\| \leqslant \||W_m| \cdots |W_1| |y_0|\| \leqslant \||W_m| \cdots |W_1|\| \, \||y_0|\| = \||W_m| \cdots |W_1|\| \|y_0\|. \tag{4.56}$$

This yields the upper bound in (4.40). □

Remark 4.2.5

(i) In (4.52), Matrices $\Lambda_i = \mathrm{Diag}\left((\lambda_{i,k})_{1\leqslant k\leqslant N_i}\right) \in \mathscr{B}_{\alpha_i}^{N_i}$. with $i \in \{1,\dots,m-1\}$ are such that $\|\Lambda_i\| \leqslant 1$ (as a consequence of (4.52)). We deduce that

$$\theta_m \leqslant \|W_m\| \cdots \|W_1\|. \tag{4.57}$$

The upper bound corresponds to a classical, less accurate estimate of the Lipschitz constant of the network [15].

(ii) Although these results extend those given in [20, Theorem 5.2], there exists a fundamental difference between the complex-valued case and the real one. For the Lipschitz bound established for the real case in [20, Theorem 5.2], the supremum is calculated on a finite set of values. In contrast, the Lipschitz constant (4.39) requires computing it on an infinite set of parameters, since the sets $(\mathscr{B}_{\alpha_i}^{N_i})_{1\leqslant i\leqslant m-1}$ are not countable. However, we will show next that, for a specific choice of the weight operators, it is easy to compute a Lipschitz constant of the network.

Proposition 4.2.6 *Consider Model 4.2.1. For every $i \in \{1,\dots,m\}$, let $W_i^+ \in [0,+\infty[^{N_i\times N_{i-1}}$. Let $(\beta_{1,k})_{1\leqslant k\leqslant N_0} \in [0,2\pi[^{N_0}$, let $(\beta_{m,k})_{1\leqslant k\leqslant N_m} \in [0,2\pi[^{N_1}$, and for every $i \in \{2,\dots,m-1\}$, let $\beta_i \in [0,2\pi[$. Suppose that Assumption 4.2.2 holds and that the weight operators of the network are such that*

$$\begin{aligned} &W_1 = W_1^+ \,\mathrm{Diag}\left(e^{\imath\beta_{1,1}},\dots,e^{\imath\beta_{1,N_0}}\right)\\ &(\forall i \in \{2,\dots,m-1\}) \quad W_i = e^{\imath\beta_i} W_i^+\\ &W_m = \mathrm{Diag}\left(e^{\imath\beta_{m,1}},\dots,e^{\imath\beta_{m,N_m}}\right) W_m^+. \end{aligned} \tag{4.58}$$

Then

$$\theta_m = \|W_m^+ \cdots W_1^+\|. \tag{4.59}$$

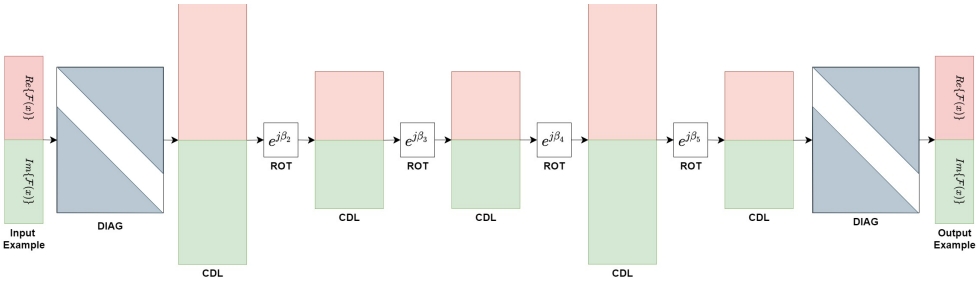
Proof. This result is a direct consequence of Proposition 4.2.4. Indeed, we have

$$\begin{aligned} \|W_m\cdots W_1\| &= \left\| \exp\left(\imath \sum_{i=2}^{m-1} \beta_i\right) W_m W_{m-1}^+ \cdots W_2^+ W_1 \right\| \\ &= \left\| \mathrm{Diag}\left(e^{\imath\beta_{m,1}},\dots,e^{\imath\beta_{m,N_m}}\right) W_m^+ \cdots W_1^+ \,\mathrm{Diag}\left(e^{\imath\beta_{1,1}},\dots,e^{\imath\beta_{1,N_0}}\right)\right\| \\ &= \|W_m^+ \cdots W_1^+\|. \end{aligned} \tag{4.60}$$

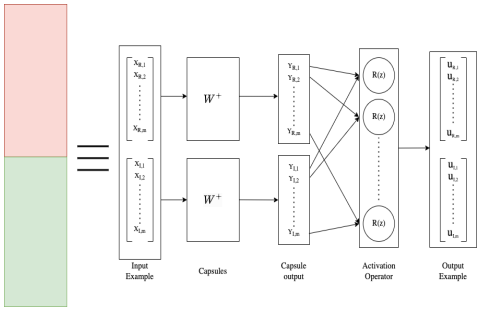
On the other hand, for every $i \in \{1,\dots,m\}$, $|W_i| = W_i^+$, and

$$\||W_m|\dots|W_1|\| = \|W_m^+ \cdots W_1^+\|. \tag{4.61}$$

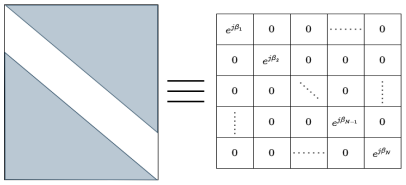
Then Expression (4.59) follows from the sandwich inequality (4.40). □



(a) The proposed architecture: 5 CDLs (1024, 512, 512, 1024, and 513 neurons, respectively) followed by a Rotation layer (ROT) or a Diagonal layer (DIAG).



(b) The structure of the dense complex layer: each group of neurons (capsule) will process jointly the real part and the imaginary part of the coefficients.



(c) The structure of a diagonal layer: the white band corresponds to the main diagonal which features non-zero coefficients.

Figure 4.3: Overview of the RCFF-Network. The red part denotes the real part, while the green accounts for the imaginary part.

Remark 4.2.7 If the activation operators used at layers $i \in \{2, \dots, m-1\}$ are phase-preserving, they commute with the multiplication by factors $e^{\imath\beta_i}$. This means that the factor $\exp\left(\imath \sum_{i=2}^{m-1} \beta_i\right)$ can be factorized and applied at the output of the neural network. This phase shift factor then becomes redundant with matrix $\mathrm{Diag}\left(e^{\imath\beta_{m,1}}, \dots, e^{\imath\beta_{m,N_m}}\right)$. In other words, when phase-preserving activation functions are used, without loss of generality, the angles $(\beta_i)_{2 \leqslant i \leqslant m}$ can be set to zero.

4.2.4 . Proposed approach

We implement our architecture to meet the requirements of Proposition 4.2.6 and design a Robust Complex Feed-Forward Neural Network (RCFF-Net). The architecture is illustrated in Figure 4.3. The network processes complex-valued data by stacking their real and imaginary parts. The weights associated with the first and last layers, are *Diagonal layers* (DIAG), as they perform phase shifts in the complex plane, which will be optimized during the training phase. At the core of the architecture stands the *Complex Dense Layer* (CDL), detailed in Figure 4.3b. This layer encapsulates two equal nonnegative linear transforms that process distinctly the real and imaginary parts. In contrast, the activation functions operate

Table 4.2: Experimental results for audio denoising

				MSE	PSNR [db]	CC
Noisy signal				7.21×10^{-3}	21.02	0.83
Baseline – Wiener Filter				3.45×10^{-3}	24.24	0.94
Baseline – NLMS Adaptive Filter				2.52×10^{-3}	25.61	0.95
Baseline – Standard FCN				2.78×10^{-3}	26.05	0.95
RCFF	$\rho(\zeta) = \mathbb{C}\mathrm{ReLU}(\zeta)$	U	$\theta_{\mathrm{upp}} = 335$	0.96×10^{-3}	30.00	0.99
		C	$\theta_m = 0.99$	2.02×10^{-3}	27.64	0.96
	$\rho(\zeta) = \mathrm{GK}(\zeta)$	U	$\theta_{\mathrm{upp}} = 73.25$	1.04×10^{-3}	29.45	0.97
		C	$\theta_m = 0.99$	2.11×10^{-3}	27.14	0.96
	$\rho(\zeta) = \frac{8}{3\sqrt{3}} \frac{\lvert\zeta\rvert}{1+\lvert\zeta\rvert^2}\zeta$	U	$\theta_{\mathrm{upp}} = 120$	0.96×10^{-3}	30.19	0.98
		C	$\theta_m = 0.93$	1.22×10^{-3}	29.02	0.97
	$\rho(\zeta) = \mathbb{C}\tanh(\zeta)$	U	$\theta_{\mathrm{upp}} = 421$	1.28×10^{-3}	28.98	0.97
		C	$\theta_m = 0.99$	2.09×10^{-3}	27.41	0.96
	$\rho(\zeta) = \frac{\zeta}{\sqrt{1+\lvert\zeta\rvert^2}}$	U	$\theta_{\mathrm{upp}} = 143$	1.90×10^{-3}	27.80	0.96
		C	$\theta_m = 0.97$	2.12×10^{-3}	26.98	0.96
	$\rho(\zeta) = \frac{\tanh(\lvert\zeta\rvert)}{\lvert\zeta\rvert}$	U	$\theta_{\mathrm{upp}} = 98$	1.43×10^{-3}	28.60	0.97
		C	$\theta_m = 0.98$	1.93×10^{-3}	27.63	0.97
	$\rho(\zeta) = \zeta^{\uparrow}$	U	$\theta_{\mathrm{upp}} = 187$	1.09×10^{-3}	30.21	0.98
		C	$\theta_m = 0.99$	1.32×10^{-3}	29.13	0.97
ACNN		C	$\theta_m = 1.00$	1.98×10^{-3}	26.24	0.96

on each pair of real-imaginary coefficients, and they can be chosen as explained in Section 4.2.1. The output is then obtained by concatenating the real and imaginary parts. Between CDL layers, we apply a rotation operation (ROT), which induces a global phase shift of the outputs of each layer.

4.2.4.1 . Training strategy

Concerning the training strategy, we propose to use a similar approach to the case of ACNNs. We employ a projected version of the AdaMax optimizer [137]. As before, we define for each layer $i \in \{1, \dots, m\}$ the associated vector of parameters η_i which gathers the weight elements W_i^+, the angles, and the bias vectors b_i. For every $i \in \{2, \dots, m-1\}$, the size of this vector is $N_i(N_{i-1}+1)+1$, whereas for the first (resp. last) layer it is equal to $N_1(N_0+1)+N_0$ (resp. $N_m(N_{m-1}+2)$). During the learning phase, the parameters $(\eta_i)_{1 \leqslant i \leqslant m}$ are constrained to belong to a closed set $\mathcal{S}$ which expresses our robustness certificate. More precisely, we adopt a block coordinate approach where at each iteration t the vector η_i is projected onto a closed and convex set

$$\mathcal{S}_{i,t} = \big\{\eta_i \mid [(\eta_{j,t+1}^\top)_{j<i} \;\; \eta_i^\top \;\; (\eta_{j,t}^\top)_{j>i}]^\top \in \mathcal{S}\big\}. \tag{4.62}$$

Table 4.3: Experimental results for audio denoising with attacked inputs

				MSE	PSNR [db]	CC	Deg.[%]
Noisy signal				7.30×10^{-3}	21.00	0.83	0.09
Baseline – Standard FCN				5.46×10^{-3}	22.87	0.90	12.24
RCFF	$\rho(\zeta) = \mathbb{C}\mathrm{ReLU}(\zeta)$	U	$\theta_{\mathrm{upp}} = 335$	4.84×10^{-3}	23.62	0.91	21.26
		C	$\theta_m = 0.99$	1.96×10^{-3}	25.43	0.95	7.99
	$\rho(\zeta) = \mathrm{GK}(\zeta)$	U	$\theta_{\mathrm{upp}} = 73.25$	5.42×10^{-3}	23.31	0.90	20.84
		C	$\theta_m = 0.99$	1.84×10^{-3}	25.72	0.95	5.23
	$\rho(\zeta) = \frac{8}{3\sqrt{3}} \frac{\lvert\zeta\rvert}{1+\lvert\zeta\rvert^2}\zeta$	U	$\theta_{\mathrm{upp}} = 120$	5.26×10^{-3}	22.05	0.90	26.96
		C	$\theta_m = 0.93$	1.34×10^{-3}	28.68	0.97	1.17
	$\rho(\zeta) = \mathbb{C}\tanh(\zeta)$	U	$\theta_{\mathrm{upp}} = 421$	5.15×10^{-3}	23.14	0.90	22.14
		C	$\theta_m = 0.99$	2.82×10^{-3}	25.41	0.95	6.20
	$\rho(\zeta) = \frac{\zeta}{\sqrt{1+\lvert\zeta\rvert^2}}$	U	$\theta_{\mathrm{upp}} = 143$	6.02×10^{-3}	22.24	0.89	26.45
		C	$\theta_m = 0.97$	2.98×10^{-3}	25.12	0.94	8.14
	$\rho(\zeta) = \frac{\tanh(\lvert\zeta\rvert)}{\lvert\zeta\rvert}$	U	$\theta_{\mathrm{upp}} = 98$	5.78×10^{-3}	21.36	0.89	23.32
		C	$\theta_m = 0.98$	5.46×10^{-3}	25.56	0.95	5.61
	$\rho(\zeta) = \zeta^{\uparrow}$	U	$\theta_{\mathrm{upp}} = 187$	4.67×10^{-3}	23.09	0.90	22.34
		C	$\theta_m = 0.99$	1.45×10^{-3}	28.20	0.95	2.60
ACNN		C	$\theta_m = 1.00$	2.46×10^{-3}	25.43	0.95	3.08

The approach is implemented with a decomposition of the training set in mini-batches $(\mathbb{M}_{q,n})_{1 \leqslant q \leqslant Q}$ at epoch $n > 0$. The k-th sample of the dataset z_k corresponds to an input-output pair. K denotes the number of such samples in the training set. The optimization process for the n-th epoch is given by Algorithm 5.

Algorithm 5: Projected AdaMax Algorithm

Partition $\{1, \ldots, K\}$ *into mini-batches* $(\mathbb{M}_{q,n})_{1 \leqslant q \leqslant Q}$
foreach $q \in \{1, \ldots, Q\}$ **do**
  $t = (n-1)Q + q$
  foreach $i \in \{1, \ldots, m\}$ **do**
    $g_{i,t} = \sum_{k \in \mathbb{M}_{q,n}} \nabla_i \ell\big(z_k, (\eta_{i,t})_{1 \leqslant i \leqslant m}\big)$
    $\mu_{i,t} = \chi_1 \mu_{i,t-1} + (1 - \chi_1) g_{i,t}$
    $\nu_{i,t} = \max(\chi_2 \nu_{i,t-1}, |g_{i,t}|)$
    $\gamma_{i,t} = \gamma \mu_{i,t} / (1 - \chi_1^t)$
    $\eta_{i,t+1} = \mathsf{P}_{\mathcal{S}_{i,t}}\Big(\eta_{i,t} - \gamma_t \mu_{i,t} / (\sqrt{\nu_{i,t}} + \epsilon)\Big)$,
    $\eta_{i,t+1} = \mathsf{P}_{\mathcal{S}_{i,t}}(\eta_{i,t} - \gamma_{i,t} / \nu_{i,t})$

In this algorithm, the modulus and the division are performed component-wise. Hereabove, ℓ denotes the loss function, ∇_i represents the gradients with respect to η_i. The vectors $\mu_{i,t}$ and $\nu_{i,t}$ represent the first and second momentum estimates at iteration t, using parameters $\chi_1 = 0.9$ and $\chi_2 = 0.999$. These variables are initialized with $\mu_{i,0} = \nu_{i,0} = 0$. Each gradient step is followed by a projection $\mathsf{P}_{\mathcal{S}_{i,t}}$ onto the constraint set $\mathcal{S}_{i,t}$. This set expresses the two constraints on which our approach is grounded. First of all, since our assumptions are valid under a

nonnegativity condition for the weights, we need to ensure that $(\forall i \in \{1, \dots, m\}$, $W_i^+ \in [0, +\infty[^{N_i \times N_{i-1}}$. Additionally, based on Proposition 4.2.6, to control the robustness we impose that $\|W_m^+ \dots W_1^+\|_{\mathrm{S}} \leqslant \overline{\vartheta}$, where $\overline{\vartheta}$ is the target Lipschitz constant of the network.

4.2.5 . Experimental results

The proposed methodology is applied to the same problem as in the previous section. We use a 5-layer RFCC-Net ($m = 5$), with diverse activation functions, and use the same pre-processing pipeline as in Section 4.1.3.2. The main difference is that the network now estimates the complex STFT coefficients and, in the post-processing phase, an inverse operation (ISTFT) is performed for signal reconstruction.

We evaluate the performance of our RCFF-Net on the same 3 standard metrics: *Peak Signal-to-Noise Ratio (PSNR)*, *Cross-correlation (CC)*, and *Mean Squared Error (MSE)*, which was also employed as the training loss. The results on the test set are summarized in Table 4.2. We compare our solution with other standard denoising techniques, namely optimal Wiener filter and adaptive filter based on Normalised Least Mean Squares (NLMS) algorithm. As another baseline, we also trained a classical $m = 5$ layers Fully Connected Network (FCN) with ReLU activation. Furthermore, we trained RCFF-Net both using constrained and unconstrained weights, referred to in Table 4.2 as `C` and `U`, respectively. For the unconstrained model, since the weights may have arbitrary signs, we only compute the upper bound θ_{upp} of the Lipschitz constant, given by the right-hand side of (4.57).

In the constrained case, we were able to train nonexpansive models ($\theta_m \leqslant 1$). In accordance with the "no free lunch" theorem [123], this improved stability comes at the expense of a loss of performance that remains acceptable with our proposed method. We also compared our results with ACNN model, which only estimates the magnitude of the STFT coefficients. Note that RCFF-Net outperforms ACNN by a significant margin, showing that taking into account the whole spectrum information, *i.e.* both magnitude and phase, has a clear impact on the performance of the model.

Moreover, to show that our solution is indeed robust against adversarial perturbations, we have tested the performance of our models when facing adversarial inputs, in the cases when it was possible and relevant. To the best of our knowledge, there are very few white-box attackers suited for regression problems, and none are currently available for complex-valued NNs. So, to create a worst-case input perturbation, we extended the gradient-based attacker proposed in [28] to operate in the complex domain. The attacked input was created by adding the aforementioned perturbation over the clean audio sample. The results in the case of perturbed inputs are presented in Table 4.3. To emphasize the effectiveness of our solution, the last column from Table 4.3 shows the degradation level (in terms of percentage of SNR) when the model faces adversarial inputs.

Regarding the influence of the activation operator, it can be observed that, from the split-complex activations, $\mathbb{C}$ReLU achieves the best results, showing that ReLU remains a powerful operator even in the complex domain. From the activation operators defined by (4.15), the squashing function distinguishes as the best phase-preserving activation. We remark that the top results are obtained when we employ a non-phase preserving activation function, namely group sort, which shows that phase plays an important role when solving our task.

It can be observed that the performance of the models trained without stability guarantees is greatly affected, with a significant increase in the degradation level. On the other hand, when the models (both RCFF and ACNN) were trained under robustness constraints, the impact of the attack was limited, bounded by the imposed Lipschitz constant, proving the validity of our robust training mechanism.

4.3 . Conclusion

This chapter proposes two new classes of neural networks. The first one, ACNN, establishes a novel link between fully connected layers and convolutional structures, whereas the second one RCFF-Net operates in the complex space. By judiciously structuring the weight matrices, we derived a tight Lipschitz bound for both proposed architectures. In the complex case, our analysis led to new theoretical results concerning nonexpansive activation functions. We also extended an existing tight Lipschitz bound for feedforward neural networks to the complex domain. Computing this bound is no longer a combinatorial problem for complex-valued neural networks, which emphasizes the challenges raised with respect to the real case. We also showed how to control Lipschitz bounds numerically in the training process. We proved the effectiveness of our method in the context of audio denoising, but our method could be applied to other tasks such as source separation.

Chapter 5 – ABBA neural networks: coping with positivity, expressivity, and robustness

In this chapter, we introduce ABBA networks, a novel class of (almost) non-negative neural networks, which are shown to possess a series of appealing properties. In particular, we demonstrate that these networks are universal approximators while enjoying the advantages of non-negative weighted networks. We derive tight Lipschitz bounds both in the fully connected and convolutional cases. We propose a strategy for designing ABBA nets that are robust against adversarial attacks, by finely controlling the Lipschitz constant of the network during the training phase. We show that our method outperforms other state-of-the-art defenses against adversarial white-box attackers. Experiments are performed on image classification tasks on four benchmark datasets.

5.1 . Introduction

It is widely accepted that humans possess the innate ability to decompose complex interactions into discrete, intuitive hierarchical categories before analyzing them [151]. Conceptually, this evolution towards part-based representation in human cognition can be linked to non-negativity restrictions on the network weights [152]. This idea, along with other factors, has sparked interest in neural networks with non-negative weights. These networks have drawn attention for several reasons. Firstly, they align with human understandability, making them more interpretable. Secondly, the non-negativity constraint can act as beneficial regularization, effectively reducing overfitting issues. Moreover, recent studies have demonstrated that it is possible to derive a tight Lipschitz bound for such networks. This Lipschitz constant serves as a valuable metric for quantifying the robustness of the network, enabling us to design networks with enhanced resilience to adversarial perturbations during the training process. Despite their advantages, one significant drawback of networks with non-negative weights is that they might be less expressive than networks with arbitrary signed weights. Another major disadvantage of standard non-negative networks is that they are not universal approximators [153], a limitation that our work overcomes.

Approach. We are interested in neural networks having non-negative weights, except for the first and last linear layers. This class of networks obviously constitutes an extension of those having all their linear layers non-negative-valued. We focus on a particular subclass of these networks for which the weight matrices have a

structure of the form

$$\begin{bmatrix} A & B \\ B & A \end{bmatrix},$$

thus enjoying a number of algebraic properties. The corresponding networks are subsequently called ABBA networks. Note that weight matrices A and B are duplicated in ABBA networks, thus allowing us to limit the number of parameters.

5.2 . Related work

Non-negative neural networks. Inspired by non-negative matrix factorization (NMF) techniques, the work of [152] introduces non-negative restrictions on the weights to create neural networks in which the hidden units correspond to identifiable concepts. [154] showed that Autoencoders (AE) trained under non-negativity constraints are able to derive meaningful representations that unearth the hidden structure of high-dimensional data. Their method showed promising results from both performance and feature interpretation viewpoints on four different classification tasks. [155] presented the first polynomial-time algorithm for Probably Approximately Correct (PAC) learning 1-layer neural networks with positive coefficients. Moreover, ensuring non-negativity has been shown to have a regularization effect, reducing feature overfitting, which is a very common problem, especially for tasks where the available training data is scarce [156]. Neural networks defining convex functions of their inputs [157] also constitute a subclass of networks with non-negative weights.

Link with other networks. From another perspective, the idea of using redundant weights is reminiscent of siamese networks [158]. These architectures are successfully used to handle similarity learning tasks, such as face verification [159], character recognition [160], and object tracking [161]. Siamese networks compute a similarity metric on the representations of the inputs, after applying the same transformation to each one. Apart from the proven efficiency on solving computer vision tasks, they have lately been employed in NLP problems, e.g., computational argumentation. In [162], it is shown that siamese architectures outperform other baselines trained on convincingness datasets.

Robustness. The robustness of neural networks against possible adversarial attacks is a topic that has received increasing attention since nowadays AI-based solutions are ubiquitous [163, 164]. A sizable body of literature on adversarial attacks and different defense strategies has emerged in recent years as a result of the work in [9]), which revealed the alluring susceptibility of neural networks to adversarial perturbations and proposed a box-constrained L-BFGS algorithm for finding adversarial examples. [14] introduced the FGSM attack as a one-step modification of the input image, following the direction of loss maximization, while

[10] incorporated this step into an iterative method known as PGD, seen as an improvement over basic FGSM. DeepFool [38] iteratively searches for the closest adversarial point that directs the optimization towards crossing the decision boundary. DDN [42] and FMN [43] attacks fall into the category of projected-gradient methods, using iterative updates of the perturbation vector towards the minimization of its magnitude. Defensive strategies have been developed to alleviate this robustness issue. [51] divides adversarial defense methods into three categories: adversarial detection, gradient masking, and robust optimization. Adversarial Training was first introduced by [14] and later improved by [66]. Recent works on AT [165, 166] have successfully analyzed and refined training techniques, however, no theoretical certificates regarding their behavior in the presence of different adversaries have been established yet. Regularization-based methods, such as [167, 168, 169], include additional terms in their objective, steering the learning process in a direction that leads to better generalization. [170] provides robustness certificates for neural networks with one hidden layer, yielding an upper bound of the error in the presence of any adversary (see [171, 172, 77] for more advanced methods. Randomized smoothing [173, 174, 175, 176, 177] certifies the robustness of a classifier around an input point by measuring the most-likely prediction over Gaussian-corrupted versions of the point.

5.3 . ABBA neural networks

5.3.1 . Problem formulation

In the remainder of this paper, $\|\cdot\|$ will denote the ℓ_2-norm when dealing with a vector, and the spectral norm when dealing with a matrix.
An m-layer feedforward neural network can be described by the following model.

Model 5.3.1 T is a feedforward neural network if there exists $(N_i)_{1\leqslant i\leqslant m} \in (\mathbb{N}\setminus\{0\})^m$ such that

$$T = T_m \circ \cdots \circ T_1 \tag{5.1}$$

where, for every layer index $i \in \{1,\dots,m\}$, $T_i = R_i(W_i\cdot + b_i)$, $W_i \in \mathbb{R}^{N_i\times N_{i-1}}$ is the weight matrix, $b_i \in \mathbb{R}^{N_i}$ the bias vector, and $R_i\colon \mathbb{R}^{N_i} \to \mathbb{R}^{N_i}$ the activation operator. N_i corresponds to the number of inputs at the i-th layer. Such a layer is convolutive if it corresponds to a weight matrix W_i having some Taoeplitz (or block Toeplitz) structure.
We will say that the activation operator R_i is symmetric, if there exists $(c_i, d_i) \in (\mathbb{R}^{N_i})^2$ such that

$$(\forall x \in \mathbb{R}^{N_i}) \quad R_i(x) - d_i = -R_i(-x + c_i). \tag{5.2}$$

In other words, $(c_i, d_i)/2$ is a symmetry center of the graph of R_i.

For example, if R_i is squashing function used in CapsNets [147], it is such that

$$(\forall x \in \mathbb{R}^{N_i}) \quad R_i(x) = \frac{\mu\|x\|}{1+\|x\|^2}x. \tag{5.3}$$

with $\mu = 8/(3\sqrt{3})$. It thus satisfies the symmetry property (5.2) with $c_i = d_i = 0$. In addition, R_i is nonexpansive, i.e. it has a Lipschitz constant equal to 1 [20] Other examples of symmetric and nonexpansive activation operators are presented in Appendix 5.8.1 [1] .

5.3.2 . ABBA Matrices

We first define ABBA matrices, which will be the main algebraic tool throughout this chapter.

Definition 5.3.2 Let $(N_1, N_2) \in (\mathbb{N}\setminus\{0\})^2$. $\mathcal{A}_{N_1,N_2}$ is the set of ABBA matrices of size $(2N_2)\times(2N_1)$, that is $M \in \mathcal{A}_{N_1,N_2}$ if there exist matrices $A \in \mathbb{R}^{N_2\times N_1}$ and $B \in \mathbb{R}^{N_2\times N_1}$ such that

$$M = \begin{bmatrix} A & B \\ B & A \end{bmatrix}. \tag{5.4}$$

The sum matrix associated with M is then defined as $\mathfrak{S}(M) = A + B$.

We give some of the most relevant properties of these matrices. In particular, we will see that the ABBA structure is stable under standard matrix operations.

Proposition 5.3.3 *Let $(N_1, N_2, N_3) \in (\mathbb{N}\setminus\{0\})^3$.*

(i) *If $M \in \mathcal{A}_{N_2,N_1}$, then its transpose $M^\top \in \mathcal{A}_{N_1,N_2}$ and $\mathfrak{S}(M^\top) = \mathfrak{S}(M)^\top$.*

(ii) *If $(M_1, M_2) \in (\mathcal{A}_{N_2,N_1})^2$, then $M_1 + M_2 \in \mathcal{A}_{N_2,N_1}$ and $\mathfrak{S}(M_1 + M_2) = \mathfrak{S}(M_1) + \mathfrak{S}(M_2)$.*

(iii) *If $M_1 \in \mathcal{A}_{N_2,N_1}$ and $M_2 \in \mathcal{A}_{N_3,N_2}$, then $M_2M_1 \in \mathcal{A}_{N_3,N_1}$ and $\mathfrak{S}(M_2M_1) = \mathfrak{S}(M_2)\mathfrak{S}(M_1)$.*

(iv) *$\mathcal{A}_{N_1,N_1}$ is a ring when equipped with the standard matrix addition and product.*

(v) *If A and B are two square matrices of the same size, the eigenvalues of $\begin{bmatrix} A & B \\ B & A \end{bmatrix}$ are those of $A + B$ and $A - B$.*

(vi) *If A and B are two matrices having the same dimensions, the spectral norm of $\begin{bmatrix} A & B \\ B & A \end{bmatrix}$ is equal to $\max\{\|A + B\|, \|A - B\|\}$.*

[1] Appendices with number of the form SMx can be found in the supplementary materials.

(vii) *If $M \in \mathcal{A}_{N_2,N_1}$ has non-negative elements, the spectral norm of M is $\|\mathfrak{S}(M)\|$.*

(viii) *Let $A \in \mathbb{R}^{N_2 \times N_1}$ and $B \in \mathbb{R}^{N_2 \times N_1}$, and let $K = \min\{N_1, N_2\}$. Let $(\lambda_k)_{1 \leqslant k \leqslant K}$ (resp. $(\mu_k)_{1 \leqslant k \leqslant K}$) be the singular values of $A + B$ (resp. $A - B$) and let $\{u_k\}_{1 \leqslant k \leqslant K}$ / $\{v_k\}_{1 \leqslant k \leqslant K}$ (resp $\{t_k\}_{1 \leqslant k \leqslant K}$ / $\{w_k\}_{1 \leqslant k \leqslant K}$) be associated orthonormal families of left/right singular vectors in $\mathbb{R}^{N_2}$ / $\mathbb{R}^{N_1}$.*[2] *Then, the singular values of $\begin{bmatrix} A & B \\ B & A \end{bmatrix}$ are $(\lambda_k, \mu_k)_{1 \leqslant k \leqslant K}$ and associated orthonormal families of left/right singular vectors are*

$$\left\{ \frac{1}{\sqrt{2}} \begin{bmatrix} u_k \\ u_k \end{bmatrix}, \frac{1}{\sqrt{2}} \begin{bmatrix} t_k \\ -t_k \end{bmatrix} \right\}_{1 \leqslant k \leqslant K} \quad / \quad \left\{ \frac{1}{\sqrt{2}} \begin{bmatrix} v_k \\ v_k \end{bmatrix}, \frac{1}{\sqrt{2}} \begin{bmatrix} w_k \\ -w_k \end{bmatrix} \right\}_{1 \leqslant k \leqslant K}.$$

(ix) *If A and B are two matrices having the same dimensions,*

$$\operatorname{rank}\left(\begin{bmatrix} A & B \\ B & A \end{bmatrix} \right) = \operatorname{rank}(A + B) + \operatorname{rank}(A - B). \tag{5.5}$$

(x) *Let f be a function from $\mathbb{R}^{(2N_2)\times(2N_1)}$ to $\mathbb{R}^{(2N_2)\times(2N_1)}$. Assume that either f operates elementwise or it is a spectral function in the sense that there exists a function $\varphi\colon \mathbb{R}_+ \to \mathbb{R}_+$ such that*

$$(\forall M \in \mathbb{R}^{(2N_2)\times(2N_1)}) \quad f(M) = \sum_{k=1}^{2K} \varphi(\tilde{\lambda}_k) \tilde{u}_k \tilde{v}_k^\top \tag{5.6}$$

where $K = \min\{N_1, N_2\}$, $(\tilde{\lambda}_k)_{1 \leqslant k \leqslant 2K}$ are the singular values of M, and $\{\tilde{u}_k\}_{1 \leqslant k \leqslant 2K}$ / $\{\tilde{v}_k\}_{1 \leqslant k \leqslant 2K}$ are associated orthonormal families of left / right singular vectors in $\mathbb{R}^{2N_2}$ / $\mathbb{R}^{2N_1}$. Then f maps any matrix in $\mathcal{A}_{N_2,N_1}$ to a matrix in $\mathcal{A}_{N_2,N_1}$.

(xi) *The best approximation of maximum rank $R < \min\{N_1, N_2\}$ (in the sense of the Frobenius norm) to a matrix in $\mathcal{A}_{N_2,N_1}$ belongs to $\mathcal{A}_{N_2,N_1}$.*

(xii) *The projection onto the spectral ball of center 0 and radius $\rho \in]0, +\infty[$ of an ABBA matrix is an ABBA matrix.*

The proofs of these properties are provided in Appendix 5.8.2.

5.3.3 . Extension to feedforward networks

We will now extend the previous algebraic concepts by introducing the class of ABBA feedforward neural networks. In the following, the structure of an ABBA fully connected network will be presented from the perspective of investigating its links with standard networks. Such networks make use of weights that respect the structure of ABBA matrices, except for the first and the last layers. More precisely, the first layer maps the input to a twice-higher dimensional space, while the last layer performs a dimension reduction by a factor of 2.

[2] This means that (5.63) and (5.64) hold.

Definition 5.3.4 Let $m \in \mathbb{N} \setminus \{0\}$. $\widetilde{T}$ is an m-layer ABBA network if

$$\widetilde{T} = (\widetilde{W}_{m+1} \cdot + \widetilde{b}_{m+1}) \widetilde{T}_m \cdots \widetilde{T}_1 \widetilde{W}_0 \tag{5.7}$$

with $\widetilde{W}_0 \in \mathbb{R}^{(2N_0) \times N_0}$, $\widetilde{W}_{m+1} \in \mathbb{R}^{N_m \times (2N_m)}$, $\widetilde{b}_{m+1} \in \mathbb{R}^{N_m}$, and

$$(\forall i \in \{1, \ldots, m\}) \quad \widetilde{T}_i = \widetilde{R}_i(\widetilde{W}_i \cdot + \widetilde{b}_i) \tag{5.8}$$

$$\widetilde{R}_i \colon \mathbb{R}^{2N_i} \to \mathbb{R}^{2N_i}, \tag{5.9}$$

$$\widetilde{b}_i \in \mathbb{R}^{2N_i}, \tag{5.10}$$

$$\widetilde{W}_i \in \mathcal{A}_{N_i, N_{i-1}}, \tag{5.11}$$

for given positive integers $(N_i)_{0 \leqslant i \leqslant m}$. $\widetilde{T}$ is an m-layer non-negative ABBA network if it is an m-layer ABBA network as defined above and, for every $i \in \{1, \ldots, m\}$, the elements of $\widetilde{W}_i$ are non-negative.

In the remainder of this chapter, $\mathcal{N}_{m,\mathcal{A}}$ will designate the class of m-layer ABBA networks and $\mathcal{N}^+_{m,\mathcal{A}}$ will designate the subclass of m-layer non-negative ABBA networks. This latter subclass will be the main topic of investigation in this work. We will also use the notation $\mathcal{N}^+_{m,\mathcal{A}}(\rho)$ to designate the set of neural networks in $\mathcal{N}^+_{m,\mathcal{A}}$ where all the activation operators operate componentwise using the same function $\rho \colon \mathbb{R} \to \mathbb{R}$.

5.3.4 . Link with standard neural networks

In this section, we show that we can reshape Model 5.3.1 as a special case of a non-negative ABBA network. At each layer $i \in \{1, \ldots, m\}$ of this model, let $W_i^+ = (W^+_{i,k,\ell})_{1 \leqslant k \leqslant N_i, 1 \leqslant \ell \leqslant N_{i-1}} \in [0, +\infty[^{N_i \times N_{i-1}}$ be the positive part of matrix $W_i = (W_{i,k,\ell})_{1 \leqslant k \leqslant N_i, 1 \leqslant \ell \leqslant N_{i-1}}$, i.e.

$$(\forall k \in \{1, \ldots, N_i\})(\forall \ell \in \{1, \ldots, N_{i-1}\}) \qquad W^+_{i,k,\ell} = \begin{cases} W_{i,k,\ell} & \text{if } W_{i,k,\ell} > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{5.12}$$

Let $W_i^- = W_i^+ - W_i \in [0, +\infty[^{N_i \times N_{i-1}}$ be the negative part of W_i, where all the positive elements of W_i have been discarded. Let us now define a non-negative ABBA neural network by using these quantities.

Definition 5.3.5 Let $m \in \mathbb{N} \setminus \{0\}$. Let T be the feedforward neural defined in Model 5.3.1. $\widetilde{T}$ is a network in $\mathcal{N}^+_{m,\mathcal{A}}$ **associated with** T if it satisfies relations (5.7)-(5.11) with

$$\widetilde{W}_0 = \begin{bmatrix} I_{N_0} \\ -I_{N_0} \end{bmatrix}, \qquad \widetilde{W}_{m+1} = \frac{1}{2}[I_{N_m} \quad -I_{N_m}], \tag{5.13}$$

and

$$(\forall i \in \{1, \ldots, m\}) \quad \widetilde{R}_i \colon \begin{bmatrix} x \\ z \end{bmatrix} \mapsto \begin{bmatrix} R_i(x) \\ R_i(z) \end{bmatrix}, \tag{5.14}$$

$$\widetilde{W}_i = \begin{bmatrix} W_i^+ & W_i^- \\ W_i^- & W_i^+ \end{bmatrix}. \tag{5.15}$$

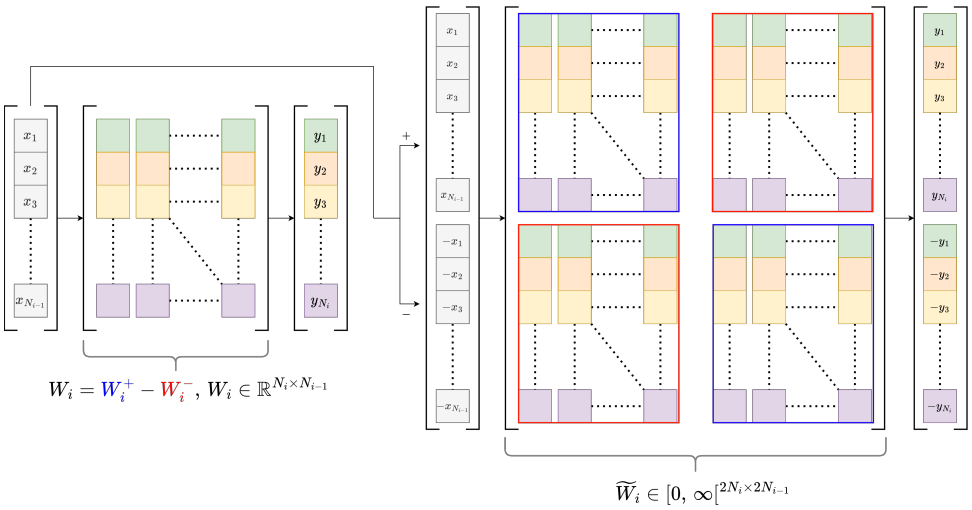


Figure 5.1: Equivalence between a standard fully-connected layer and its ABBA correspondent.

As we will show next, the main result is that, if the activation functions are symmetric, network $\tilde{T}$ defined above is identical to network T in terms of input-output relation, for judicious choices of the biases of $\tilde{T}$.

Proposition 5.3.6 *Let T be the m-layer feedforward network in Model 5.3.1. Assume that, for every $i \in \{1, \ldots, m\}$, the activation operator R_i in the i-th layer of T satisfies the symmetry relation (5.2) where $c_i \in \mathbb{R}^{N_i}$ and $d_i \in \mathbb{R}^{N_i}$. Let $\widetilde{T}$ be the neural network of $\mathcal{N}^+_{m,\mathcal{A}}$ associated with T whose bias vectors $(\widetilde{b}_i)_{1 \leqslant i \leqslant m}$ are linked to those $(b_i)_{1 \leqslant i \leqslant m}$ of T by the relations*

$$(\forall i \in \{1, \ldots, m\}) \quad \widetilde{b}_i = \begin{bmatrix} b_i - W_i^- d_{i-1} \\ c_i - b_i - W_i^+ d_{i-1} \end{bmatrix}, \tag{5.16}$$

$$\widetilde{b}_{m+1} = -\frac{d_m}{2}, \tag{5.17}$$

with $d_0 = 0$. Then, for every input, $\widetilde{T}$ delivers the same output as T.

The proof of this proposition is provided in Appendix 5.8.3. An illustration of the link between fully-connected layers and ABBA matrices is shown in Figure 5.1.

5.3.5 . Expressivity of non-negative ABBA networks

One of the main advantages of non-negative ABBA networks with respect to standard networks with non-negative weights is that they are universal approximators. More specifically, we have the following result.

Proposition 5.3.7 *Let $(n_e, n_r) \in (\mathbb{N} \setminus \{0\})^2$. Let $f : \mathbb{R}^{n_e} \to \mathbb{R}^{n_r}$ be a continuous function. Let $\mathbb{K}$ be any nonempty compact subset of $\mathbb{R}^{n_e}$ and let $\epsilon \in]0, +\infty[$.*

(i) *Let $\rho\colon \mathbb{R} \to \mathbb{R}$ be a symmetric non-polynomial activation function. There exists a network $\widetilde{T} \in \mathcal{N}^{+}_{1,\mathcal{A}}(\rho)$ with $N_0 = n_{\mathrm{e}}$ inputs and $N_2 = n_{\mathrm{r}}$ outputs such that*

$$(\forall x \in \mathbb{K}) \quad \|\widetilde{T}(x) - f(x)\| < \epsilon. \tag{5.18}$$

(ii) *Let $\rho\colon \mathbb{R} \to \mathbb{R}$ be a symmetric continuous activation function that is continuously differentiable around at least one point where its derivative is nonzero. Then there exists $m \geqslant 3$ and $\widetilde{T} \in \mathcal{N}^{+}_{m,\mathcal{A}}(\rho)$ with $N_0 = n_{\mathrm{e}}$ inputs, $N_{m+1} = n_{\mathrm{r}}$ outputs, and $2N_i = 2(n_{\mathrm{e}} + n_{\mathrm{r}} + 2)$ neurons in every layer $i \in \{1, \ldots, m\}$ such that (5.18) holds.*

Proof. Proposition 5.3.6 shows that non-negative ABBA networks can be as expressive as signed networks. Combining this fact with existing universal approximation results for signed networks (see [178] for (i) and [179] for (ii)) allows us to deduce these results. □

(i) addresses the case of shallow wide networks where the number of neurons in the hidden layer can be arbitrarily large, while (ii) corresponds to the case of deep networks having a limited number of neurons per layer. An illustration of these results is provided in Appendix 5.8.11.

5.3.6 . Lipschitz bounds for ABBA fully-connected networks

As mentioned in the previous sections, the robustness of neural networks with respect to adversarial perturbations can be evaluated through their Lipschitz constant. However, most of the existing techniques for computing a tight estimate of the constant have a high computational complexity for deep or wide networks, whereas simpler upper bounds may turn out to be over-pessimistic.

Nevertheless, in the context of non-negative weighted neural networks [20] proved that tight approximations to the Lipschitz constant can be achieved. In the following, we extend this result and show that we can derive a simple expression for the Lipschitz constant, using a separable bound, for non-negative ABBA networks.

Proposition 5.3.8 *Let $m \in \mathbb{N} \setminus \{0\}$ and let $\widetilde{T} \in \mathcal{N}^{+}_{m,\mathcal{A}}$ be given by (5.7)-(5.11). Assume that, for every $i \in \{1, \ldots, m-1\}$, $\widetilde{R}_i$ is a separable nonexpansive operator. A Lipschitz constant of $\widetilde{T}$ is*

$$\theta_m = \|\widetilde{W}_{m+1}\| \; \|\mathfrak{S}(\widetilde{W}_m) \cdots \mathfrak{S}(\widetilde{W}_1)\| \, \|\widetilde{W}_0\|. \tag{5.19}$$

The proof of this result is detailed in Appendix 5.8.4.

A standard separable upper bound for the Lipschitz constant [9] for the ABBA network $\widetilde{T}$ considered in the previous proposition is

$$\overline{\theta}_m = \|\widetilde{W}_{m+1}\| \, \|\widetilde{W}_m\| \cdots \|\widetilde{W}_1\| \, \|\widetilde{W}_0\|. \tag{5.20}$$

According to Proposition 5.3.3(vii), this bound reads also

$$\overline{\theta}_m = \|\widetilde{W}_{m+1}\| \, \|\mathfrak{S}(\widetilde{W}_m)\| \cdots \|\mathfrak{S}(\widetilde{W}_1)\| \, \|\widetilde{W}_0\|, \tag{5.21}$$

which, by simple norm inequalities, is looser than θ_m.

If T is the feedforward network defined in Model 5.3.1 and we apply Proposition 5.3.8 to the associated non-negative ABBA network $\widetilde{T}$ of Definition 5.3.5. We have

$$\|\widetilde{W}_0\| = \|\widetilde{W}_0^\top \widetilde{W}_0\|^{1/2} = \|2\, I_{N_0}\|^{1/2} = \sqrt{2} \tag{5.22}$$

and

$$\|\widetilde{W}_{m+1}\| = \|\widetilde{W}_{m+1}\widetilde{W}_{m+1}^\top\|^{1/2} = \frac{1}{\sqrt{2}}. \tag{5.23}$$

In turn, for every $i \in \{1, \dots, m\}$,

$$\mathfrak{S}(\widetilde{W}_i) = W_i^+ + W_i^- = |W_i|. \tag{5.24}$$

where $|W_i|$ is the matrice whose elements are the absolute values of those of W_i. Hence the Lipschitz constant of $\widetilde{T}$ in (5.19) reduces to

$$\theta_m = \||W_m| \dots |W_1|\|. \tag{5.25}$$

It then follows from Proposition 5.3.6 that θ_m is also a Lipschitz constant of T when using symmetric activation functions. Note that this bound was actually already derived in [20, Proposition 5.12].

5.4 . Convolutional networks

We will now extend the results presented in Section 5.3 to convolutional layers.

5.4.1 . ABBA convolutional layers

For any $i \in \{1, \dots, m\}$, $\mathcal{W}_i$ is a convolutional layer with $\zeta_{i-1} \in \mathbb{N} \setminus \{0\}$ input channels, ζ_i output channels, kernels $(w_{i,q,p})_{1 \leqslant p \leqslant \zeta_{i-1}, 1 \leqslant q \leqslant \zeta_i}$, and stride $s_i \in \mathbb{N} \setminus \{0\}$. The output $(y_q)_{1 \leqslant q \leqslant \zeta_i}$ of this layer (prior applying any activation operation) is linked to its input $(x_p)_{1 \leqslant p \leqslant \zeta_{i-1}}$ by

$$\begin{aligned} (\forall q \in \{1, \dots, \zeta_i\}) \quad u_q &= \sum_{p=1}^{\zeta_{i-1}} w_{i,q,p} * x_p \\ y_q &= (u_q)_{\downarrow_{s_i}}. \end{aligned} \tag{5.26}$$

Hereabove, for every $p \in \{1, \dots, \zeta_{i-1}\}$, $x_p = \big(x_p(\mathbf{n})\big)_{\mathbf{n} \in \mathbb{Z}^d}$ designates a d-dimensional discrete signal. Dimension $d = 1$ corresponds to 1D signals and $d = 2$ to images. A similar notation is used for other signals, in particular u_q and $w_{i,q,p}$ with $q \in \{1, \dots, \zeta_i\}$. The d-dimensional discrete convolution is denoted by

$*$ and $(\cdot)\downarrow_{s_i}$ is the decimation (or subsampling) by a factor s_i.

The ABBA convolutional layer $\widetilde{W}_i$ associated with $\mathcal{W}_i$ has twice the number of input channels and twice the number of output ones. More specifically, its input consists of ζ_{i-1} signals $(\widetilde{x}_p^+)_{1\leqslant p\leqslant\zeta_{i-1}}$ and ζ_{i-1} signals $(\widetilde{x}_p^-)_{1\leqslant p\leqslant\zeta_{i-1}}$. Similarly, its output consists of ζ_i signals $(\widetilde{y}_q^+)_{1\leqslant q\leqslant\zeta_i}$ and ζ_i signals $(\widetilde{y}_q^-)_{1\leqslant q\leqslant\zeta_i}$. To make the input-output relations more explicit, let us define the kernels $w_{i,q,p}^+$ and $w_{i,q,p}^-$ analogously to the fully connected case:

$$\begin{aligned}(\forall \mathbf{n}\in\mathbb{Z}^d)\quad w_{i,q,p}^+(\mathbf{n}) &= \begin{cases} w_{i,p,q}(\mathbf{n}) & \text{if } w_{i,p,q}(\mathbf{n})>0\\ 0 & \text{otherwise,}\end{cases}\\ w_{i,q,p}^-(\mathbf{n}) &= w_{i,q,p}^+(\mathbf{n}) - w_{i,p,q}(\mathbf{n}).\end{aligned} \tag{5.27}$$

Then the outputs of the ABBA layer are linked to its inputs by the relations

$$\begin{aligned}(\forall q\in\{1,\dots,\zeta_i\})\quad \widetilde{u}_q^+ &= \sum_{p=1}^{\zeta_{i-1}} w_{i,q,p}^+ * \widetilde{x}_p^+ + \sum_{p=1}^{\zeta_{i-1}} w_{i,q,p}^- * \widetilde{x}_p^-\\ \widetilde{u}_q^- &= \sum_{p=1}^{\zeta_{i-1}} w_{i,q,p}^- * \widetilde{x}_p^+ + \sum_{p=1}^{\zeta_{i-1}} w_{i,q,p}^+ * \widetilde{x}_p^-\\ \widetilde{y}_q^+ &= (\widetilde{u}_q^+)_{\downarrow_{s_i}}\\ \widetilde{y}_q^- &= (\widetilde{u}_q^-)_{\downarrow_{s_i}}.\end{aligned} \tag{5.28}$$

The above equations provide the general form of a convolutional ABBA layer when relaxing (5.27). An alternative formulation of convolutional layers in a matrix form, along with its correspondent d-dimensional spectral representation, is possible (see Appendix 5.8.5). This basically amounts to characterize layer (5.26) by the following matrices

$$(\forall \mathbf{n}\in\mathbb{Z}^d)\quad \boldsymbol{W}_i(\mathbf{n}) = \begin{bmatrix} w_{i,1,1}(\mathbf{n}) & \dots & w_{i,1,\zeta_{i-1}}(\mathbf{n})\\ \vdots & & \vdots\\ w_{i,\zeta_i,1}(\mathbf{n}) & \dots & w_{i,\zeta_i,c_{i-1}}(\mathbf{n})\end{bmatrix} \in \mathbb{R}^{\zeta_i\times\zeta_{i-1}}, \tag{5.29}$$

defining the so-called MIMO impulse response of $\mathcal{W}_i$, which plays a prominent role in dynamical system theory [180]. The MIMO impulse response of the ABBA layer $\widetilde{\mathcal{W}}_i$ is then characterized by ABBA matrices:

$$(\forall \mathbf{n}\in\mathbb{Z}^d)\quad \widetilde{\boldsymbol{W}}_i(\mathbf{n}) = \begin{bmatrix} \boldsymbol{W}_i^+(\mathbf{n}) & \boldsymbol{W}_i^-(\mathbf{n})\\ \boldsymbol{W}_i^-(\mathbf{n}) & \boldsymbol{W}_i^+(\mathbf{n})\end{bmatrix} \in [0,+\infty[^{(2\zeta_i)\times(2\zeta_{i-1})}, \tag{5.30}$$

where $\boldsymbol{W}_i^+(\mathbf{n}) = (w_{i,q,p}^+(\mathbf{n}))_{1\leqslant q\leqslant\zeta_i,1\leqslant p\leqslant\zeta_{i-1}} \in [0,+\infty[^{\zeta_i\times\zeta_{i-1}}$ and $\boldsymbol{W}_i^-(\mathbf{n}) = (w_{i,q,p}^-(\mathbf{n}))_{1\leqslant q\leqslant\zeta_i,1\leqslant p\leqslant\zeta_{i-1}} \in [0,+\infty[^{\zeta_i\times\zeta_{i-1}}$. This alternative view will be useful in the following sections.

5.4.2 . Lipschitz bounds for convolutional networks

In this section, we establish bounds on the Lipschitz constant of an m-layer convolutional neural network T [61]. Each linear operator $\mathcal{W}_i$ corresponding to layer $i \in \{1, \ldots, m\}$ will be defined by (5.26). We also define a variable

$$\sigma_i = \prod_{l=1}^{i} s_l \tag{5.31}$$

aggregating strides from layer 1 to layer i. Subsequently, we will assume that, for every $i \in \{1, \ldots, m\}$, the activation operators $(R_i)_{1 \leqslant i \leqslant m}$ are nonexpansive operators. Moreover, for every $i \in \{1, \ldots, m-1\}$, these operators are separable. By extending the standard definition for fully connected networks (see Appendix 5.8.1), this means that, for every $i \in \{1, \ldots, m-1\}$, there exists a function ρ_i from $\mathbb{R}$ to $\mathbb{R}$ such that

$$\begin{aligned} & (\forall \mathbf{x} \in \mathcal{H}_i) \quad \mathbf{y} = R_i(\mathbf{x}) \\ \Leftrightarrow \quad & (\forall p \in \{1, \ldots, c_i\})(\forall \mathbf{n} \in \mathbb{Z}^d) \quad y_p(\mathbf{n}) = \rho_i\big(x_p(\mathbf{n})\big). \end{aligned} \tag{5.32}$$

In Appendix 5.8.6, we derive frequency-based expressions allowing us to calculate bounds on the Lipschitz constant of T.

For accurate numerical evaluations, the frequency transform in these expressions has to be replaced by a Discrete Fourier Transform involving a significant number of frequency bins (e.g., 128^d). Due to this fact, a computation bottleneck occurs when MIMO filters are characterized by a large number of input/output channels (e.g., for 2D applications). In the following, we provide an alternative lower-complexity formulation for calculating bounds on the Lipschitz constant. In the case of non-negative kernels, we show that this bound is tight.

Theorem 5.4.1 *Let $(\sigma_i)_{1 \leqslant i \leqslant m}$ be the aggregated stride factors of network T, as defined by (5.31), and let*

$$\boldsymbol{W} = (\boldsymbol{W}_m)_{\uparrow \sigma_{m-1}} * \cdots * (\boldsymbol{W}_2)_{\uparrow \sigma_1} * \boldsymbol{W}_1 \tag{5.33}$$

where $(\boldsymbol{W}_i)_{1 \leqslant i \leqslant m}$ are the MIMO impulse responses of each layer of network T and, for every $i \in \{2, \ldots, m\}$, $(\boldsymbol{W}_i)_{\uparrow \sigma_{i-1}}$ is the interpolated sequence by a factor σ_{i-1} of $\boldsymbol{W}_i$ (see (5.92)). For every $\mathbf{j} \in \mathbb{S}(\sigma_m) = \{0, \ldots, \sigma_m - 1\}^d$, we define the following matrix:

$$\overline{\boldsymbol{W}}^{(\mathbf{j})} = \sum_{\mathbf{n} \in \mathbb{Z}^d} \boldsymbol{W}(\sigma_m \mathbf{n} + \mathbf{j}) \in [0, +\infty[^{\zeta_m \times \zeta_0}. \tag{5.34}$$

Then

$$\theta_m = \Big\| \sum_{\mathbf{j} \in \mathbb{S}(\sigma_m)} \overline{\boldsymbol{W}}^{(\mathbf{j})} \big(\overline{\boldsymbol{W}}^{(\mathbf{j})}\big)^\top \Big\|^{1/2} \tag{5.35}$$

is a lower bound on the Lipschitz constant estimate of network T. In addition, if for every $i \in \{1,\dots,m\}$, $p \in \{1,\dots,\zeta_{i-1}\}$, and $q \in \{1,\dots,\zeta_i\}$, $w_{i,q,p} = (w_{i,q,p}(\mathbf{n}))_{\mathbf{n}\in\mathbb{Z}^d}$ is a non-negative kernel, then θ_m is a Lipschitz constant of T.

The proof of Theorem 5.4.1 is given in Appendix 5.8.7.

The constant θ_m in (5.134) is actually equal to the one calculated in Appendix 5.8.6. The following majorization if thus obtained (see (5.97)):

$$\theta_m \leqslant \overline{\theta}_m = \|\mathcal{W}_m\| \cdots \|\mathcal{W}_1\|. \tag{5.36}$$

By applying Theorem 5.4.1 to each individual layer $(\mathcal{W}_i)_{1\leqslant i\leqslant m}$ assumed to be with non-negative kernels, we get the following expression for the upper-bound:

$$\overline{\theta}_m = \prod_{i=1}^{m} \Big\| \sum_{\mathbf{j}\in\mathbb{S}(s_i)} \overline{\boldsymbol{W}}_i^{(\mathbf{j})} (\overline{\boldsymbol{W}}_i^{(\mathbf{j})})^\top \Big\|^{1/2}, \tag{5.37}$$

where

$$(\forall i \in \{1,\dots,m\})(\forall \mathbf{j} \in \mathbb{S}(s_i)) \qquad \overline{\boldsymbol{W}}_i^{(\mathbf{j})} = \sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{W}_i(s_i\mathbf{n}+\mathbf{j}). \tag{5.38}$$

The bound $\overline{\theta}_m$ is generally more tractable than θ_m since it separates the influence of each layer and does not require to compute the global matrix sequence $\boldsymbol{W}$ as expressed by (5.33). However, such separable bounds are usually loose. According to our observations, it turns out that, in the special case of convolutional layers with non-negative kernels, θ_m and $\overline{\theta}_m$ are quite close (see numerical tests in Appendix 5.8.8).

To illustrate these results, the computation of the Lipschitz bound of a layer corresponding to an average pooling is presented as an example in Appendix 5.8.9.

5.4.3 . Bounds for ABBA convolutional networks

Let us extend the previous results to the ABBA context. The linear operators of the considered ABBA network $\widetilde{T}$ are denoted by $(\widetilde{\mathcal{W}}_i)_{0\leqslant i\leqslant m+1}$. The weights in $\widetilde{\mathcal{W}}_0$ and $\widetilde{\mathcal{W}}_{m+1}$ are signed, whereas $(\widetilde{\mathcal{W}}_i)_{1\leqslant i\leqslant m}$ are convolutional layers with d-dimensional non-negative kernels. More precisely, we assume that, for every $i \in \{1,\dots,m\}$, the i-th layer of the ABBA network has $2\zeta_{i-1}$ input channels, $2\zeta_i$ output channels, and stride $s_i \in \mathbb{N}\setminus\{0\}$. The MIMO impulse response of such a layer is of the form (5.30). We make the same assumptions of nonexpansiveness and separability for the activation operators as in the previous section. We recall that $(\boldsymbol{W})_{\uparrow\sigma}$ denotes the interpolated version by a factor σ of a MIMO impulse response $\boldsymbol{W} = \big(\boldsymbol{W}(\mathbf{n})\big)_{\mathbf{n}\in\mathbb{Z}^d}$.

The following result is then established in Appendix 5.8.10 :

Theorem 5.4.2 *Under the above assumptions on the convolutional ABBA network $\widetilde{T}$, let*

$$(\forall i \in \{1,\dots,m\})(\forall \mathbf{j} \in \mathbb{S}(s_i)) \quad \Omega_i^{(\mathbf{j})} = \sum_{\mathbf{n}\in\mathbb{Z}^d} \mathfrak{S}(\widetilde{\boldsymbol{W}}_i(s_i\mathbf{n}+\mathbf{j})) \in [0,+\infty[^{\zeta_i\times\zeta_{i-1}}, \tag{5.39}$$

where $(\widetilde{\boldsymbol{W}}_i(\mathbf{n}))_{n\in\mathbb{Z}}$ is the MIMO impulse response of the ABBA layer of index i. Then a Lipschitz constant of $\widetilde{T}$ is

$$\overline{\theta}_m = \|\widetilde{W}_{m+1}\| \Big(\prod_{i=1}^{m} \Big\| \sum_{\mathbf{j}\in\mathbb{S}(s_i)} \Omega_i^{(\mathbf{j})} \big(\Omega_i^{(\mathbf{j})}\big)^\top \Big\| \Big)^{1/2} \|\widetilde{W}_0\|, \tag{5.40}$$

where $\|\widetilde{W}_{m+1}\|$ (resp. $\|\widetilde{W}_0\|$) is the spectral norm of the linear operator employed in the last (resp. first layer).

The bound (5.40) will be subsequently used to control the Lipschitz constant of non-negative ABBA networks during their training.

5.5 . Lipschitz-constrained training

The theoretical bounds established in the previous sections provide a relatively easy way of computing a tight estimate of the global Lipschitz constant. We propose a simple approach to control it during the training phase. Since our networks contain mostly layers having non-negative weights and a few layers having arbitrary-signed weights, their Lipschitz constant will be controlled separately, and different constraint sets will be handled for each case.

To train a robust ABBA network, we employ a projected version of the well-known ADAM optimizer. Each layer i is parameterized by a vector Ψ_i. In the case of a dense layer, Ψ_i is a vector gathering the elements of the weight matrix $\widetilde{W}_i$, the components of the associated bias $\widetilde{b}_i$, and a possible additional parameter that will be introduced hereafter. For an ABBA layer, Ψ_i is thus a vector of dimension $2N_i(N_{i-1}+1)$ or $2N_i(N_{i-1}+1)+1$. In the context of a 2D convolutional layer, an array w_i of scalar convolutional kernels is substituted for the weight matrix. In the ABBA case, we have $2\zeta_i\zeta_{i-1}$ such kernels. To ensure nonnegativity (if needed) and Lipschitz bound conditions on the weight operator, we project Ψ_i onto a suitable closed and convex constraint set. Considering pairs $(z_k)_{1\leqslant k\leqslant K}$ of inputs images and their associated labels, the operations performed at each epoch $n>0$ to minimize a loss function ℓ are presented in Algorithm 6.

After each iteration t of the optimizer, we perform a projection $\mathrm{proj}_{\mathcal{S}_{i,t}}$ onto a constraint set $\mathcal{S}_{i,t}$ The definition of this set and the corresponding method for managing the projection is detailed in the following, according to the network type.

Algorithm 6: Projected ADAM Algorithm

Partition $\{1, \dots, K\}$ *into minibatches* $(\mathbb{L}_{q,n})_{1 \leqslant q \leqslant Q}$
$t = (n-1)Q + q$ # iteration index
sweep minibatches
foreach $q \in \{1, \dots, Q\}$ **do**
  foreach *layer* i **do**
    $g_{i,t} = \sum_{k \in \mathbb{M}_{q,n}} \nabla_i \ell\big(z_k, (\Psi_{i,t})_{1 \leqslant i \leqslant m}\big)$ # grad. computation
    $\mu_{i,t} = \beta_1 \mu_{i,t-1} + (1-\beta_1) g_{i,t}$ # classical ADAM updates
    $\nu_{i,t} = \beta_2 \nu_{i,t-1} + (1-\beta_2) g_{i,t}^2$
    $\gamma_t = \gamma \sqrt{1-\beta_2^t}/(1-\beta_1^t)$
    $\widetilde{\Psi}_{i,t} = \Psi_{i,t} - \gamma_t \mu_{i,t} / (\sqrt{\nu_{i,t}} + \epsilon)$
  foreach *layer* i **do**
    $\Psi_{i,t+1} = \mathrm{proj}_{\mathcal{S}_{i,t}}(\widetilde{\Psi}_{i,t})$ # projection step

Handling Lipschitz constants for fully-connected layers. Consider the network defined by Model (5.3.5). In the case of fully connected networks, the Lipschitz constant is given by Proposition 5.3.8, which basically splits the bound into three terms: the first and the last account for the starting and ending layers, respectively, while the middle one encompasses all the ABBA layers. For the two former arbitrary-signed layers, we control the Lipschitz constants individually during training, by imposing a bound on each weight matrix spectral norm. This defines the following two constraints:

$$(\forall i \in \{0, m+1\}) \quad \|\widetilde{W}_i\| \leqslant \overline{\theta}_{m,i}, \tag{5.41}$$

where $\overline{\theta}_{m,i}$ is the imposed Lipschitz bound for the i-th layer. To deal with this constraint, we decompose the weight matrix as $\widetilde{W}_i = \overline{\theta}_{m,i} \widetilde{W}'_i$, which yields the constraint set

$$(\forall i \in \{0, m+1\}) \quad \mathcal{S}_{i,t} = \{\widetilde{W}'_i \mid \|\widetilde{W}'_i\| \leqslant 1\}. \tag{5.42}$$

The projection onto $\mathcal{S}_{i,t}$ is performed by clipping the singular values of $\widetilde{W}'_i$ to 1.

In our proposed training procedure, we set $\overline{\theta}_{m,0}\overline{\theta}_{m,m+1} = 1$. This gives the network one degree of freedom to automatically adapt the value of the Lipschitz constant of these two layers. To do so, we adopt the following parametrization

$$\overline{\theta}_{m,0} = \exp(\alpha), \qquad \overline{\theta}_{m,m+1} = \exp(-\alpha), \tag{5.43}$$

where $\alpha \in \mathbb{R}$ is a trainable parameter. It constitutes an extra component of the vector Ψ_i when $i \in \{0, m+1\}$.

In the case of ABBA dense layers, we need to handle two requirements: ensure that, for every $i \in \{1, \dots, m\}$, $\widetilde{W}_i$ is a non-negative ABBA matrix, and to constrain the product of all the weight matrices to be such that $\|\widetilde{W}_m \cdots \widetilde{W}_1\| \leqslant \overline{\theta}_m$. Since $\overline{\theta}_{m,0}\overline{\theta}_{m,m+1} = 1$, $\overline{\theta}_m$ corresponds to the target Lipschitz bound for the ABBA network.

For every $i \in \{1, \dots, m\}$, $\widetilde{W}_i$ is parameterized by W_i^+ and W_i^-. We define the following two constraint sets:

$$\mathcal{D}_i = \{(W_i^+, W_i^-) \in (\mathbb{R}^{N_i \times N_{i-1}})^2 \mid W_i^+ \geqslant 0 \text{ and } W_i^- \geqslant 0\}, \tag{5.44}$$

$$\mathcal{C}_{i,t} = \left\{(W_i^+, W_i^-) \in (\mathbb{R}^{N_i \times N_{i-1}})^2 \mid \left\| A_{i,t} \begin{bmatrix} W_i^+ & W_i^- \\ W_i^- & W_i^+ \end{bmatrix} B_{i,t} \right\| \leqslant \overline{\theta}_m \right\}. \tag{5.45}$$

Here-above, matrix $A_{i,t}$ (resp. $B_{i,t}$) is an ABBA matrix, which is the product of the weight matrices for the posterior (resp. previous layers). In this case, $\mathcal{S}_{i,t} = \mathcal{D}_i \cap \mathcal{C}_{i,t}$. To perform the projection onto the intersection of these two sets, we use an instance of the proximal algorithm presented in [156], which alternates between elementary projections onto $\mathcal{D}_i$ and projections onto the spectral ball with center 0 and radius $\overline{\theta}_m$. Because of Proposition 5.3.3(xii), the latter projection allows us to keep the structure of ABBA matrices.

Handling Lipschitz constants for convolutional layers. In the case of convolutional ABBA networks, we derived the bound in (5.40) which consists of the product of $m+2$ terms. The Lipschitz bound constraint is managed by introducing auxiliary variables $(\overline{\theta}_{m,i})_{0 \leqslant i \leqslant m+1}$ defining upper bounds for each layer. At iteration t of the algorithm estimates $(\overline{\theta}_{m,i,t})_{0 \leqslant i \leqslant m+1}$ of the auxiliary bounds are updated. Similarly to the fully connected case, we use two different types of constraints.

For the i-th ABBA convolutional layer with $i \in \{1, \dots, m\}$, we consider the constraint set

$$\mathcal{C}_{i,t} = \{\mathcal{W}_i \mid \Big\| \sum_{\mathbf{j} \in \mathbb{S}(s_i)} \Omega_i^{(\mathbf{j})} \big(\Omega_i^{(\mathbf{j})}\big)^\top \Big\| \leqslant \overline{\theta}_{m,i,t}^2\} \tag{5.46}$$

where matrices $(\Omega_i^{(\mathbf{j})})_{\mathbf{j} \in \mathbb{S}(s_i)}$ are linked to the convolution kernels by the linear relation (5.39). By concatenating all these s_i^d matrices horizontally, we obtain a rectangular matrix Ω_i which allows us to reexpress (5.46) in the simpler form:

$$\mathcal{C}_{i,t} = \{\mathcal{W}_i \mid \|\Omega_i\| \leqslant \overline{\theta}_{m,i,t}\}. \tag{5.47}$$

We also have to impose the non-negativity of the filters alongside the stability bound. This corresponds to a constraint set $\mathcal{D}_i$. Projecting onto $\mathcal{S}_{i,t} = \mathcal{C}_{i,t} \cap \mathcal{D}_i$ is performed by using the same iterative proximal algorithm as previously.

For the first and the last layers, we impose similarly that $\|\mathcal{W}_i\| \leqslant \overline{\theta}_{m,i,t}$ with $i \in \{0, m+1\}$. Since the kernels are signed, we resort a frequency formulation

(see (5.111)) to estimate the spectral norm of the convolutional operator. The procedure we use is described in Appendix 5.8.12.

Convolutional layers are usually post-processed by an ABBA fully connected network. This part will be handled as explained previously. However, we need to set the upper bounds $(\overline{\theta}_{m,i,t})_{0\leqslant i\leqslant m+1}$ used in the convolutional part and the upper bound of the ABBA fully connected part. With a slight abuse of notation, let us denote this latter bound by $\overline{\theta}_{m,m+2,t}$, while the target Lipschitz constant for the global network is still denoted by $\overline{\theta}_m$. We have to deal with the following constraint:

$$\prod_{i=0}^{m+2} \overline{\theta}_{m,i,t} = \overline{\theta}_m. \tag{5.48}$$

We proceed by computing the Lipschitz constants $(\widetilde{\theta}_{m,i,t})_{0\leqslant i\leqslant m+2}$ of the layers after the ADAM update. Then, we set

$$(\forall i \in \{0,\dots,m+2\}) \quad \overline{\theta}_{m,i,t} = \widetilde{\theta}_{m,i,t} \left(\frac{\overline{\theta}_m}{\prod_{i'=0}^{m+2} \widetilde{\theta}_{m,i',t}} \right)^{\frac{1}{m+3}}, \tag{5.49}$$

which guarantees that (5.48) holds. Update (5.49) can be interpreted as the orthogonal projection onto the constraint set defined by (5.48) after a logarithmic transform of the auxiliary variables. The benefit of such a transform is to convexify the constraint.

5.6 . Experiments

In this section, we show the versatility of ABBA neural networks in solving classification tasks. The objective of our experiments is three-fold.

(i) First, we compare positive ABBA structures with their classic non-negative counterparts and check that our method yields significantly better results in all considered cases.

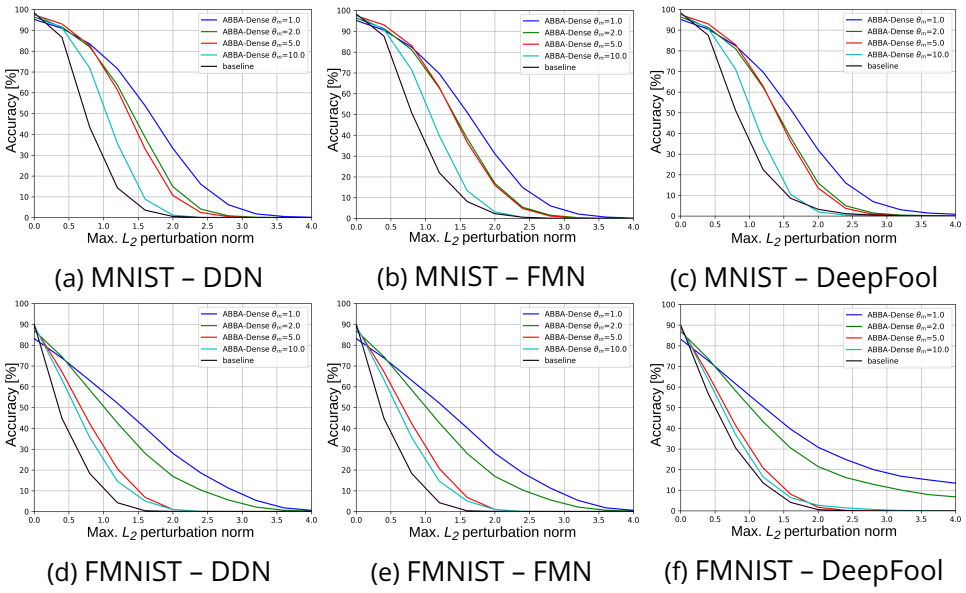
(ii) We then train ABBA models constrained to different Lipschitz bound values and evaluate their robustness against several adversarial attacks.

(iii) Finally, we compare our proposed approach with three other well-established defense strategies, namely *Adversarial Training* (AT), *Trade-off-inspired adversarial defense* (TRADES) [169], and *Deel-Lip* proposed by [64].

We validate our ABBA networks on four benchmark image classification datasets: MNIST, its more complex variant Fashion MNIST [3], a variant [4] of the Rock-Paper-Scissors (RPS) dataset [181], and a binary classification on CelebA [182]. For the

[3] https://github.com/zalandoresearch/fashion-mnist

[4] https://github.com/DrGFreeman/rps-cv



(a) MNIST – DDN (b) MNIST – FMN (c) MNIST – DeepFool

(d) FMNIST – DDN (e) FMNIST – FMN (f) FMNIST – DeepFool

Figure 5.2: *Accuracy vs. Perturbation for different Lipschitz constants – Dense Architecture*.

last dataset, inspired from [64] where eyeglass detection is performed, we specialized our models on a different attribute, namely to identify whether a person is bald or not. To explore the features of different ABBA network topologies, we experiment with two main types of ABBA architectures: one having only fully connected layers, further referred to as *ABBA Dense*, and another one which includes a convolutional part for feature extraction, followed by a fully connected classification module, *ABBA Conv*. Depending on the dataset, the particularities of each architecture are slightly different. A detailed description of all the small networks employed in this work, as well as other training details, are provided in Appendix 5.8.13. For all experiments, the input images were scaled in the $[-1, 1]$ interval.

ABBA networks vs. non-negative networks. First, we compare our non-negative ABBA networks with standard ones trained under non-negativity constraints. We consider standard neural networks to have the same number of parameters as their ABBA equivalent. The results are summarized in Table 5.1, indicating that ABBA neural networks yield far superior results, in all cases. In the case of fully connected architectures, the difference in terms of accuracy is around $\sim 3\%$ and $\sim 5\%$ for MNIST and FMNIST, respectively. The difference is even higher when we consider Conv architectures (e.g. $\sim 5\%$, $\sim 7\%$, $\sim 31\%$ and $\sim 32\%$ for MNIST, FMNIST, RPS, and CelebA, respectively). This shows that standard non-negative convolutional kernels are often suboptimal for extracting relevant information from image data. On the other hand, training standard neural networks having arbitrary-signed weights gives results very similar to their ABBA equivalents

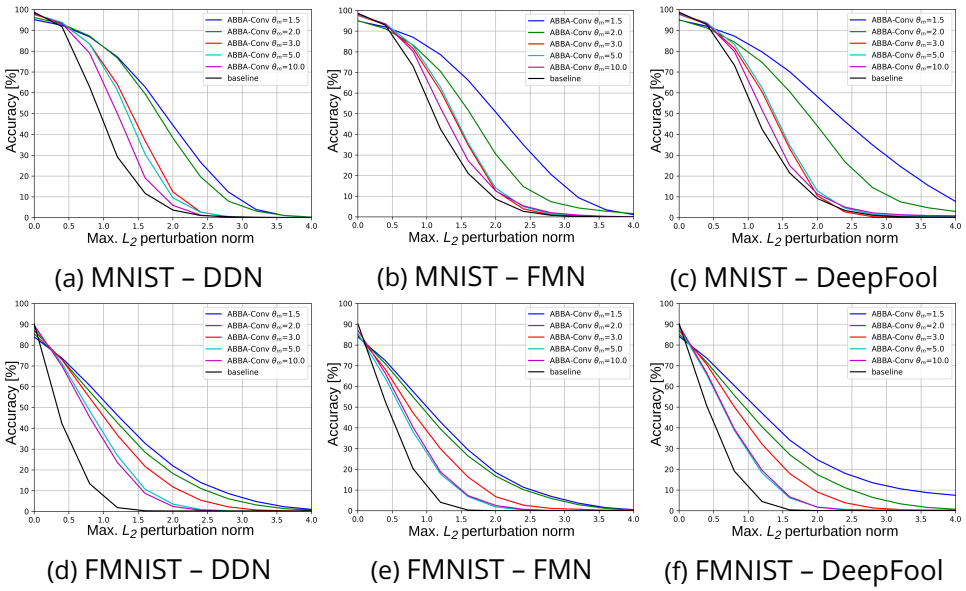
Dataset	Network	Architecture	Accuracy [%]
MNIST	ABBA	Dense	98.33
		Conv	98.70
	Non-Negative	Dense	94.95
		Conv	93.27
	Baseline	Dense	98.35
		Conv	98.68
FMNIST	ABBA	Dense	90.02
		Conv	90.17
	Non-Negative	Dense	84.56
		Conv	83.09
	Baseline	Dense	90.00
		Conv	90.20
RPS	ABBA	Conv	99.08
	Non-Negative	Conv	67.30
	Baseline	Conv	98.86
CelebA	ABBA	Conv	90.21
	Non-Negative	Conv	61.04
	Baseline	Conv	90.17

Table 5.1: Comparison between ABBA, full non-negative and arbitrary-signed (baseline) networks.

in all the cases, showing that ABBA networks do not suffer from these shortcomings. These results are in agreement with Proposition 5.3.6.

Stability vs. Performance. According to the "no free lunch" theorem [123], stability guarantees may impact the system performance on clean data. In this work, we train several models by following the approach described in Section 5.5. The Lipschitz constant of the network is varied in an effort to find the optimal trade-off between robustness and classification accuracy. This compromise is usually use-case specific, depending on the architecture complexity and on the dataset particularities, so the tightness of the imposed stability bound must be chosen accordingly. In our experiments, we limited the maximum Lipschitz constant we impose, so that the drop in performance does not exceed 5% of the baseline model accuracy (i.e., the model trained without robustness constraints).

Adversarial attack validation. We train several robust ABBA models by varying the global Lipschitz bound $\overline{\theta}_m$. We then evaluate their robustness against inputs corrupted with different levels of adversarial perturbations, by studying their influence on the overall performance of the system. The adversarial data is obtained by adding to the original image a perturbation. To create this adversarial noise, we use three white-box attackers, as described next. **DDN** [42] is a gradient-based ℓ_2



(a) MNIST – DDN (b) MNIST – FMN (c) MNIST – DeepFool

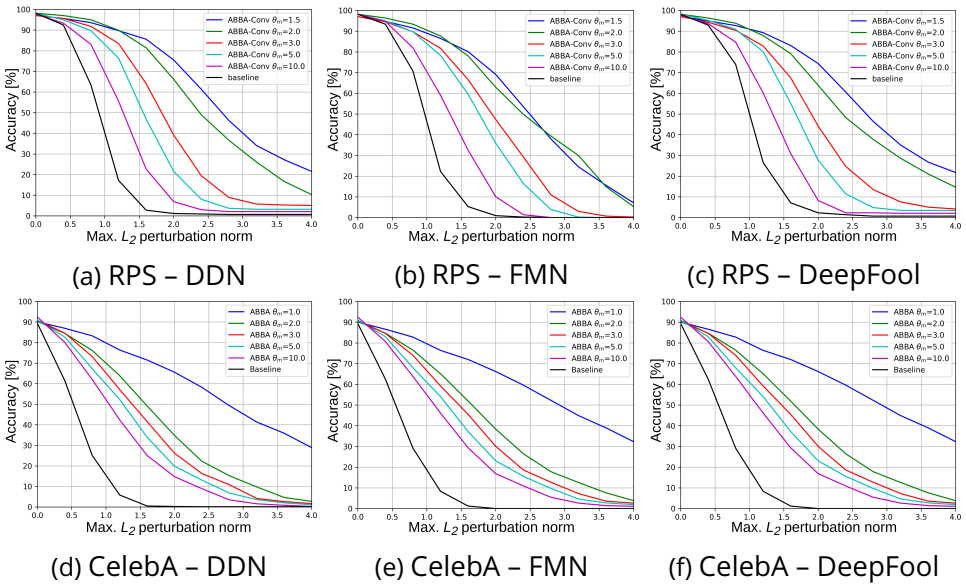
(d) FMNIST – DDN (e) FMNIST – FMN (f) FMNIST – DeepFool

Figure 5.3: *Acc. vs. Perturbation for different Lipschitz constants ABBA Conv-Dense Architectures – MNIST and FMNIST.*

adversarial attack method that seeks to decouple the direction and norm of the additive perturbation. By doing so, this attack is able to generate effective examples, while requiring fewer iterations than other methods. **DeepFool** [38] considers a linear approximation to the model and refines the attack sample iteratively, by selecting the point that would cross the decision boundary with minimal effort in the logit space. The **FMN** [43] attack improves the approach in DDN by introducing adaptive norm constraints on the perturbation, in order to balance the trade-off between the magnitude of the perturbation and the level of miss-classification. This results in a powerful attack that is able to generate adversarial examples with small perturbation levels.

For a given maximum ℓ_2 perturbation norm, we ran each attack for 300 steps, using the default hyperparameters for each of the three attackers. Although all images are normalized in the $[-1, 1]$ range, we report the robust accuracy w.r.t. ℓ_2 perturbation measured in $[0, 1]$ range, which is the common practice in the literature.

The results are summarized in Figures 5.2, 5.3, and 5.4 which show the robustness of MNIST, FMNIST, RPS, and CelebA ABBA models, w.r.t. increasing ℓ_2 norm perturbation, generated with DDN, FMN, and DeepFool attacks. A baseline model, trained without stability constraints and arbitrary-signed weights, is provided as a reference. These graphs could be interpreted as the expected performance of the model if the attack is allowed to influence the input image with an ℓ_2-norm less than ϵ, where the level of perturbation ϵ varies. For a better understanding of the



(a) RPS – DDN (b) RPS – FMN (c) RPS – DeepFool

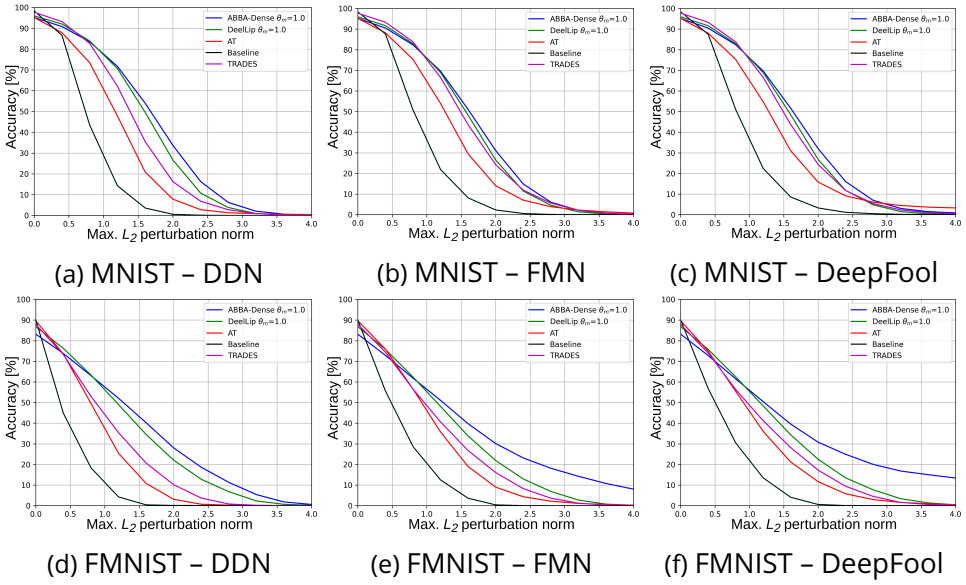
(d) CelebA – DDN (e) CelebA – FMN (f) CelebA – DeepFool

Figure 5.4: *Acc. vs. Perturbation for different Lipschitz constants ABBA Conv-Dense Architectures – RPS and CelebA.*

adversarial perturbation effect, some visual examples of the attacked inputs, for all the datasets, are presented below. For all datasets, adversarial examples created by using DDN attack are displayed in Figures 5.8, 5.9, 5.10, and 5.11. We generated adversarial samples using untargeted DDN attacks, with a budget of 300 iterations and initial parameters as proposed by the authors. We did not limit the maximum perturbation ϵ, in order to find the minimum one, allowing us to fool the model. It can be easily seen that for DeelLip and ABBA networks, the required perturbations for misclassification are higher. In particular, we observe that the perturbations needed to fool ABBA networks lead to severe artifacts in the images.

It can be observed that our robust ABBA models are significantly less affected by adversarial inputs than the undefended baseline. This demonstrates that carefully controlling the Lipschitz constant during training improves the network stability against adversarial attacks. Naturally, as the imposed bound gets lower, the system becomes more robust. Although the difference in robustness between similar values of the Lipschitz constant depends on the intrinsic structure of the dataset, our results show that a good trade-off between robustness and performance can be achieved in all cases.

Comparison with other defense strategies. In the following, we compare our method for training robust models using ABBA networks with other defense strategies. *Deel-Lip*, developed in [64], is a popular Lipschitz-based approach, which uses the spectral normalization of each layer to offer robustness certificates during training. *TRADES* [169] introduces a robustness regularization term into the train-



(a) MNIST – DDN (b) MNIST – FMN (c) MNIST – DeepFool

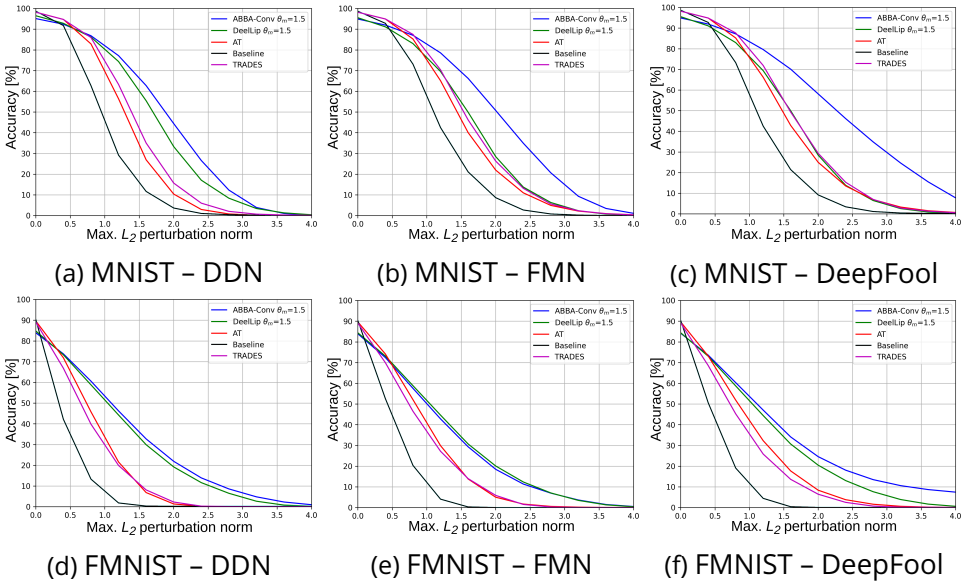
(d) FMNIST – DDN (e) FMNIST – FMN (f) FMNIST – DeepFool

Figure 5.5: *Comparisons with other defense techniques Dense Architectures*

ing objective. This regularizer encourages the network to have similar predictions on both the original input and its adversarial counterparts. On the other hand, *Adversarial Training* (AT) implies augmenting the training data with adversarial samples, increasing the network generalization capabilities to different input alterations. However, this technique offers weak theoretical stability guarantees, as it is mainly dependent on the strength of the adversary used during training.

For all experiments regarding AT, we used Projected Gradient Descent (PGD) attack to generate the adversarial samples with a perturbation level $\epsilon = 0.5$ and then we employed the scheduling strategy introduced by [66]. Concerning TRADES, we set $\lambda = 1$ for MNIST and FMNIST, and $\lambda = 1/2$ for RPS and CelebA datasets. For all the presented techniques, we considered the equivalent baseline to each ABBA network.

Comparisons, in the same adversarial set-up as before, are depicted in Figures 5.5, 5.6, and 5.7. We observe that using our theoretically certified Lipschitz bound yields models which are generally more robust than AT and TRADES. For simple datasets, such as MNIST and FMNIST, robust ABBA and Deel-lip models exhibit similar behavior for low-magnitude adversarial attacks, but as we increase the maximum perturbation ϵ, our method performs better. In the case of real-world datasets (RPS and CelebA), ABBA models exhibit high robustness properties against all the tested attacks, showing that our approach allows us to train neural models to reach great stability properties, without losing their generalization power.



(a) MNIST – DDN (b) MNIST – FMN (c) MNIST – DeepFool

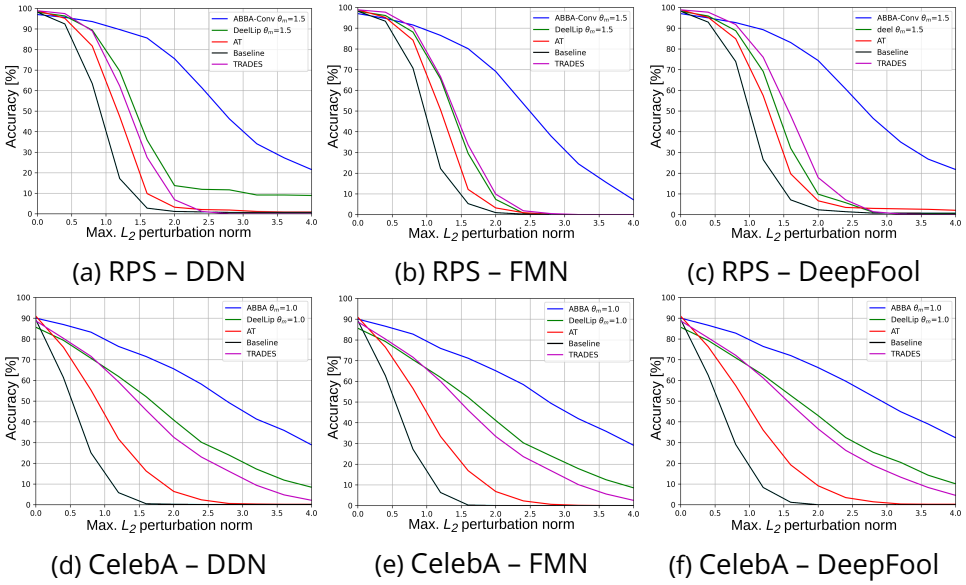
(d) FMNIST – DDN (e) FMNIST – FMN (f) FMNIST – DeepFool

Figure 5.6: *Comparisons with other defense techniques Conv-Dense Architectures – MNIST and FMNIST.*

Limitations. The main limitation of our method is that non-negative ABBA operators require more parameters to meet the universal approximation conditions. More precisely, for a given depth m and number of neurons $(N_i)_{0\leqslant i\leqslant m}$ per layer, a network $\widetilde{T} \in \mathcal{N}_{m,\mathcal{A}}$ has the same number of inputs and outputs as the standard feedforward network T in Model 5.3.1. All the layers of $\widetilde{T}$, except the first one, have however twice more inputs than T. Because of the ABBA structure of the weight matrices in (5.11), the maximum number of parameters of $\widetilde{T}$ is $2(N_0^2 + \sum_{i=1}^m N_i(N_{i-1}+1)+N_m^2)$ while the number of parameters of T is $\sum_{i=1}^m N_i(N_{i-1}+1)$. By storing $\widetilde{W}_1' = \widetilde{W}_1\widetilde{W}_0 \in \mathbb{R}^{(2N_1)\times N_0}$ instead of $\widetilde{W}_1$ and $\widetilde{W}_0$ separately, the maximum number of parameters is reduced to $2(\sum_{i=1}^m N_i(N_{i-1}+1)+N_m^2)$. Moreover, since the weights are non-negative, the model does not necessarily require signed representations' storage, so the memory space occupied by $\widetilde{T}$ could also be reduced. While our method does not provide any certification regarding the accuracy of the classifier in adversarial environments, it delivers a certified value for the Lipschitz constant of the network.

5.7 . Conclusions

In this chapter, we introduce ABBA networks, a novel class of neural networks where the majority of weights are non-negative. We demonstrate that these networks are universal approximators, possessing all the expressive properties of conventional signed neural architectures. Additionally, we unveil their remarkable algebraic characteristics, enabling us to derive precise Lipschitz bounds for both



(a) RPS – DDN (b) RPS – FMN (c) RPS – DeepFool

(d) CelebA – DDN (e) CelebA – FMN (f) CelebA – DeepFool

Figure 5.7: *Comparisons with other defense techniques Conv-Dense Architectures – RPS and CelebA.*

fully connected and convolutive operators.

Leveraging these bounds, we construct robust neural networks suitable for various classification tasks. For future research, it would be intriguing to explore the application of ABBA networks in regression problems, where controlling the Lipschitz constant may present more challenges. Moreover, extending our theoretical bounds to different structures, such as recurrent or attention-based networks, holds promise for further advancements.

Finally, we recognize the necessity of investigating the scalability of the proposed training method to deep architectures. One of the main hurdles in this endeavour is the increased number of parameters that deep ABBA architectures entail.

Figure 5.8: Adversarial examples with DDN attack for Conv-Dense models, on MNIST dataset. ℓ_2 perturbation magnitude is given in the top-left corner.

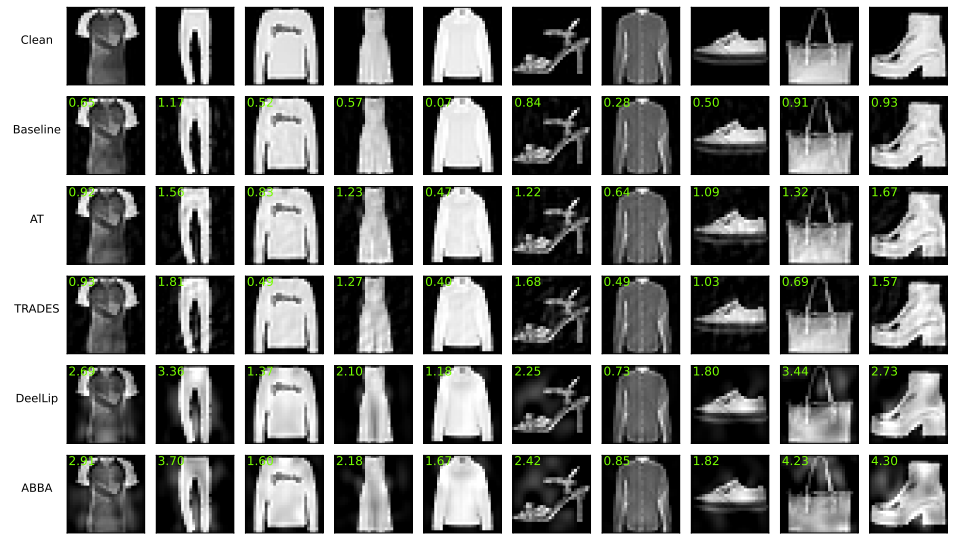


Figure 5.9: Adversarial examples with DDN attack for Conv-Dense models, on FMNIST dataset. ℓ_2 perturbation magnitude is given in the top-left corner.

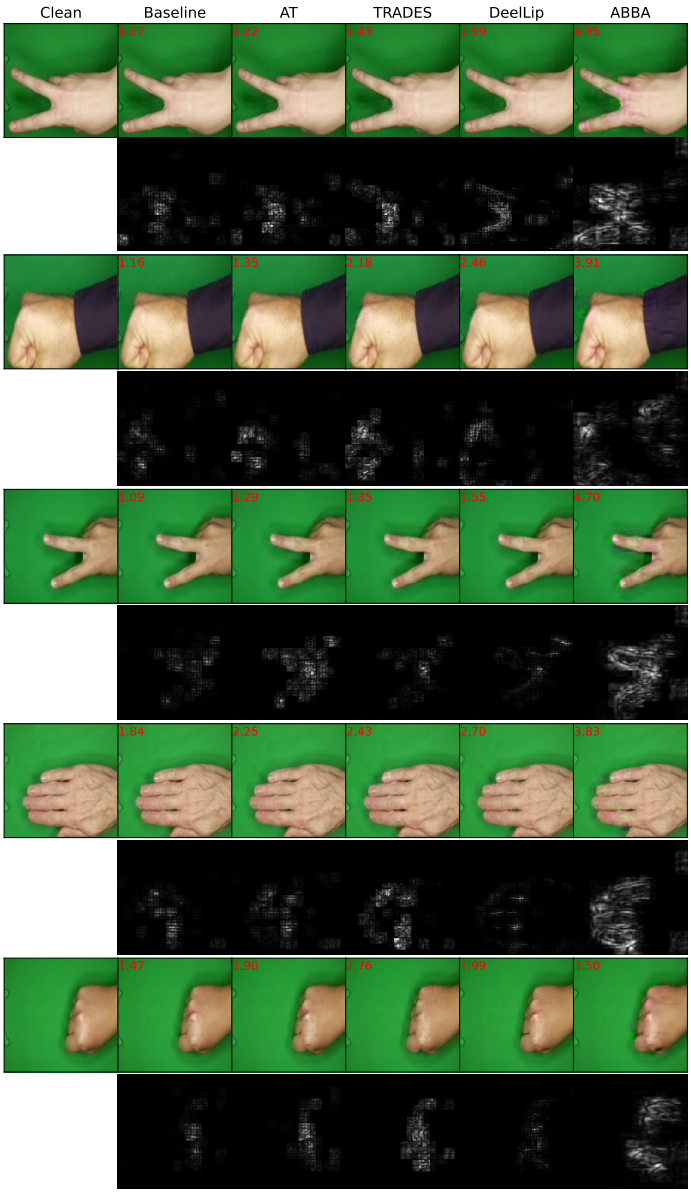


Figure 5.10: Adversarial examples generated with DDN, on RPS dataset. For each example: first row – adversarial images; second row – pixel differences between adversarial and clean sample.

Figure 5.11: Adversarial examples with DeepFool attack for CelebA. ℓ_2 perturbation magnitude is given in the top-left corner.

5.8 . Appendix

5.8.1 . Symmetric activation functions

In practice, the activation operator R_i is often separable, that is

$$\big(\forall x = (\xi_k)_{1\leqslant k\leqslant N_i} \in \mathbb{R}^{N_i}\big) \quad R_i x = \big(\varrho_i(\xi_k)\big)_{1\leqslant k\leqslant N_i}, \tag{5.50}$$

where, for every $k \in \{1, \dots, N_i\}$, $\varrho_i \colon \mathbb{R} \to \mathbb{R}$. Examples of odd functions allowing us to define symmetric separable activation operators R_i with $c_i = d_i = 0$ are

- the hyperbolic tangent activation function $\rho_i = \tanh$
- the arctangent activation function $\rho_i = (2/\pi)\arctan$
- the inverse square root linear unit function $\varrho_i \colon \mathbb{R} \to \mathbb{R} \colon \xi \mapsto \xi/\sqrt{1+\xi^2}$
- the Elliot activation function $\varrho_i \colon \mathbb{R} \to \mathbb{R} \colon \xi \mapsto \xi/(1+|\xi|)$.

Some examples of separable activation operators which are non-odd are described below. The capped ReLU function is given by

$$(\forall \xi \in \mathbb{R}) \quad \rho_i(\xi) = \begin{cases} 0 & \text{if } \xi < 0 \\ \xi & \text{if } 0 \leqslant \xi < \chi \\ \chi & \text{otherwise,} \end{cases} \tag{5.51}$$

where $\chi \in]0, +\infty[$. We have then $c_i = d_i = \chi 1_{N_i}$ with $1_{N_i} = [1, \dots, 1]^\top \in \mathbb{R}^{N_i}$. We can also define a leaky version of this function as

$$(\forall \xi \in \mathbb{R}) \qquad \rho_i(\xi) = \begin{cases} \alpha\xi & \text{if} \quad \xi < 0, \\ \xi & \text{if} \quad 0 \leqslant \xi < \chi, \\ \alpha(\xi - \chi) + \chi & \text{otherwise,} \end{cases} \tag{5.52}$$

where $\alpha \in]0, 1[$ and $\chi \in]0, +\infty[$ are hyperparameters.

5.8.2 . Proof of the properties of ABBA matrices

(i)-(iii): These properties follow from basic algebra. We will just detail the proof of the third one. Let

$$M_1 = \begin{bmatrix} A_1 & B_1 \\ B_1 & A_1 \end{bmatrix} \quad \text{and} \quad M_2 = \begin{bmatrix} A_2 & B_2 \\ B_2 & A_2 \end{bmatrix}, \tag{5.53}$$

where $(A_1, B_1) \in \mathbb{R}^{N_2 \times N_1}$ and $(A_2, B_2) \in \mathbb{R}^{N_3 \times N_2}$. Then

$$M_2 M_1 = \begin{bmatrix} A_2A_1 + B_2B_1 & A_2B_1 + B_2A_1 \\ A_2B_1 + B_2A_1 & A_2A_1 + B_2B_1 \end{bmatrix} \in \mathcal{A}_{N_3, N_1}. \tag{5.54}$$

In addition,

$$\begin{aligned}\mathfrak{S}(M_2M_1) &= A_2A_1 + B_2B_1 + A_2B_1 + B_2A_1\\ &= (A_2 + B_2)(A_1 + B_1)\\ &= \mathfrak{S}(M_2)\mathfrak{S}(M_1).\end{aligned} \tag{5.55}$$

(iv): This property is a direct consequence of (ii) and (iii).
(v): Let $M = \begin{bmatrix} A & B \\ B & A \end{bmatrix} \in \mathbb{R}^{(2N_1)\times(2N_1)}$. $\lambda \in \mathbb{C}$ is an eigenvalue of M if and only if

$$\det(M - \lambda \operatorname{Id}) = 0 \quad \Leftrightarrow \quad \det\left(\begin{bmatrix} A - \lambda \operatorname{Id} & B \\ B & A - \lambda \operatorname{Id} \end{bmatrix}\right) = 0. \tag{5.56}$$

We have

$$\begin{bmatrix} A - \lambda \operatorname{Id} & B \\ B & A - \lambda \operatorname{Id} \end{bmatrix} \begin{bmatrix} \operatorname{Id} & -\operatorname{Id} \\ \operatorname{Id} & \operatorname{Id} \end{bmatrix} = \begin{bmatrix} A + B - \lambda \operatorname{Id} & -A + B + \lambda \operatorname{Id} \\ A + B - \lambda \operatorname{Id} & A - B - \lambda \operatorname{Id} \end{bmatrix}. \tag{5.57}$$

Since $A - B - \lambda \operatorname{Id}$ and $-A + B + \lambda \operatorname{Id}$ commute, we have [183]

$$\det\left(\begin{bmatrix} A + B - \lambda \operatorname{Id} & -A + B + \lambda \operatorname{Id} \\ A + B - \lambda \operatorname{Id} & A - B - \lambda \operatorname{Id} \end{bmatrix}\right) = 2^N \det\big((A+B-\lambda \operatorname{Id})(A-B-\lambda \operatorname{Id})\big). \tag{5.58}$$

Similarly

$$\det\left(\begin{bmatrix} \operatorname{Id} & -\operatorname{Id} \\ \operatorname{Id} & \operatorname{Id} \end{bmatrix}\right) = 2^N. \tag{5.59}$$

We deduce from (5.57) that

$$\begin{aligned}&\det\left(\begin{bmatrix} A - \lambda \operatorname{Id} & B \\ B & A - \lambda \operatorname{Id} \end{bmatrix}\right) = \det\big((A + B - \lambda \operatorname{Id})(A - B - \lambda \operatorname{Id})\big)\\ \Leftrightarrow \quad &\det(M - \lambda \operatorname{Id}) = \det(A + B - \lambda \operatorname{Id}) \det(A - B - \lambda \operatorname{Id}).\end{aligned} \tag{5.60}$$

So λ is an eigenvalue of M if and only if $\det(A + B - \lambda \operatorname{Id}) = 0$ or $\det(A - B - \lambda \operatorname{Id}) = 0$, i.e., λ is an eigenvalue of $A + B$ or $A - B$.
(vi) Let M be defined similarly to previously with $(A, B) \in (\mathbb{R}^{N_2\times N_1})^2$. We have

$$\|M\| = \|MM^\top\|^{1/2} = \left\|\begin{bmatrix} AA^\top + BB^\top & AB^\top + BA^\top \\ AB^\top + BA^\top & AA^\top + BB^\top \end{bmatrix}\right\|^{1/2} \tag{5.61}$$

According to (v), the eigenvalues of $MM^\top \in \mathcal{A}_{N_2,N_2}$ are those of $AA^\top + BB^\top + AB^\top + BA^\top = (A + B)(A + B)^\top$ and $AA^\top + BB^\top - AB^\top - BA^\top = (A - B)(A - B)^\top$. The maximum eigenvalues of the two latter matrices are $\|A + B\|^2$ and $\|A - B\|^2$, respectively. Therefore $\|M\| = \max\{\|A + B\|, \|A - B\|\}$.

(vii): In addition, if A and B have nonnegative elements,

$$\begin{aligned}\|A-B\| &= \sup_{x\in\mathbb{R}^N\setminus\{0\}} \frac{\|Ax-Bx\|}{\|x\|}\\ &\leqslant \sup_{x\in\mathbb{R}^N\setminus\{0\}} \frac{\||A|x|+B|x|\|}{\|x\|}\\ &= \sup_{a\in[0,+\infty[^N\setminus\{0\}} \frac{\|Aa+Ba\|}{\|a\|}\\ &\leqslant \|A+B\|, \end{aligned} \tag{5.62}$$

where $|x|$ denotes the vector whose components are the absolute values of those of vector x. We deduce from (vi) that $\|M\| = \|A+B\| = \|\mathfrak{S}(M)\|$.
(viii): We have

$$A+B = \sum_{k=1}^{K} \lambda_k u_k v_k^\top \tag{5.63}$$

$$A-B = \sum_{k=1}^{K} \mu_k t_k w_k^\top. \tag{5.64}$$

Thus

$$A = \frac{1}{2}\sum_{k=1}^{K} (\lambda_k u_k v_k^\top + \mu_k t_k w_k^\top) \tag{5.65}$$

$$B = \frac{1}{2}\sum_{k=1}^{K} (\lambda_k u_k v_k^\top - \mu_k t_k w_k^\top) \tag{5.66}$$

and we deduce that

$$\begin{aligned}\begin{bmatrix} A & B\\ B & A\end{bmatrix} &= \sum_{k=1}^{K} \frac{1}{2}\left(\lambda_k \begin{bmatrix} u_k v_k^\top & u_k v_k^\top\\ u_k v_k^\top & u_k v_k^\top\end{bmatrix} + \mu_k \begin{bmatrix} t_k w_k^\top & -t_k w_k^\top\\ -t_k w_k^\top & t_k w_k^\top\end{bmatrix}\right)\\ &= \sum_{k=1}^{K} \frac{1}{2}\left(\lambda_k \begin{bmatrix} u_k\\ u_k\end{bmatrix}\begin{bmatrix} v_k\\ v_k\end{bmatrix}^\top + \mu_k \begin{bmatrix} t_k\\ -t_k\end{bmatrix}\begin{bmatrix} w_k\\ -w_k\end{bmatrix}^\top\right). \end{aligned} \tag{5.67}$$

On the other hand, for every $(k,\ell)\in\{1,\dots,K\}^2$,

$$\begin{aligned}\begin{bmatrix} u_k\\ u_k\end{bmatrix}^\top \begin{bmatrix} u_\ell\\ u_\ell\end{bmatrix} &= 2u_k^\top u_\ell = \begin{cases} 2 & \text{if } k=\ell\\ 0 & \text{otherwise,}\end{cases}\\ \begin{bmatrix} t_k\\ -t_k\end{bmatrix}^\top \begin{bmatrix} t_\ell\\ -t_\ell\end{bmatrix} &= 2t_k^\top t_\ell = \begin{cases} 2 & \text{if } k=\ell\\ 0 & \text{otherwise,}\end{cases}\\ \begin{bmatrix} u_k\\ u_k\end{bmatrix}^\top \begin{bmatrix} t_\ell\\ -t_\ell\end{bmatrix} &= 0, \end{aligned} \tag{5.68}$$

which shows that $\left\{ \frac{1}{\sqrt{2}} \begin{bmatrix} u_k \\ u_k \end{bmatrix}, \frac{1}{\sqrt{2}} \begin{bmatrix} t_k \\ -t_k \end{bmatrix} \right\}_{1 \leqslant k \leqslant K}$ is an orthonormal family of $\mathbb{R}^{2N_2}$. For similar reasons, $\left\{ \frac{1}{\sqrt{2}} \begin{bmatrix} v_k \\ v_k \end{bmatrix}, \frac{1}{\sqrt{2}} \begin{bmatrix} w_k \\ -w_k \end{bmatrix} \right\}_{1 \leqslant k \leqslant K}$ is an orthonormal family of $\mathbb{R}^{2N_1}$. This allows us to conclude that (5.67) provides a singular value decomposition of $\begin{bmatrix} A & B \\ B & A \end{bmatrix}$.

(ix): The rank of $\begin{bmatrix} A & B \\ B & A \end{bmatrix}$ is equal to the number of its nonzero singular values. From the previous result, it is thus equal to the sum of the nonzero values of $A+B$ and those of $A-B$, that is the sum of the ranks of matrices $A+B$ and $A-B$.

(x): The fact that the ABBA structure is kept by matrix mappings operating elementwise is obvious. Let us thus focus on the case of spectral functions. By using the same notation as in (viii), it follows from (5.67) that

$$f\left(\begin{bmatrix} A & B \\ B & A \end{bmatrix} \right) = \sum_{k=1}^{K} \frac{1}{2} \left(\varphi(\lambda_k) \begin{bmatrix} u_k v_k^\top & u_k v_k^\top \\ u_k v_k^\top & u_k v_k^\top \end{bmatrix} + \varphi(\mu_k) \begin{bmatrix} t_k w_k^\top & -t_k w_k^\top \\ -t_k w_k^\top & t_k w_k^\top \end{bmatrix} \right) = \begin{bmatrix} \tilde{A} & \tilde{B} \\ \tilde{B} & \tilde{A} \end{bmatrix}, \tag{5.69}$$

where

$$\begin{aligned} \tilde{A} + \tilde{B} &= \sum_{k=1}^{N} \varphi(\lambda_k) u_k v_k^\top \\ \tilde{A} - \tilde{B} &= \sum_{k=1}^{N} \varphi(\mu_k) t_k w_k^\top . \end{aligned} \tag{5.70}$$

(xi): By using the same notation as in (5.6), The best approximation of rank less than or equal to R to a matrix M_0 in $\mathbb{R}^{(2N_2)\times(2N_1)}$ is $f(M_0)$ where f is given by (5.6) with

$$(\forall \lambda \in \mathbb{R}_+) \quad \varphi(\lambda) = \begin{cases} \lambda & \text{if } \lambda \leqslant \tilde{\lambda}_{0,[R]} \\ 0 & \text{otherwise,} \end{cases} \tag{5.71}$$

and $\tilde{\lambda}_{0,[R]}$ is the R-th eigenvalue of M_0 when these are ordered by decreasing value: $\tilde{\lambda}_{0,1} \geqslant \ldots \geqslant \tilde{\lambda}_{0,K}$. It thus follows from (x) that if $M_0 \in \mathcal{A}_{N_2,N_1}$, then $f(M_0) \in \mathcal{A}_{N_2,N_1}$.

(xii): The projection onto the spectral ball of center 0 and radius $\rho \in]0, +\infty[$ of a matrix $M \in \mathcal{A}_{N_2,N_1}$ is given by (5.6) where

$$(\forall \xi \in \mathbb{R}) \qquad \varphi(\xi) = \min\{\xi, \rho\}.$$

The result then follows from Property (x).

Remark 5.8.1 The last result can be generalized as follows. Let $\psi \colon \mathbb{R} \to]-\infty, +\infty]$ be a lower-semicontinuous function, which is proper, even, and

convex, and let

$$\begin{aligned} g\colon \mathbb{R}^{(2N_2)\times(2N_1)} &\to \left]-\infty,+\infty\right] \\ M &\mapsto \sum_{i=1}^{2K} \psi(\tilde{\lambda}_k) \end{aligned} \tag{5.72}$$

where $K = \min\{N_1, N_2\}$ and $(\tilde{\lambda}_k)_{1\leqslant k\leqslant 2K}$ are the singular values of M. The proximity operator of g at $M \in \mathbb{R}^{(2N_2)\times(2N_1)}$ is [149, Proposition 24.68]:

$$\begin{aligned} \operatorname{prox}_g\colon M &\mapsto \operatorname*{argmin}_{P\in\mathbb{R}^{(2N_2)\times(2N_1)}} \frac{1}{2}\|P-M\|_{\mathrm{F}}^2 + g(P) \\ &= \sum_{k=1}^{2K} \operatorname{prox}_{\psi}(\tilde{\lambda}_k)\tilde{u}_k\tilde{v}_k^\top, \end{aligned} \tag{5.73}$$

where $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm. It then follows from Property (x) that, if $M \in \mathcal{A}_{N_2,N_1}$, then $\operatorname{prox}_g(M) \in \mathcal{A}_{N_2,N_1}$.

5.8.3 . Proof of Proposition 5.3.6

For every $i \in \{1,\dots,m\}$, let $x_i = T_i(x_{i-1})$ where $x_0 \in \mathbb{R}^{N_0}$ is an arbitrary input of network T and $x_m \in \mathbb{R}^{N_m}$ its corresponding output. By using the symmetry properties of the activation operators, for every $i \in \{1,\dots,m\}$, we have

$$\begin{aligned} (\forall i \in \{1,\dots,m\}) \quad x_i &= R_i(W_i x_{i-1} + b_i) \\ &= R_i(W_i^+ x_{i-1} - W_i^- x_{i-1} + b_i), \end{aligned} \tag{5.74}$$

$$\begin{aligned} -x_i &= -R_i(W_i x_{i-1} + b_i) \\ &= R_i(-W_i x_{i-1} - b_i + c_i) - d_i \\ &= R_i(W_i^- x_{i-1} - W_i^+ x_{i-1} - b_i + c_i) - d_i. \end{aligned} \tag{5.75}$$

By making use of notation (5.14), (5.74) and (5.75) can be rewritten more concisely as

$$(\forall i \in \{1,\dots,m\}) \quad \begin{bmatrix} x_i \\ -x_i \end{bmatrix} = \widetilde{R}_i\Big(\begin{bmatrix} W_i^+ & W_i^- \\ W_i^- & W_i^+ \end{bmatrix} \begin{bmatrix} x_{i-1} \\ -x_{i-1} \end{bmatrix} + \begin{bmatrix} b_i \\ c_i - b_i \end{bmatrix} \Big) - \begin{bmatrix} 0 \\ d_i \end{bmatrix}. \tag{5.76}$$

Let us define, for every $i \in \{0,\dots,m\}$,

$$\widetilde{x}_i = \begin{bmatrix} x_i \\ -x_i + d_i \end{bmatrix}. \tag{5.77}$$

Altogether (5.16), (5.11), (5.15), and (5.76) yield

$$\begin{aligned} (\forall i \in \{1,\dots,m\}) \quad \widetilde{x}_i &= \widetilde{R}_i \left(\begin{bmatrix} W_i^+ & W_i^- \\ W_i^- & W_i^+ \end{bmatrix} \Big(\widetilde{x}_{i-1} - \begin{bmatrix} 0 \\ d_{i-1} \end{bmatrix} \Big) + \begin{bmatrix} b_i \\ c_i - b_i \end{bmatrix} \right) \\ &= \widetilde{R}_i\Big(\widetilde{W}_i \widetilde{x}_{i-1} + \widetilde{b}_i \Big). \end{aligned} \tag{5.78}$$

This shows that, if $(\widetilde{T}_i)_{1\leqslant i\leqslant m}$ are given by (5.8), $\widetilde{x}_m = \widetilde{T}_m \cdots \widetilde{T}_1(\widetilde{x}_0)$. By using the forms of $\widetilde{W}_0$ and $\widetilde{W}_{m+1}$ in (5.13), we deduce that

$$x_m = \widetilde{W}_{m+1}\widetilde{x}_m + \widetilde{b}_{m+1} = \widetilde{T}(x_0). \tag{5.79}$$

5.8.4 . Proof for Proposition 5.3.8

According to [16] [20, Proposition 5.5],

$$\vartheta_m = \sup_{\substack{\Lambda_1 \in \mathscr{D}_{\{-1,1\}}^{2N_1} \\ \vdots \\ \Lambda_m \in \mathscr{D}_{\{-1,1\}}^{2N_m}}} \|\widetilde{W}_{m+1}\Lambda_m\widetilde{W}_m \cdots \Lambda_1\widetilde{W}_1\widetilde{W}_0\|. \tag{5.80}$$

where, for every $i \in \{1,\dots,m\}$, $\mathscr{D}_{\{-1,1\}}^{2N_i}$ designates the space of diagonal matrices of size $(2N_i)\times(2N_i)$ with diagonal entries equal to -1 or 1. For every $(\Lambda_1,\dots,\Lambda_m) \in \mathscr{D}_{\{-1,1\}}^{2N_1} \times \cdots \times \mathscr{D}_{\{-1,1\}}^{2N_m}$,

$$\|\widetilde{W}_{m+1}\Lambda_m\widetilde{W}_m \cdots \Lambda_1\widetilde{W}_1\widetilde{W}_0\| \leqslant \|\widetilde{W}_{m+1}\|\|\widetilde{W}_m\Lambda_{m-1}\widetilde{W}_{m-1} \cdots \Lambda_2\widetilde{W}_2\Lambda_1\widetilde{W}_1\|\|\widetilde{W}_0\|. \tag{5.81}$$

On the other hand, for every $i \in \{1,\dots,m\}$, $\widetilde{W}_i \in [0,+\infty[^{(2N_i)\times(2N_{i-1})}$. It then follows from [20, Proposition 5.10] that

$$\sup_{\substack{\Lambda_1 \in \mathscr{D}_{\{-1,1\}}^{2N_1} \\ \vdots \\ \Lambda_{m-1} \in \mathscr{D}_{\{-1,1\}}^{2N_{m-1}}}} \|\widetilde{W}_m\Lambda_{m-1}\widetilde{W}_{m-1} \cdots \Lambda_2\widetilde{W}_2\Lambda_1\widetilde{W}_1\| = \|\widetilde{W}_m\widetilde{W}_{m-1} \cdots \widetilde{W}_2\widetilde{W}_1\|. \tag{5.82}$$

According to Proposition 5.3.3(iii), $\widetilde{W}_m\widetilde{W}_{m-1} \cdots \widetilde{W}_2\widetilde{W}_1 \in \mathcal{A}_{N_m,N_0}$. Since this matrix has nonnegative elements, we deduce from Proposition 5.3.3(vii) that

$$\begin{aligned}\vartheta_m &\leqslant \|\widetilde{W}_{m+1}\|\,\|\widetilde{W}_m \cdots \widetilde{W}_1\|\,\|\widetilde{W}_0\| \\ &= \|\widetilde{W}_{m+1}\|\,\|\mathfrak{S}(\widetilde{W}_m \cdots \widetilde{W}_1)\|\,\|\widetilde{W}_0\| \\ &= \|\widetilde{W}_{m+1}\|\,\|\mathfrak{S}(\widetilde{W}_m) \cdots \mathfrak{S}(\widetilde{W}_1)\|\,\|\widetilde{W}_0\|,\end{aligned} \tag{5.83}$$

where the last equality is also a consequence of Proposition 5.3.3(iii). This leads to the Lipschitz bound in (5.19).

5.8.5 . Link between Conv layers and MIMO systems

To be rigorous, let us first define the space $\mathcal{H}_{i-1}$ (resp. $\mathcal{H}_i$) in which signals $(x_p)_{1\leqslant p\leqslant \zeta_{i-1}}$ (resp. $(y_q)_{1\leqslant q\leqslant \zeta_i}$) used in (5.26) live. Typically, $\mathcal{H}_i$ is some finite-dimensional subspace of $(\ell^2(\mathbb{Z}^d))^{\zeta_i}$ where $\ell^2(\mathbb{Z}^d)$ denotes the space of square summable discrete d-dimensional fields. For the discrete convolution $*$ to be properly defined, kernels $(w_{i,q,p})_{1\leqslant p\leqslant \zeta_{i-1},1\leqslant q\leqslant \zeta_i}$ are then assumed to be summable. In practice, this assumption is satisfied since these kernels are chosen with finite size.

For $x = \big(x(\mathbf{n})\big)_{\mathbf{n}\in\mathbb{Z}^d} \in \ell^2(\mathbb{Z}^d)$, the decimation operation $(\cdot)\downarrow_{s_i}$ returns the output signal

$$(\forall \mathbf{n} \in \mathbb{Z}^d) \quad y(\mathbf{n}) = u(s_i\mathbf{n}). \tag{5.84}$$

Eq. (5.26) defines a MIMO (multi-input multi-output) filter that can be reexpressed in a matrix form as

$$(\forall \mathbf{n} \in \mathbb{Z}^d) \quad \mathbf{u}(\mathbf{n}) = \sum_{\mathbf{n} \in \mathbb{Z}^d} \boldsymbol{W}_i(\mathbf{n}') \, \mathbf{x}(\mathbf{n} - \mathbf{n}') \\ = (\boldsymbol{W}_i * \mathbf{x})(\mathbf{n}), \tag{5.85}$$

where

$$\mathbf{u}(\mathbf{n}) = \begin{bmatrix} u_1(\mathbf{n}) \\ \vdots \\ u_{\zeta_i}(\mathbf{n}) \end{bmatrix} \in \mathbb{R}^{\zeta_j}, \quad \mathbf{x}(\mathbf{n}) = \begin{bmatrix} x_1(\mathbf{n}) \\ \vdots \\ x_{\zeta_{i-1}}(\mathbf{n}) \end{bmatrix} \in \mathbb{R}^{\zeta_{i-1}}, \tag{5.86}$$

and $\boldsymbol{W}_i(\mathbf{n})$ is given by (5.29). $(\boldsymbol{W}_i(\mathbf{n}))_{\mathbf{n} \in \mathbb{Z}^d}$ defines the so-called MIMO impulse response of $\mathcal{W}_i$. The MIMO impulse response of an ABBA layer is similarly given by (5.30).

These relations can also be written more concisely in the d-dimensional frequency domain[5] as

$$(\forall \boldsymbol{\nu} \in [0,1]^d) \quad \widehat{\mathbf{u}}(\boldsymbol{\nu}) = \widehat{\boldsymbol{W}}_i(\boldsymbol{\nu}) \, \widehat{\mathbf{x}}(\boldsymbol{\nu}), \tag{5.87}$$

where

$$\widehat{\mathbf{x}}(\boldsymbol{\nu}) = \sum_{\mathbf{n} \in \mathbb{Z}^d} \mathbf{x}(\mathbf{n}) \exp\big(-\imath 2\pi \mathbf{n}^\top \boldsymbol{\nu}\big) \in \mathbb{C}^{\zeta_{j-1}}, \tag{5.88}$$

$$\widehat{\boldsymbol{W}}_i(\boldsymbol{\nu}) = \sum_{\mathbf{n} \in \mathbb{Z}^d} \boldsymbol{W}_i(\mathbf{n}) \exp\big(-\imath 2\pi \mathbf{n}^\top \boldsymbol{\nu}\big) \in \mathbb{C}^{\zeta_{j-1} \times \zeta_j}, \tag{5.89}$$

and $\widehat{\boldsymbol{W}}_i$ is the frequency response of the associated MIMO filter.

Note that $\int_{[0,1]^d} \|\widehat{\mathbf{x}}(\boldsymbol{\nu})\|^2 d\boldsymbol{\nu} < +\infty$, whereas $\widehat{\boldsymbol{W}}_i$ is a continuous (hence bounded) function on $[0,1]^d$. Another useful result from sampling theory [184] is that the Fourier transform of $\mathbf{y} = (y_q)_{1 \leqslant q \leqslant \zeta_j}$ in (5.26) is deduced from the Fourier transform of $\mathbf{u}$ by the relation

$$(\forall \boldsymbol{\nu} \in [0,1]^d) \quad \widehat{\mathbf{y}}(\boldsymbol{\nu}) = \frac{1}{s_i^d} \sum_{\mathbf{j} \in \mathbb{S}(s_i)} \widehat{\mathbf{u}}\left(\frac{\boldsymbol{\nu} + \mathbf{j}}{s_i}\right). \tag{5.90}$$

where

$$(\forall \sigma \in \mathbb{N} \setminus \{0\}) \quad \mathbb{S}(\sigma) = \{0, \ldots, \sigma - 1\}^d. \tag{5.91}$$

It is also worth noting that the interpolation by a factor s of $\mathbf{y}$

$$\mathbf{v} = \mathbf{y}_{\uparrow_s} \quad \Leftrightarrow \quad (\forall \mathbf{n} \in \mathbb{Z}^d) \quad \mathbf{v}(\mathbf{n}) = \begin{cases} \mathbf{y}\left(\dfrac{\mathbf{n}}{s}\right) & \text{if } \mathbf{n} \in s\,\mathbb{Z}^d \\ 0 & \text{otherwise,} \end{cases} \tag{5.92}$$

translates into

$$(\forall \nu \in [0,1]) \quad \widehat{\mathbf{y}}_{\uparrow_s}(\boldsymbol{\nu}) = \widehat{\mathbf{y}}(s\boldsymbol{\nu}), \tag{5.93}$$

in the frequency domain.

[5] Alternatively, we could use the d-dimensional z-transform since we are dealing with discrete-space signals.

5.8.6 . Frequency expressions of Lipschitz bounds

In this appendix, we establish frequency-based bounds of the Lipschitz constant of an m-layer convolutional neural network T.

Based on the MIMO concepts introduced in Appendix 5.8.5, we will introduce the following global frequency response of the network:

$$(\forall \boldsymbol{\nu} \in [0,1]^d) \quad \widehat{\boldsymbol{W}}(\boldsymbol{\nu}) = \widehat{\boldsymbol{W}}_m(\sigma_{m-1}\boldsymbol{\nu}) \cdots \widehat{\boldsymbol{W}}_2(\sigma_1\boldsymbol{\nu})\widehat{\boldsymbol{W}}_1(\boldsymbol{\nu}) \in \mathbb{C}^{\zeta_m \times \zeta_0}, \quad (5.94)$$

where $\widehat{\boldsymbol{W}}_i$ is the frequency response associated to filter $\boldsymbol{W}_i$ (see (5.89)).

We have then the following result providing a frequency formula for evaluating the Lipschitz constant of a convolutional network.

Proposition 5.8.2 *The quantity*

$$\theta_m = \frac{1}{\sigma_m^{d/2}} \sup_{\boldsymbol{\nu} \in [0,1/\sigma_m]^d} \left\| \sum_{\mathbf{j} \in \mathbb{S}(\sigma_m)} \widehat{\boldsymbol{W}}\left(\boldsymbol{\nu} + \frac{\mathbf{j}}{\sigma_m}\right) \widehat{\boldsymbol{W}}\left(\boldsymbol{\nu} + \frac{\mathbf{j}}{\sigma_m}\right)^{\mathsf{H}} \right\|^{1/2}. \quad (5.95)$$

provides a lower bound on the Lipschitz constant estimate of network T [6]. In addition, if for every $i \in \{1,\dots,m\}$, $p \in \{1,\dots,\zeta_{i-1}\}$, and $q \in \{1,\dots,\zeta_i\}$, $w_{i,q,p} = (w_{i,q,p}(\mathbf{n}))_{\mathbf{n}\in\mathbb{Z}^d}$ is a nonnegative kernel i.e.,

$$(\forall \mathbf{n} \in \mathbb{Z}^d) \quad w_{i,p,q}(\mathbf{n}) \geqslant 0, \quad (5.96)$$

then θ_m is a Lipschitz constant of T.

Proof. In the considered case all activation operators are nonexpansive and they are assumed separable, except maybe at the last layer. Thus T is a special case of the networks investigated in [20, Section 5] for which a tight estimate of the Lipschitz constant was provided. It then follows from [20, Theorem 5.2] that a lower bound on this Lipschitz constant estimate is

$$\theta_m = \|\mathcal{W}_m \circ \cdots \circ \mathcal{W}_1\|. \quad (5.97)$$

In addition, under the additional assumption that all the kernels are nonnegative, T is an instance of the positively weighted networks investigated in [20, Section 5.3] and it follows from [20, Proposition 5.10] that θ_m is then a Lipschitz constant of T.

So the problem is to calculate the norm of the linear operator $\mathcal{W} = \mathcal{W}_m \circ \cdots \circ \mathcal{W}_1$. Each operator $\mathcal{W}_i$ with $i \in \{1,\dots,m\}$ is the composition of a d-dimensional MIMO filter with a decimator. It follows from Noble identities [184] that $\mathcal{W}$ reduces to cascading a $\zeta_m \times \zeta_0$ MIMO filter with frequency response $\widetilde{\boldsymbol{W}}$ with a decimation of

[6] $(\cdot)^{\mathsf{H}}$ denotes the Hermitian transpose operation.

each output by a factor σ_m. More precisely, if $\mathbf{x} \in \mathcal{H}_0$ is the input of this linear system and $\mathbf{y}$ its output, we have in the frequency domain:

$$(\forall \boldsymbol{\nu} \in [0,1]^d) \quad \widehat{\mathbf{y}}(\boldsymbol{\nu}) = \frac{1}{\sigma_m^d} \sum_{\mathbf{j} \in \mathbb{S}(\sigma_m)} \widehat{\boldsymbol{W}}\left(\frac{\boldsymbol{\nu}+\mathbf{j}}{\sigma_m}\right) \widehat{\mathbf{x}}\left(\frac{\boldsymbol{\nu}+\mathbf{j}}{\sigma_m}\right)$$
$$= \frac{1}{\sigma_m^d} \widetilde{\boldsymbol{W}}\left(\frac{\boldsymbol{\nu}}{\sigma_m}\right) \widetilde{\mathbf{x}}\left(\frac{\boldsymbol{\nu}}{\sigma_m}\right), \tag{5.98}$$

where $\widetilde{\mathbf{x}}\left(\frac{\boldsymbol{\nu}}{\sigma_m}\right)$ is a vector of dimension $\mathbb{C}^{\sigma_m^d \zeta_0}$ where the vectors $\left(\widehat{\mathbf{x}}((\boldsymbol{\nu}+\mathbf{j})/\sigma_m)\right)_{\mathbf{j} \in \mathbb{S}(\sigma_m)}$ are stacked columnwise and $\widetilde{\boldsymbol{W}}\left(\frac{\boldsymbol{\nu}}{\sigma_m}\right)$ is a $c_m \times \sigma_m^d \zeta_0$ matrix where the matrices $\left(\widehat{\boldsymbol{W}}((\boldsymbol{\nu}+\mathbf{j})/\sigma_m)\right)_{\mathbf{j} \in \mathbb{S}(\sigma_m)}$ are stacked rowwise. For example, when $d = 2$, we have, for every $\boldsymbol{\nu} = (\nu_1, \nu_2) \in [0,1]^2$,

$$\widetilde{\mathbf{x}}(\boldsymbol{\nu}) = \begin{bmatrix} \check{\mathbf{x}}(\nu_1, \nu_2) \\ \check{\mathbf{x}}\left(\nu_1, \nu_2 + \frac{1}{\sigma_m}\right) \\ \vdots \\ \check{\mathbf{x}}\left(\nu_1, \nu_2 + \frac{\sigma_m - 1}{\sigma_m}\right) \end{bmatrix} \in \mathbb{C}^{\sigma_m^2 \zeta_0} \tag{5.99}$$

$$\check{\mathbf{x}}(\boldsymbol{\nu}) = \begin{bmatrix} \widehat{\mathbf{x}}(\nu_1, \nu_2) \\ \widehat{\mathbf{x}}\left(\nu_1 + \frac{1}{\sigma_m}, \nu_2\right) \\ \vdots \\ \widehat{\mathbf{x}}\left(\nu_1 + \frac{\sigma_m - 1}{\sigma_m}, \nu_2\right) \end{bmatrix} \in \mathbb{C}^{\sigma_m \zeta_0} \tag{5.100}$$

$$\widetilde{\boldsymbol{W}}(\boldsymbol{\nu}) = \left[\check{\boldsymbol{W}}(\nu_1, \nu_2) \;\; \check{\boldsymbol{W}}\left(\nu_1, \nu_2 + \frac{1}{\sigma_m}\right) \;\; \ldots \;\; \check{\boldsymbol{W}}\left(\nu_1, \nu_2 + \frac{\sigma_m - 1}{\sigma_m}\right)\right] \in \mathbb{C}^{\zeta_m \times \sigma_m^2 \zeta_0} \tag{5.101}$$

$$\check{\boldsymbol{W}}(\boldsymbol{\nu}) = \left[\widehat{\boldsymbol{W}}(\nu_1, \nu_2) \;\; \widehat{\boldsymbol{W}}\left(\nu_1 + \frac{1}{\sigma_m}, \nu_2\right) \;\; \ldots \;\; \widehat{\boldsymbol{W}}\left(\nu_1 + \frac{\sigma_m - 1}{\sigma_m}, \nu_2\right)\right] \in \mathbb{C}^{\zeta_m \times \sigma_m \zeta_0}. \tag{5.102}$$

By using now Parseval's formula,

$$\begin{aligned}
\|\mathbf{y}\|^2 &= \int_{[0,1]^d} \|\widehat{\mathbf{y}}(\boldsymbol{\nu})\|^2 d\boldsymbol{\nu} \\
&= \frac{1}{\sigma_m^{2d}} \int_{[0,1]^d} \left\| \widetilde{\boldsymbol{W}}\left(\frac{\boldsymbol{\nu}}{\sigma_m}\right) \widetilde{\mathbf{x}}\left(\frac{\boldsymbol{\nu}}{\sigma_m}\right) \right\|^2 d\boldsymbol{\nu} \\
&\leqslant \frac{1}{\sigma_m^{d}} \int_{[0,1/\sigma_m]^d} \left\| \widetilde{\boldsymbol{W}}(\boldsymbol{\nu}) \right\|^2 \|\widetilde{\mathbf{x}}(\boldsymbol{\nu})\|^2 d\nu_1 d\nu_2 \\
&\leqslant \frac{1}{\sigma_m^{d}} \sup_{\boldsymbol{\nu}\in[0,1/\sigma_m]^d} \left\| \widetilde{\boldsymbol{W}}(\boldsymbol{\nu}) \right\|^2 \int_{[0,1/\sigma_m]^d} \|\widetilde{\mathbf{x}}(\boldsymbol{\nu})\|^2 d\boldsymbol{\nu} \\
&= \frac{1}{\sigma_m^{2}} \sup_{\boldsymbol{\nu}\in[0,1/\sigma_m]^d} \left\| \widetilde{\boldsymbol{W}}(\boldsymbol{\nu}) \right\|^2 \sum_{\mathbf{j}\in\mathbb{S}(\sigma_m)} \int_{[0,1/\sigma_m]^d} \left\| \widehat{\mathbf{x}}\left(\boldsymbol{\nu}+\frac{\mathbf{j}}{\sigma_m}\right) \right\|^2 d\boldsymbol{\nu} \\
&= \frac{1}{\sigma_m^{d}} \sup_{\boldsymbol{\nu}\in[0,1/\sigma_m]^d} \left\| \widetilde{\boldsymbol{W}}(\boldsymbol{\nu}) \right\|^2 \int_{[0,1]^d} \|\widehat{\mathbf{x}}(\boldsymbol{\nu})\|^2 d\boldsymbol{\nu} \\
&= \frac{1}{\sigma_m^{d}} \sup_{\boldsymbol{\nu}\in[0,1/\sigma_m]^d} \left\| \widetilde{\boldsymbol{W}}(\boldsymbol{\nu}) \right\|^2 \|\mathbf{x}\|^2. \qquad (5.103)
\end{aligned}$$

This shows that

$$\theta_m^2 \leqslant \frac{1}{\sigma_m^{d}} \sup_{\boldsymbol{\nu}\in[0,1/\sigma_m]^d} \left\| \widetilde{\boldsymbol{W}}(\boldsymbol{\nu}) \right\|^2. \qquad (5.104)$$

On the other hand since $\widehat{\boldsymbol{W}}$ is continuous, $\widetilde{\boldsymbol{W}}$ is also continuous, and there exists $\widehat{\boldsymbol{\nu}} \in [0,1/\sigma_m]^d$ such that

$$\sup_{\boldsymbol{\nu}\in[0,1/\sigma_m]^d} \left\| \widetilde{\boldsymbol{W}}(\boldsymbol{\nu}) \right\| = \left\| \widetilde{\boldsymbol{W}}(\widehat{\boldsymbol{\nu}}) \right\|. \qquad (5.105)$$

Let us now choose, for every $\boldsymbol{\nu} \in [0,1/\sigma_m]^d$, $\widetilde{\mathbf{x}}(\boldsymbol{\nu}) = \alpha_\epsilon(\boldsymbol{\nu})\mathbf{u}(\boldsymbol{\nu})$ where $\mathbf{u}(\boldsymbol{\nu})$ is a unit norm eigenvector associated with the maximum eigenvalue of $\widetilde{\boldsymbol{W}}(\boldsymbol{\nu})^{\mathsf{H}}\widetilde{\boldsymbol{W}}(\boldsymbol{\nu})$, $\epsilon \in]0,+\infty[$, and

$$\alpha_\epsilon(\boldsymbol{\nu}) = \begin{cases} \frac{1}{\epsilon^{d/2}} & \text{if } (\exists \mathbf{j} \in \{-1,0,1\}^d)\ \|\boldsymbol{\nu} + \frac{\mathbf{j}}{\sigma_m} - \widehat{\boldsymbol{\nu}}\|_\infty \leqslant \frac{\epsilon}{2} \\ 0 & \text{otherwise.} \end{cases} \qquad (5.106)$$

Then we see that when $\epsilon \to 0$, the upper bound in (5.103) is reached. We conclude that

$$\theta_m = \frac{1}{\sigma_m^{d/2}} \sup_{\boldsymbol{\nu}\in[0,1/\sigma_m]^d} \left\| \widetilde{\boldsymbol{W}}(\boldsymbol{\nu}) \right\|. \qquad (5.107)$$

In addition, by using the relation between $\widetilde{\boldsymbol{W}}$ and $\widehat{\boldsymbol{W}}$ (i.e., (5.101) and (5.102) in the 2D case),

$$\begin{aligned}
\left\| \widetilde{\boldsymbol{W}}(\boldsymbol{\nu}) \right\|^2 &= \left\| \widetilde{\boldsymbol{W}}(\boldsymbol{\nu}) \widetilde{\boldsymbol{W}}(\boldsymbol{\nu})^{\mathsf{H}} \right\| \\
&= \left\| \sum_{\mathbf{j}\in\mathbb{S}(\sigma_m)} \widehat{\boldsymbol{W}}\left(\boldsymbol{\nu}+\frac{\mathbf{j}}{\sigma_m}\right) \widehat{\boldsymbol{W}}\left(\boldsymbol{\nu}+\frac{\mathbf{j}}{\sigma_m}\right)^{\mathsf{H}} \right\|. \qquad (5.108)
\end{aligned}$$

Gathering the last two equalities yields (5.95). □

When there is no decimation, i.e. the strides $(s_i)_{1\leqslant i\leqslant m}$ are all equal to 1, (5.95) reduces to

$$\theta_m = \sup_{\boldsymbol{\nu}\in[0,1]^d} \|\widehat{\boldsymbol{W}}_m(\boldsymbol{\nu})\cdots\widehat{\boldsymbol{W}}_2(\boldsymbol{\nu})\widehat{\boldsymbol{W}}_1(\boldsymbol{\nu})\|. \tag{5.109}$$

We recall that the following upper bound holds [9]:

$$\theta_m \leqslant \overline{\theta}_m = \prod_{i=1}^{m} \|\mathcal{W}_i\|. \tag{5.110}$$

Applying our result to the one-layer case shows that, for every $i \in \{1,\ldots,m\}$,

$$\|\mathcal{W}_i\| = \frac{1}{s_i^{d/2}} \sup_{\boldsymbol{\nu}\in[0,1/s_i]^2} \left\| \sum_{\mathbf{j}\in\mathbb{S}(s_i)} \widehat{\boldsymbol{W}}_i\left(\boldsymbol{\nu}+\frac{\mathbf{j}}{s_i}\right)\widehat{\boldsymbol{W}}_i\left(\boldsymbol{\nu}+\frac{\mathbf{j}}{s_i}\right)^{\mathsf{H}} \right\|^{1/2}. \tag{5.111}$$

Note that the resulting upper bound in (5.110) gives a loose estimate of the Lipschitz constant, which has however the merit to be valid for convolutional networks having kernels with an arbitrary sign.

5.8.7 . Proof of Theorem 5.4.1

Before giving the proof of our main result, we will introduce a link between the Fourier and the spatial representations for a nonnegative convolutional kernel.

Lemma 5.8.3 *Let $(c,c') \in (\mathbb{N}\setminus\{0\})^2$ and let*

$$(\forall \mathbf{n}\in\mathbb{Z}^d) \quad \boldsymbol{H}(\mathbf{n}) = \big(h_{q,p}(\mathbf{n})\big)_{1\leqslant q\leqslant c',1\leqslant p\leqslant c} \in [0,+\infty[^{c'\times c} \tag{5.112}$$

where, for every $p\in\{1,\ldots,c\}$ and $q\in\{1,\ldots,c'\}$, $h_{q,p}\in\ell^1(\mathbb{Z}^d)$ [7]. Then, the Fourier transform $\widehat{\boldsymbol{H}}$ of $\big(\boldsymbol{H}(\mathbf{n})\big)_{\mathbf{n}\in\mathbb{Z}^d}$ is such that

$$\sup_{\boldsymbol{\nu}\in[0,1]^d} \|\widehat{\boldsymbol{H}}(\boldsymbol{\nu})\| = \Big\| \sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{H}(\mathbf{n})\Big\|. \tag{5.113}$$

Proof. For every $\boldsymbol{\nu}\in[0,1]^d$,

$$\widehat{\boldsymbol{H}}(\boldsymbol{\nu}) = \sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{H}(\mathbf{n})\exp\big(-\imath 2\pi\pi\mathbf{n}^\top\boldsymbol{\nu}\big). \tag{5.114}$$

[7] $\ell^1(\mathbb{Z}^d)$ is the space of summable d-dimensional sequences

For every $\mathbf{u} = [u_1, \ldots, u_c]^\top \in \mathbb{C}^c$, by using the triangle inequality,

$$\begin{aligned}
\|\widehat{\boldsymbol{H}}(\boldsymbol{\nu})\mathbf{u}\|^2 &= \sum_{q=1}^{c'} \Big| \sum_{p=1}^{c} [\boldsymbol{H}(\boldsymbol{\nu})]_{q,p} u_p \Big|^2 \\
&= \sum_{q=1}^{c'} \Big| \sum_{p=1}^{c} \sum_{\mathbf{n}\in\mathbb{Z}^d} h_{q,p}(\mathbf{n}) \exp\big(-\imath 2\pi \mathbf{n}^\top \boldsymbol{\nu}\big) u_p \Big|^2 \\
&\leqslant \sum_{q=1}^{c'} \Big(\sum_{p=1}^{c} \sum_{\mathbf{n}\in\mathbb{Z}^d} h_{q,p}(\mathbf{n}) |u_p| \Big)^2 \\
&= \Big\| \sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{H}(\mathbf{n}) |\mathbf{u}| \Big\|^2 \\
&\leqslant \Big\| \sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{H}(\mathbf{n}) \Big\|^2 \, \big\| |\mathbf{u}| \big\|^2 \\
&= \Big\| \sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{H}(\mathbf{n}) \Big\|^2 \, \|\mathbf{u}\|^2,
\end{aligned} \tag{5.115}$$

where $|\mathbf{u}|$ denotes the vector of moduli of the components of vector $\mathbf{u}$. This shows that

$$\|\widehat{\boldsymbol{H}}(\boldsymbol{\nu})\| \leqslant \Big\| \sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{H}(\mathbf{n}) \Big\|. \tag{5.116}$$

and, consequently,

$$\sup_{\boldsymbol{\nu}\in[0,1]^d} \|\widehat{\boldsymbol{H}}(\boldsymbol{\nu})\| \leqslant \Big\| \sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{H}(\mathbf{n}) \Big\|. \tag{5.117}$$

In addition, the upper bound is attained since

$$\widehat{\boldsymbol{H}}(\mathbf{0}) = \sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{H}(\mathbf{n}). \tag{5.118}$$

□

Next, we derive the proof for Theorem 5.4.1 in light of Lemma 5.8.3.

Proof. $\boldsymbol{W}$ being the impulse response of the MIMO filter with the frequency response given by (5.94), it follows from Noble identities [180] that $\mathcal{W}$ is equivalent to a convolution with $\boldsymbol{W}$ followed by a decimation by a factor σ_m. Let $\mathbf{x} \in \mathcal{H}_0$. Let the σ_m-polyphase representation of $\mathbf{x}$ (resp. $\boldsymbol{W}$) be defined as

$$\big(\forall \mathbf{j} \in \mathbb{S}(\sigma_m)\big)\big(\forall \mathbf{n} \in \mathbb{Z}^d\big) \quad \mathbf{x}^{(\mathbf{j})}(\mathbf{n}) = \mathbf{x}(\sigma_m \mathbf{n} - \mathbf{j}) \tag{5.119}$$

$$\Big(\text{resp. } \boldsymbol{W}^{(\mathbf{j})}(\mathbf{n}) = \boldsymbol{W}(\sigma_m \mathbf{n} + \mathbf{j})\Big). \tag{5.120}$$

Then, as a result of multirate digital filtering, $\mathbf{y} = \mathcal{W}\mathbf{x}$ if and only if

$$\mathbf{y} = \sum_{\mathbf{j}\in\mathbb{S}(\sigma_m)} \boldsymbol{W}^{(\mathbf{j})} * \mathbf{x}^{(\mathbf{j})}. \tag{5.121}$$

This sum of MIMO convolutions can be reformulated as a single one

$$\mathbf{y} = \boldsymbol{H} * \mathbf{e}, \tag{5.122}$$

where $\boldsymbol{H}$ is the $c_m \times \sigma_m^d c_0$ MIMO impulse response obtained by stacking rowwise the polyphase MIMO impulse responses $(\boldsymbol{W}^{(\mathbf{j})})_{\mathbf{j}\in\mathbb{S}(\sigma_m)}$ and $\mathbf{e}$ is the $\sigma_m^d c_0$-component d-dimensional signal obtained by stacking columnwise the polyphase signal components $(\mathbf{x}^{(\mathbf{j})})_{\mathbf{j}\in\mathbb{S}(\sigma_m)}$. For example, if $d = 2$, we have

$$\begin{aligned}&(\forall \mathbf{n} \in \mathbb{Z}^2)\\ &\boldsymbol{H}(\mathbf{n}) = [\boldsymbol{W}^{(0,0)}(\mathbf{n}), \ldots, \boldsymbol{W}^{(\sigma_m-1,0)}(\mathbf{n}), \ldots, \boldsymbol{W}^{(0,\sigma_m-1)}(\mathbf{n}), \ldots, \boldsymbol{W}^{(\sigma_m-1,\sigma_m-1)}(\mathbf{n})]\end{aligned} \tag{5.123}$$

and

$$(\forall \mathbf{n} \in \mathbb{Z}^2) \quad \mathbf{e}(\mathbf{n}) = \begin{bmatrix} \mathbf{x}^{(0,\sigma_m-1)}(\mathbf{n}) \\ \vdots \\ \mathbf{x}^{(\sigma_m-1,\sigma_m-1)}(\mathbf{n}) \\ \vdots \\ \mathbf{x}^{(\sigma_m-1,\sigma_m-1)}(\mathbf{n}) \\ \vdots \\ \mathbf{x}^{(\sigma_m-1,\sigma_m-1)}(\mathbf{n}) \end{bmatrix}. \tag{5.124}$$

Note that, according to (5.119),

$$\begin{aligned}\|\mathbf{e}\|^2 &= \sum_{\mathbf{n}\in\mathbb{Z}^d} \|\mathbf{e}(\mathbf{n})\|^2 \\ &= \sum_{\mathbf{n}\in\mathbb{Z}^d} \sum_{\mathbf{j}\in\mathbb{S}(\sigma_m)} \|\mathbf{x}^{(\mathbf{j})}(\mathbf{n})\|^2 \\ &= \sum_{\mathbf{n}\in\mathbb{Z}^d} \|\mathbf{x}(\mathbf{n})\|^2 \\ &= \|\mathbf{x}\|^2.\end{aligned} \tag{5.125}$$

This equality and (5.122) imply that

$$\|\mathcal{W}\| = \sup_{\boldsymbol{\nu}\in[0,1]^d} \|\widehat{\boldsymbol{H}}(\boldsymbol{\nu})\|. \tag{5.126}$$

We thus deduce from Lemma 5.8.3 that

$$\|\mathcal{W}\| = \Big\| \sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{H}(\mathbf{n}) \Big\|. \tag{5.127}$$

On the other hand, by using (5.120),

$$\begin{aligned}
&\Big\| \sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{H}(\mathbf{n}) \Big\| \\
&= \left\| \Big(\sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{H}(\mathbf{n}) \Big) \Big(\sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{H}(\mathbf{n}) \Big)^\top \right\|^{1/2} \\
&= \left\| \sum_{\mathbf{j}\in\mathbb{S}(\sigma_m)} \Big(\sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{W}^{(\mathbf{j})}(\mathbf{n}) \Big) \Big(\sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{W}^{(\mathbf{j})}(\mathbf{n}) \Big)^\top \right\|^{1/2} \\
&= \Big\| \sum_{\mathbf{j}\in\mathbb{S}(\sigma_m)} \overline{\boldsymbol{W}}^{(\mathbf{j})} \big(\overline{\boldsymbol{W}}^{(\mathbf{j})}\big)^\top \Big\|^{1/2}. \qquad (5.128)
\end{aligned}$$

□

5.8.8 . Numerical evaluation of the Lipschitz constant of nonnegative convolutional networks

We compare the tight bound θ_m in Theorem 5.4.1 with the separable one $\overline{\theta}_m$ given by (5.37) for a classic convolutional network using non-negative kernels. The results provided in Table 5.2 correspond to the convolutive part of LeNet-5 [185]. In our experiments, we initialized the networks with randomly sampled weights drawn from a uniform distribution on $[0, 1]$. Table 5.2 shows the relative difference

$$\epsilon_{\mathrm{r}} = \frac{\overline{\theta}_m - \theta_m}{\overline{\theta}_m},$$

for 10 distinct noise realizations. We thus observe that the difference between the two bounds is small. Similar observations can be made on various convolutive architectures. In contrast, for fully connected networks, a separable bound is usually overpessimistic.

	LeNet-5									
	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
θ_m	30302.73	27734.91	30298.73	29374.35	30180.16	28632.60	30615.02	30395.67	34828.90	30097.62
$\overline{\theta}_m$	30696.07	28114.29	30860.56	29821.62	30670.05	29298.64	31152.06	30866.87	35220.36	30367.71
$\epsilon_{\mathrm{r}}\,[\%]$	1.28	1.35	1.82	1.50	1.60	2.27	1.72	1.53	1.11	0.89

Table 5.2: Lipschitz bounds obtained for 10 independent realizations of random positive initialization for LeNet-5.

5.8.9 . Lipschitz constant of average pooling

We consider the case when the i-th layer is an *average pooling* where the average is computed on patches of length L_i in each dimension and with stride s_i. For simplicity, we suppose that L_i is a multiple of s_i. The number of input and output channels is then equal, i.e. $\zeta_i = \zeta_{i-1}$. The average is calculated

on each channel independently, this operation is a special case of a nonnegative convolutional layer where, for every $\mathbf{n} \in \mathbb{Z}^d$, $\boldsymbol{W}_i(\mathbf{n})$ is a diagonal matrix. The diagonal elements of this matrix are

$$(\forall p \in \{1, \dots, \zeta_i\})(\forall \mathbf{n} \in \mathbb{Z}^d) \quad w_{i,p,p}(\mathbf{n}) = \begin{cases} \frac{1}{L_i^d} & \text{if } \mathbf{n} \in [0, L_i - 1]^d \\ 0 & \text{otherwise.} \end{cases} \tag{5.129}$$

We deduce that, for every $\mathbf{j} \in \mathbb{S}(s_i)$, the matrix $\overline{\boldsymbol{W}}_i^{(\mathbf{j})}$ is also a diagonal matrix. More precisely, the sum in (5.38) can be restricted to values of $\mathbf{n} \in \{0, \dots, L_i/s_i - 1\}^d$ and $\overline{\boldsymbol{W}}_i^{(\mathbf{j})} = \frac{1}{s_i^d}\mathbf{Id}$. We deduce that the Lipschitz constant of the average pooling layer is

$$\|\mathcal{W}_i\| = \Big\| \sum_{\mathbf{j} \in \mathbb{S}(s_i)} \overline{\boldsymbol{W}}_i^{(\mathbf{j})} \big(\overline{\boldsymbol{W}}_i^{(\mathbf{j})}\big)^\top \Big\|^{1/2} = \frac{1}{s_i^{d/2}}. \tag{5.130}$$

We see that this constant is independent of the patch size and is a decreasing function of the stride.

5.8.10 . Proof of Theorem 5.4.2

This result is a consequence of Theorem 5.4.1, which provides a Lipschitz bound for nonnegative convolutional neural networks. By following a similar reasoning to Appendix 5.8.4, a Lipschitz constant of the ABBA network is

$$\theta_m = \|\widetilde{W}_{m+1}\| \|\widetilde{W}_m \circ \ldots \circ \widetilde{W}_1\| \|\|\widetilde{W}_0\|. \tag{5.131}$$

Let

$$\widetilde{\boldsymbol{W}} = (\widetilde{\boldsymbol{W}}_m)_{\uparrow \sigma_{m-1}} * \cdots * (\widetilde{\boldsymbol{W}}_2)_{\uparrow \sigma_1} * \widetilde{\boldsymbol{W}}_1 \tag{5.132}$$

and let

$$(\forall \mathbf{j} \in \mathbb{S}(\sigma_m)) \quad \Omega^{(\mathbf{j})} = \sum_{\mathbf{n} \in \mathbb{Z}^d} \mathfrak{S}\big(\widetilde{\boldsymbol{W}}(\sigma_m \mathbf{n} + \mathbf{j})\big) \in [0, +\infty[^{\zeta_m \times \zeta_0}. \tag{5.133}$$

Since $(\widetilde{W}_i)_{1 \leqslant i \leqslant m}$ are convolutional operators with nonnegative kernels, it follows from Theorem 5.4.1 that

$$\theta_m = \|\widetilde{W}_{m+1}\| \Big\| \sum_{\mathbf{j} \in \mathbb{S}(\sigma_m)} \overline{\boldsymbol{W}}^{(\mathbf{j})} \big(\overline{\boldsymbol{W}}^{(\mathbf{j})}\big)^\top \Big\|^{1/2} \|\widetilde{W}_0\|. \tag{5.134}$$

where, for every $\mathbf{j} \in \mathbb{S}(\sigma_m)$,

$$\overline{\boldsymbol{W}}^{(\mathbf{j})} = \sum_{\mathbf{n} \in \mathbb{Z}^d} \widetilde{\boldsymbol{W}}(\sigma_m \mathbf{n} + \mathbf{j}) \in [0, +\infty[^{(2\zeta_m) \times (2\zeta_0)}. \tag{5.135}$$

On the other hand, for every $\mathbf{n} \in \mathbb{Z}^d$, $\widetilde{\boldsymbol{W}}_i(\mathbf{n})$ is an ABBA matrix. Since $\widetilde{\boldsymbol{W}}(\mathbf{n})$ is obtained by multiplication and addition of such matrices, it follows from Proposition 5.3.3(ii) and (iii) that it is also an ABBA matrix. We deduce that, for every $\mathbf{j} \in$

$\mathbb{S}(\sigma_m)$, $\overline{\boldsymbol{W}}^{(\mathbf{j})}$ is an ABBA matrix and, by using Proposition 5.3.3(i), $\overline{\boldsymbol{W}}^{(\mathbf{j})}\big(\overline{\boldsymbol{W}}^{(\mathbf{j})}\big)^\top$ is ABBA. By invoking now Proposition 5.3.3(i)-(iii) and (vii), we deduce that

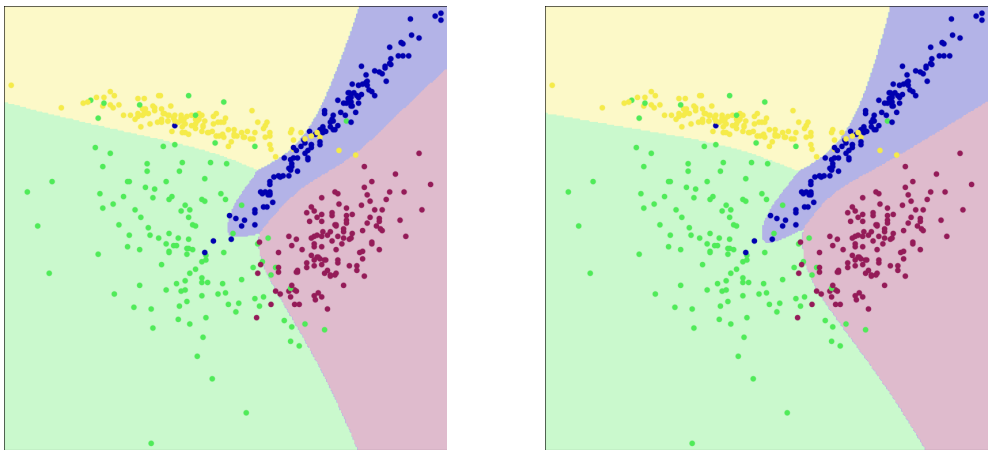
$$\begin{aligned}\Big\| \sum_{\mathbf{j}\in\mathbb{S}(\sigma_m)} \overline{\boldsymbol{W}}^{(\mathbf{j})}\big(\overline{\boldsymbol{W}}^{(\mathbf{j})}\big)^\top \Big\| &= \Big\| \mathfrak{S}\Big(\sum_{\mathbf{j}\in\mathbb{S}(\sigma_m)} \overline{\boldsymbol{W}}^{(\mathbf{j})}\big(\overline{\boldsymbol{W}}^{(\mathbf{j})}\big)^\top \Big) \Big\| \\ &= \Big\| \sum_{\mathbf{j}\in\mathbb{S}(\sigma_m)} \mathfrak{S}\Big(\overline{\boldsymbol{W}}^{(\mathbf{j})}\Big)\mathfrak{S}\Big(\overline{\boldsymbol{W}}^{(\mathbf{j})}\Big)^\top \Big\| \\ &= \Big\| \sum_{\mathbf{j}\in\mathbb{S}(\sigma_m)} \Omega^{(\mathbf{j})}\big(\Omega^{(\mathbf{j})}\big)^\top \Big\|. \end{aligned} \tag{5.136}$$

This shows that a Lipschitz constant of the ABBA network $\widetilde{T}$ is

$$\theta_m = \|\widetilde{W}_{m+1}\| \Big\| \sum_{\mathbf{j}\in\mathbb{S}(\sigma_m)} \Omega^{(\mathbf{j})}\big(\Omega^{(\mathbf{j})}\big)^\top \Big\|^{1/2} \|\widetilde{W}_0\|, \tag{5.137}$$

Similarly to the derivation of (5.37), we deduce that $\theta_m \leqslant \overline{\theta}_m$ where $\overline{\theta}_m$ is given by (5.40).

5.8.11 . Expressivity of ABBA networks – simulations



(a) Standard model (b) ABBA model

Figure 5.12: Decision space comparison between fitting an ABBA network and a standard arbitrary-signed one.

For this experiment, we randomly sampled points from four distinct 2D Gaussian distributions, with different means and covariance matrices, totalling 125 2-dimensional points per class. Figure 5.12 shows a comparison between decision boundaries resulting from training two models: a standard one trained conventionally and its non-negative ABBA equivalent. The two models reach a similar solution, showing that the theoretical properties proved in this paper are also observed in practical simulations.

5.8.12 . Constrained training of signed convolutional layers

The first and the last layers of an ABBA convolutional network have signed kernels. The norm of these layers is computed by using (5.111) and constrained to be less than $\overline{\theta}_{m,i,t}$ with $i \in \{0, m+1\}$. Note that (5.111) makes use of the frequency response $\widehat{\boldsymbol{W}}_i$ of filter $\mathcal{W}_i$. A discrete Fourier transform (DFT) is actually implemented (using 128×128 discrete frequencies). In the discrete frequency domain, the upper bound constraint is thus decomposed into 128^2 matrix norm bounds obtained by summing over s_i^2 frequencies. The projection onto each of these elementary constraint sets is computed by truncating a singular value decomposition. An additional constraint, however, is to be addressed, which is related to the fact that the kernels are of finite size. This implicitly defines a linear constraint. Projecting onto the associated vector space is simply obtained by truncating the kernel (after inverse DFT) to the desired size. The set $\mathcal{S}_{i,t}$ is thus defined as the intersection of the former matrix norm constraint set and the latter vector space. Projecting onto this intersection can be achieved by an iterative convex optimization approach. In our case, we use a Douglas-Rachford algorithm [148].

5.8.13 . ABBA architectures

Layer type	RPS	stride	CelebA	stride
Input	$150 \times 150 \times 3$		$128 \times 128 \times 3$	
Conv2D	$150 \times 150 \times 8$	1	$128 \times 128 \times 8$	1
ABBA Conv2D + CLR	–	–	$64 \times 64 \times 8(\times 2)$	2
ABBA Conv2D + CLR	$75 \times 75 \times 32(\times 2)$	2	$32 \times 32 \times 16(\times 2)$	2
ABBA Conv2D + CLR	$37 \times 37 \times 64(\times 2)$	2	$16 \times 16 \times 32(\times 2)$	2
ABBA Conv2D + CLR	$18 \times 18 \times 128(\times 2)$	2	$8 \times 8 \times 64(\times 2)$	2
Conv2D	$18 \times 18 \times 128$	1	$8 \times 8 \times 64$	1
Global Max-Pooling2D	$128(\times 2)$		$64(\times 2)$	
ABBA Dense + CLR	$128(\times 2)$		–	
ABBA Dense + CLR	$64(\times 2)$		–	
ABBA Dense + CLR	$32(\times 2)$		–	
Dense	3		2	

Table 5.3: ABBA Conv architectures details for RPS and CelebA datasets.

Table 5.4 details the ABBA Dense and ABBA Conv architectures used for MNIST and FMNIST datasets, while Table 5.3 shows our choices for RPS and CelebA datasets. As the ABBA layers have a specific form, their output size will be twice the number of filters. The used activation operator is the Capped Leaky ReLu (CLR) function defined in (5.52) for all Dense layers. For convolutional operators we employed a 3×3 kernel, using the same activation.

We used the official train-test split provided by the Tensorflow framework for MNIST/FMNIST datasets and did not employ any augmentation strategy during training. For RPS and CelebA models, we resized the input images to 150×150,

Layer type	MNIST/FMNIST
Input	$28 \times 28 \times 1$
Conv2D	$28 \times 28 \times 32$
ABBA Conv2D + CLR	$28 \times 28 \times 16$
ABBA Conv2D + CLR	$28 \times 28 \times 16$
Conv2D	$28 \times 28 \times 1$
Dense	256
ABBA Dense + CLR	128
ABBA Dense + CLR	64
Dense	10

Layer type	MNIST	FMNIST
Input	784	784
Dense	256	256
ABBA Dense + CLR	128	128
ABBA Dense + CLR	64	64
ABBA Dense + CLR	–	32
Dense	10	10

Table 5.4: ABBA Dense and ABBA Conv architecture details for MNIST and FMNIST datasets. For convolutional layers, the stride is set to 1.

Dataset	Optimizer	No. Epochs	Learning rate	Batch size
MNIST	projected ADAM	150	10^{-3}	1024
FMNIST	projected ADAM	200	10^{-3}	1024
RPS	projected ADAM	250	10^{-4}	64
CelebA	projected ADAM	100	10^{-4}	128

Table 5.5: Training hyperparameters.

resp. 128×128, before feeding them to the network. In the case of CelebA dataset, we opted for a binary classification task on the *bald* feature. We extracted all the images containing the *bald* attribute, and we randomly selected the same number of examples from the *non-bald* class, in order to avoid class imbalance. Additional information regarding the optimization parameters used during training is provided in Table 5.5.

Chapter 6 – Conclusions

6.1 . Summary

Despite the fact that they may appear at the forefront of developments in Data Science, neural networks raise challenges in the areas of safety, privacy, and security due to their susceptibility to a wide variety of threats and perturbations that may arise while they are in operation. It is therefore vital to understand the reasons for neural network instability, identify the areas of concern, and develop solutions that aim to improve their stability in order to guarantee the existence of AI-based systems that are agnostic to small variations of their inputs.

During this thesis, our main focus was the design and training of neural networks that are intrinsically robust against adversarial perturbations of their inputs. Thus, we proposed several robust training techniques, and we proved their effectiveness in solving both classification and regression problems. We showed that our research is applicable to a wide range of applications and that its results may be useful in real-life scenarios as well.

First, we focused on simple feed-forward networks, that contain only linear layers. Our research started from the results established in [20], which state that in the case of non-negative weighted neural networks, tight Lipschitz bounds can be derived. We design several robust training algorithms, trying to achieve a good trade-off between robustness and performance.

In Chapter 3, we presented a real-time Robust Automatic Gesture Recognition system, leveraging sEMG signals and neural networks. We experimented on four distinct datasets, encompassing a variety of gestures, ranging from basic hand movements to more complex grasping gestures. We proposed several methods to finely control the Lipschitz constant of the model during training, using spectral norm constraints. The estimation of a precise Lipschitz constant was efficiently accomplished by directing our attention toward neural networks featuring positive weights, drawing inspiration from the approach outlined in [21].

Additionally, we delved into an analysis aimed at comprehending the extent to which the proposed theoretical solutions contribute to the improvement of the classifier robustness in various contexts. To achieve this objective, we evaluated our models in three scenarios. In the first one, we solidified the credibility of our solution by testing across multiple white-box and black-box attacks. Next, we tested the effect of noisy inputs, simulating perturbations akin to those encountered in

practical scenarios where sensors may add some artifacts during the acquisition process. Lastly, we assessed the final architecture performance in real-life experiments, where we showed that our robust model outperformed conventionally trained models.

Second, we extended our method towards different classes of non-negative feed-forward neural networks, and we proposed two novel architectures. For both of them, we indicated that we can derive tight Lipschitz bounds and presented constrained training mechanisms that ensure the robustness of the resulting models. We showed that our algorithms are not limited to classification problems, as we validated our solution in denoising audio music clips corrupted by various levels of additive white noise.

In Chapter 4, we first introduced a novel class of neural networks inspired by MIMO filters. This class, named ACNN, functions as an intermediary solution, bridging the conceptual gap between CNNs and FCNs. Our proposed contribution includes a training approach that constrains the network Lipschitz constant, bolstering its robustness against adversarial noise. Moreover, we introduce another original architecture named RCFF-Net, operating within the complex-valued domain, and derived comprehensive bounds for its Lipschitz constant. To effectively train the proposed structures while simultaneously managing their global Lipschitz constant, we established carefully crafted constrained learning strategies. In order to demonstrate the versatility of our solution beyond classification problems, we evaluated both the ACNN and RCFF architectures within the context of audio signal denoising tasks.

Third, we have extended our contributions toward convolutional operators as well, and we introduced ABBA-Nets, a novel class of (almost) positive neural networks. A very important aspect is that we proved that ABBA-Nets are as expressive as arbitrary-signed networks while enjoying all the advantages that non-negative weighted neural networks possess. This showed we were able to train robust DNNs and CNNs models that were validated in computer vision tasks.

Chapter 5 highlights the contributions related to ABBA networks. In this part, we proved that ABBA layers exhibit a host of compelling properties. Demonstrating our solution versatility, we proved that any signed network can be transformed into an ABBA form—a property applicable to both fully connected and convolutional neural networks.

We also derived universal approximation theorems tailored to networks incorporating non-negatively weighted ABBA layers. To efficiently manage the Lipschitz constant of ABBA networks, we proposed a method that is applicable to both fully connected and convolutional scenarios. Our empirical validation involved subjecting ABBA networks to numerical experiments using standard image datasets. The

results underscored the remarkable performance of ABBA networks, particularly in cases involving small models. Notably, these networks showcased substantial gains in both performance and robustness compared to networks relying solely on non-negative weights. Moreover, robust ABBA networks proved to be competitive from a stability viewpoint with other state-of-the-art defense strategies.

6.2 . Perspectives

In this section, we propose some possible extensions of the aforementioned methods that could be worth investigating in future works.

6.2.1 . Training 1-Lipschitz denoisers

A possible way to extend the work presented in this thesis would be leveraging our established methods for controlling the Lipschitz constant of neural networks to generate 1-Lipschitz denoisers, as presented in [186]. These denoisers could be integrated into unrolled optimization or plug-and-play architectures, enabling the resolution of sophisticated signal/image processing problems, such as multichannel audio reverberation. To ensure the convergence of such schemes, it becomes paramount to impose nonexpansivity constraints on the denoiser. This would not only guarantee the stability of the optimization process but also pave the way for more effective and efficient solutions in the realm of audio signal processing.

6.2.2 . Expanding the applications of complex-valued neural networks

In future works, it would be interesting to apply RCFF-Net to a larger panel of signal processing applications involving complex-valued data, like audio unmixing where robust CVNNs could play a pivotal role. Additionally, in the domain of magnetic resonance imaging (MRI), where signals are inherently complex due to their definition in Fourier space (k-space), the robustness guarantees offered by our methods could help enhance the image reconstruction quality.

Furthermore, there is an exciting prospect in exploring the extension of our ABBA networks to the complex case. This endeavour would open up a wealth of possibilities for employing ABBA-inspired solutions to tackle complex-valued data across various fields, from signal processing to image reconstruction.

6.2.3 . Controlling the Lipschitz constant of more complex layer structures

Given the progress made in this thesis, particularly in the effective management of the Lipschitz constant to enhance the stability of linear and convolutional layers within neural networks, a compelling prospect emerges for future research endeavours. Extending our current methodologies to encompass other layer structures represents a promising next step in advancing the field of neural network robustness.

It is noteworthy that a crucial initial step has been accomplished by adapting our methodology to the context of graph neural networks, and some preliminary results have been already obtained [187]. Expanding our efforts to more complex layer structures, such as recurrent layers, attention mechanisms [188], and transformers [189], presents a big challenge. These layer structures play pivotal roles in a wide array of applications, including natural language processing, sequential data analysis, and graph-based tasks. Adapting our techniques to address the unique challenges posed by these layers is likely to yield valuable insights and significantly enhance the overall robustness of neural networks in a broader spectrum of applications.

Additionally, designing methods for controlling the Lipschitz constant of more complex architectures and scaling our methods to larger models would help us build robust neural networks for more sophisticated tasks. This endeavour not only aligns with the ongoing pursuit of creating robust machine learning models but also ensures that our research continues to contribute meaningfully to the evolving landscape of AI security and reliability.

6.2.4 . Combining Lipschitz control with other certifiable defenses

In the context of improving neural networks' stability against adversarial threats, a promising avenue for future research lies in the integration of our current Lipschitz constant control mechanisms with complementary defense strategies. Of particular interest is the potential synergy between our approach and certified defenses, such as GloRoNets [78]. This would be possible if we insert our projection-based training into the GloRoNet pipeline. Certified defenses provide rigorous guarantees on model robustness for a certain perturbation, and by incorporating our Lipschitz control methods, we can enhance the overall robustness of the resulting model. Additionally, we could also combine our method with empirical defenses, such as AT [66]. The fusion of both techniques could yield a substantially more potent defense mechanism [190]. This combined approach has the potential to not only thwart adversarial attacks but also provide provable bounds on the model performance under various attacks.

Furthermore, exploring the combination of our Lipschitz control techniques with other state-of-the-art defense strategies, including adversarial training [66] and defensive distillation [41], can offer a multifaceted defense framework capable of withstanding a wider spectrum of threats. This pursuit would contribute to the creation of more reliable and trustworthy AI systems.

6.2.5 . Studying the effect of other regularization techniques

Another interesting direction to follow would be the comprehensive study of the effects of various regularization techniques on the stability of the model. While a preliminary examination was conducted in Chapter 3 it would be compelling to broaden this endeavour across all our model architectures. Existing literature has proposed similar studies [122, 191], yet there remains a relative scarcity of research in this particular direction. Thus, an extensive analysis holds the potential to unravel deeper insights into the intricacies of the learning process within neural networks.

By systematically investigating the impact of diverse regularization strategies, ranging from ℓ_1 and ℓ_2 regularization to dropout and batch normalization, we can gain a holistic understanding of how these techniques influence network robustness. This deeper comprehension can guide us in the development of more effective methods for constructing robust models.

Additionally, such an inquiry can shed light on the interplay between the regularization effect and Lipschitz control. Ultimately, this line of research has the potential to significantly advance our knowledge in the quest for building neural networks that are not only accurate but also highly robust when facing adversarial challenges.

6.2.6 . Extending to other distances

Extending our current methods for controlling the robustness of neural networks to encompass other metrics is another research perspective. Presently, our techniques primarily address ℓ_2 perturbations, but the practicality of real-world systems demands a more comprehensive approach [40]. Real-life adversarial inputs can stem from a variety of sources, including faulty sensors or environmental anomalies, and these may not always adhere to the ℓ_2 norm. Thus, our efforts should expand to incorporate other distance metrics, such as ℓ_1, ℓ_∞ norms, or even the ℓ_0 pseudo-norm. Computing the Lipschitz constant with respect to such measures might also be sometimes easier. By embracing a broader range of distances, we can build systems with stability guarantees that are resilient to a wider spectrum of adversarial inputs. This undertaking can improve the pragmatic use of our methodologies and also aligns with the crucial objective of strengthening neural networks to address real-scenario threats.

Bibliography

[1] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proc. ACM Conf. on Comp. Comm. Sec.*, page 1528–1540, 2016.

[2] Cristina Andronache, Marian Negru, Ana Neacșu, George Cioroiu, Anamaria Rădoi, and Corneliu Burileanu. Towards extending real-time EMG-based gesture recognition system. In *Proc. IEEE Int. Conf. Telecomm. Signal Porcess.*, pages 301–304, 2020.

[3] Jianxing He, Sally L Baxter, Jie Xu, Jiming Xu, Xingtao Zhou, and Kang Zhang. The practical implementation of artificial intelligence technologies in medicine. *Nat. med.*, 25(1):30–36, 2019.

[4] Peiliang Li, Xiaozhi Chen, and Shaojie Shen. Stereo R-CNN based 3D object detection for autonomous driving. In *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, pages 7644–7652, 2019.

[5] NI Widiastuti. Convolution neural network for text mining and natural language processing. In *IOP Conf.: Mater. Sci. Eng.*, volume 662, 2019.

[6] Jinxian Qi, Guozhang Jiang, Gongfa Li, Ying Sun, and Bo Tao. Intelligent human-computer interaction based on surface EMG gesture recognition. *IEEE Access*, 7:61378–61387, 2019.

[7] David Gamarnik, Eren C. Kızıldağ, and Ilias Zadik. Self-regularity of output weights for overparameterized two-layer neural networks. In *Proc. IEEE Int. Symp. Info. Theory*, pages 819–824, 2021.

[8] Suzanna Parkinson, Greg Ongie, and Rebecca Willett. Linear neural network layers promote learning single- and multiple-index models. In *arXiv:2305.15598*, 2023.

[9] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *Proc. Int. Conf. Learn. Represent.*, 2014.

[10] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *Proc. Int. Conf. Learn. Represent.*, 2016.

[11] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. Hidden voice commands. In *USENIX Security Symp.*, pages 513–530, 2016.

[12] Miyato Takeru, Kataoka Toshiki, Koyama Masanori, and Yoshida Yuichi. Spectral normalization for generative adversarial networks. In *Int. Conf. Learn. Represent.*, 2018.

[13] Chuan Guo, Brian Karrer, Kamalika Chaudhuri, and Laurens van der Maaten. Bounding training data reconstruction in private (deep) learning. In *Proc. Int. Conf. Machine Learn.*, pages 8056–8071, 2022.

[14] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Proc. Int. Conf. Learn. Represent.*, 2015.

[15] Ana Neacșu, Jean-Christophe Pesquet, and Corneliu Burileanu. Accuracy-robustness trade-off for positively weighted neural networks. In *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, pages 8389–8393, 2020.

[16] Kevin Scaman and Aladin Virmaux. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *Proc. Ann. Conf. Neur. Inform. Proc. Syst.*, pages 3839–3848, 2018.

[17] Harris Drucker and Yann LeCun. Improving generalization performance using double backpropagation. *IEEE Trans. Neural Netw. Learn. Syst.*, 3:991–997, 1992.

[18] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Proc. Ann. Conf. Neur. Inform. Proc. Syst.*, pages 5767–5777, 2017.

[19] Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. In *Proc. Int. Conf. Mach. Learn.*, pages 854–863, 2017.

[20] Patrick L Combettes and Jean-Christophe Pesquet. Lipschitz certificates for layered network structures driven by averaged activation operators. In *J. Math. Data Sci.*, volume 2, pages 529–557, 2020.

[21] Jan. Chorowski and Jacek M. Zurada. Learning understandable neural networks with nonnegative weight constraints. *IEEE Trans. Neural Netw. Learn. Syst.*, 26(1):62–69, 2015.

[22] Adi Shamir, Odelia Melamed, and Oriel BenShmuel. The dimpled manifold model of adversarial examples in machine learning. *arXiv preprint arXiv:2106.10151*, 2021.

[23] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *Proc. Int. Conf. Mach. Learn.*, pages 284–293, 2018.

[24] Ling Huang, Anthony D. Joseph, Blaine Nelson, Benjamin I.P. Rubinstein, and J. D. Tygar. Adversarial machine learning. In *Proc. of the 4th ACM Workshop on Sec. and Art. Intell.*, AISec '11, page 43–58, 2011.

[25] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, pages 1625–1634, 2018.

[26] Alexey Kurakin, Zhengyou Zhang, and Zicheng Liu. A real time system for dynamic hand gesture recognition with a depth sensor. In *Proc. IEEE European Signal Processing Conf.*, pages 1975–1979, 2012.

[27] Xiaogang Xu, Hengshuang Zhao, and Jiaya Jia. Dynamic divide-and-conquer adversarial training for robust semantic segmentation. In *Proc. IEEE Int. Conf. Comput. Vis.*, pages 7486–7495, 2021.

[28] Kavya Gupta, Jean-Christophe Pesquet, Beatrice Pesquet-Popescu, Fragkiskos D Malliaros, and Fateh Kaakai. An adversarial attacker for neural networks in regression problems. In *Proc. Int. Joint Conf. Art. Intell.*, 2021.

[29] Battista Biggio, Blaine Nelson, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Poisoning attacks against support vector machines. In *Proc. Int. Conf. Mach. Learn.*, pages 1467–1474, 2012.

[30] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proc. Int. Conf. Mach. Learn.*, page 1885–1894, 2017.

[31] Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison Frogs! Targeted clean-label poisoning attacks on neural networks. In *Proc. Ann. Conf. Neur. Inform. Proc. Syst.*, page 6106–6116, 2018.

[32] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Represent.*

[33] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proc. ACM workshop on Art. Intell. and Sec.*, pages 15–26, 2017.

[34] Debayan Deb, Jianbang Zhang, and Anil K Jain. Advfaces: Adversarial face synthesis. In *Proc. Int. Joint Conf. Bio.*, pages 1–10, 2020.

[35] Chaowei Xiao, Bo Li, Jun Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. In *Proc. Int. Joint Conf. Art. Intell.*, pages 3905–3911, 2018.

[36] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Proc. European Conf. Mach. Learn. Princ. Practice of Knowledge Disc. in Databases*, pages 387–402, 2013.

[37] Dong C Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Programm.*, 45(1-3):503–528, 1989.

[38] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, pages 2574–2582, 2016.

[39] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *IEEE Symp. Security Privacy*, 2016.

[40] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symp. Security and Privacy*, pages 39–57, 2017.

[41] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symp. Security Privacy*, pages 582–597, 2016.

[42] Jérôme Rony, Luiz G Hafemann, Luiz S Oliveira, Ismail Ben Ayed, Robert Sabourin, and Eric Granger. Decoupling direction and norm for efficient gradient-based ℓ_2 adversarial attacks and defenses. In *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, pages 4322–4330, 2019.

[43] Maura Pintor, Fabio Roli, Wieland Brendel, and Battista Biggio. Fast minimum-norm adversarial attacks through adaptive norm constraints. In *Proc. Ann. Conf. Neur. Inform. Proc. Syst.*, volume 34, pages 20052–20062, 2021.

[44] Jérôme Rony, Eric Granger, Marco Pedersoli, and Ismail Ben Ayed. Augmented Lagrangian adversarial attacks. In *Proc. IEEE Int. Conf. Comput. Vis.*, pages 7738–7747, 2021.

[45] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proc. ACM on Asia Conf. on Comp. and Comm. Sec.*, pages 506–519, 2017.

[46] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *Proc. Int. Conf. Mach. Learn.*, pages 2137–2146, 2018.

[47] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *J. Mach. Learn. Res.*, 15(1):949–980, 2014.

[48] Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Analysis of classifiers' robustness to adversarial perturbations. *Mach. Learn.*, 107(3):481–508, 2018.

[49] SM Moosavi Dezfooli, F Alhussein, F Omar, F Pascal, and S Stefano. Analysis of universal adversarial perturbations. *arXiv:1705.09554*, 2017.

[50] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Robustness of classifiers: from adversarial to random noise. *Proc. Ann. Conf. Neur. Inform. Proc. Syst.*, 29, 2016.

[51] Samuel Henrique Silva and Peyman Najafirad. Opportunities and challenges in deep learning adversarial robustness: A survey. In *arXiv:2007.00753*, 2020.

[52] Han Xu, Yao Ma, Hao-Chen Liu, Debayan Deb, Hui Liu, Ji-Liang Tang, and Anil K Jain. Adversarial attacks and defenses in images, graphs and text:A review. *Int. J. Autom. and Comp.*, 17:151–178, 2020.

[53] Chaoning Zhang, Philipp Benz, Chenguo Lin, Adil Karjauv, Jing Wu, and In So Kweon. A survey on universal adversarial attack. *arXiv:2103.01498*, 2021.

[54] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proc. Int. Conf. Mach. Learn.*, pages 833–840, 2011.

[55] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv:1412.5068*, 2014.

[56] Grigory Khromov and Sidak Pal Singh. Some fundamental aspects about Lipschitz continuity of neural network functions. In *arXiv:2302.10886*, 2023.

[57] Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. *Proc. Ann. Conf. Neur. Inform. Proc. Syst.*, 30:823–846, 2017.

[58] Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks. In *Proc. Ann. Conf. Neur. Inform. Proc. Syst.*, pages 11423–11434, 2019.

[59] Jean-Jacques Moreau. Proximité et dualité dans un espace hilbertien. *Bulletin de la Société mathématique de France*, 93:273–299, 1965.

[60] Patrick Combettes and Jean-Christophe Pesquet. Deep neural network structures solving variational inequalities. *Set-Valued Var. Anal.*, pages 1–28, 2020.

[61] Dongmian Zou, Radu Balan, and Maneesh Singh. On Lipschitz bounds of general convolutional neural networks. *IEEE Trans. Inform. Theory*, 66(3):1738–1759, 2019.

[62] Takeru Miyato, Shin ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. Distributional smoothing with virtual adversarial training, 2016.

[63] Cem Anil, James Lucas, and Roger Grosse. Sorting out Lipschitz function approximation. In *Proc. Int. Conf. Machine Learn.*, pages 291–301, 2019.

[64] Mathieu Serrurier, Franck Mamalet, Alberto González-Sanz, Thibaut Boissin, Jean-Michel Loubes, and Eustasio Del Barrio. Achieving robustness in classification using optimal transport with hinge regularization. In *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, pages 505–514, 2021.

[65] Yujia Huang, Huan Zhang, Yuanyuan Shi, J Zico Kolter, and Anima Anandkumar. Training certifiably robust neural networks with efficient local lipschitz bounds. *Proc. Ann. Conf. Neur. Inform. Proc. Syst.*, 34:22745–22757, 2021.

[66] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *Proc. Int. Conf. Learn. Represent.*, 2018.

[67] Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. Recent advances in adversarial training for adversarial robustness. *arXiv preprint arXiv:2102.01356*, 2021.

[68] Jiajin Zhang, Hanqing Chao, and Pingkun Yan. Toward adversarial robustness in unlabeled target domains. *IEEE Trans. Image Process.*, 32:1272–1284, 2023.

[69] Nanyang Ye, Qianxiao Li, Xiao-Yun Zhou, and Zhanxing Zhu. An annealing mechanism for adversarial training acceleration. *IEEE Trans. Neural Netw. Learn. Syst.*, 34(2):882–893, 2023.

[70] Damilola Adesina, Chung-Chu Hsieh, Yalin E Sagduyu, and Lijun Qian. Adversarial machine learning in wireless communications using rf data: A review. *IEEE Comm. Surveys & Tutorials*, 2022.

[71] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*, 2019.

[72] Nicholas Carlini, Guy Katz, Clark Barrett, and David L Dill. Ground-truth adversarial examples.

[73] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Proc. Int. Conf. Comp. Aided Verif.*, pages 97–117, 2017.

[74] Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM review*, 38(1):49–95, 1996.

[75] Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. *Proc. Ann. Conf. Neur. Inform. Proc. Syst.*, 30, 2017.

[76] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *Proc. Int. Conf. Mach. Learn.*, pages 5286–5295, 2018.

[77] Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *Proc. Ann. Conf. Neur. Inform. Proc. Syst.*, volume 31, 2018.

[78] Klas Leino, Zifan Wang, and Matt Fredrikson. Globally-robust neural networks. In *Proc. Int. Conf. Mach. Learn.*, pages 6212–6222, 2021.

[79] Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. *Proc. Ann. Conf. Neur. Inform. Proc. Syst.*, 31, 2018.

[80] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[81] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. Countering adversarial images using input transformations. In *Int. Conf. Learn. Represent.*, 2017.

[82] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. In *Int. Conf. Learn. Represent.*, 2018.

[83] Guneet S Dhillon, Kamyar Azizzadenesheli, Zachary C Lipton, Jeremy D Bernstein, Jean Kossaifi, Aran Khanna, and Animashree Anandkumar. Stochastic activation pruning for robust adversarial defense. In *Int. Conf. Learn. Represent.*

[84] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. In *Int. Conf. Learn. Represent..*

[85] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proc. ACM workshop on Art. Intell. and Sec.*, pages 3–14, 2017.

[86] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples. *CoRR*, 2017.

[87] Zhitao Gong, Wenlu Wang, and Wei-Shinn Ku. Adversarial and clean data are not twins. *arXiv e-prints*, pages arXiv–1704, 2017.

[88] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. In *Int. Conf. Learn. Represent.*

[89] Dan Hendrycks and Kevin Gimpel. Early methods for detecting adversarial images. *Int. Conf. Learn. Represent*, 2016.

[90] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *J. Mach. Learn. Res.*, 13(1):723–773, 2012.

[91] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

[92] Manfredo Atzori, Arjan Gijsberts, Claudio Castellini, Barbara Caputo, Anne-Gabrielle Mittaz Hager, Simone Elsig, Giorgio Giatsidis, Franco Bassetto, and Henning Müller. Electromyography data for non-invasive naturally-controlled robotic hand prostheses. *Sci. data*, (140053), 2014.

[93] Jolanta Pauk. Different techniques for EMG signal processing. *J. of Vibroeng.*, 10:571–576, 12 2008.

[94] NR Womack, NS Williams, JHM Holmfield, JFB Morrison, and KC Simpkins. New method for the dynamic assessment of anorectal function in constipation. *British J. of Surgery*, 72(12):994–998, 1985.

[95] Jeffrey R. Cram. *Introduction to surface electromyography*. Aspen Publishers, 1998.

[96] Ming Jin Cheok, Zaid Omar, and Mohamed Hisham Jaward. A review of hand gesture and sign language recognition techniques. *Int. J. Mach. Learn. Cyber.*, 10(1):131–153, 2019.

[97] Feng Wen, Zhongda Sun, Tianyiyi He, Qiongfeng Shi, Minglu Zhu, Zixuan Zhang, Lianhui Li, Ting Zhang, and Chengkuo Lee. Machine learning glove using self-powered conductive superhydrophobic triboelectric textile for gesture recognition in VR/AR applications. *Adv. Sci.*, 7(14):2000261, 2020.

[98] Xiaojie Gao, Yueming Jin, Qi Dou, and Pheng-Ann Heng. Automatic gesture recognition in robot-assisted surgery with reinforcement learning and tree search. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 8440–8446, 2020.

[99] Okan Kopuklu, Neslihan Kose, and Gerhard Rigoll. Motion fused frames: Data level fusion strategy for hand gesture recognition. In *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, pages 2103–2111, 2018.

[100] Trevor J Darrell, Irfan A Essa, and Alex P Pentland. Task-specific gesture analysis in real-time using interpolated views. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(12):1236–1242, 1996.

[101] Zhiwen Yang, Du Jiang, Ying Sun, Bo Tao, Xiliang Tong, Guozhang Jiang, Manman Xu, Juntong Yun, Ying Liu, Baojia Chen, and Jianyi Kong. Dynamic gesture recognition using surface EMG signals based on multi-stream residual network. *Front. in Bioengin. and Biotech.*, 9, 2021.

[102] Natalia Neverova, Christian Wolf, Graham Taylor, and Florian Nebout. Moddrop: adaptive multi-modal gesture recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(8):1692–1706, 2015.

[103] Enea Ceolini, Charlotte Frenkel, Sumit Bam Shrestha, Gemma Taverni, Lyes Khacef, Melika Payvand, and Elisa Donati. Hand-gesture recognition based on EMG and event-based camera sensor fusion: A benchmark in neuromorphic computing. *Front. in Neurosci.*, 14, 2020.

[104] Yujian Jiang, Lin Song, Junming Zhang, Yang Song, and Ming Yan. Multi-category gesture recognition modeling based on sEMG and IMU signals. *Sensors*, 22(15), 2022.

[105] Fulai Peng, Cai Chen, Danyang Lv, Ningling Zhang, Xingwei Wang, Xikun Zhang, and Zhiyong Wang. Gesture recognition by ensemble extreme learning machine based on surface electromyography signals. *Front. in Human Neurosci.*, 16, 2022.

[106] Rami N Khushaba and Sarath Kodagoda. Electromyogram (EMG) feature reduction using mutual components analysis for multifunction prosthetic fingers control. In *Int. Conf. Control Autom. Robotics & Vision*, 5–7 Dec 2012.

[107] Jonghwa Kim, Stephan Mastnik, and Elisabeth André. EMG-based hand gesture recognition for realtime biosignal interfacing. In *Proc. Int. Conf. Intell. User Interfac.*, pages 30–39, 2008.

[108] Alvaro David Orjuela-Cañón, Andrés F Ruíz-Olaya, and Leonardo Forero. Deep neural network for EMG signal classification of wrist position: Preliminary results. In *Proc. IEEE Latin American Conf. Comput. Intell.*, pages 1–5, 2017.

[109] Patrick Combettes and Jean-Christophe Pesquet. Proximal thresholding algorithm for minimization over orthonormal bases. *SIAM J. on Optim.*, 18(4):1351–1376, 2008.

[110] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *Proc. Int. Conf. Learn. Represent.*, 30 Apr.–03 May 2018.

[111] Fabian Latorre, Paul Rolland, and Volkan Cevher. Lipschitz constant estimation of neural networks via sparse polynomial optimization. In *Int. Conf. on Learning Represent.*, 2020.

[112] Tong Chen, Jean-Bernard Lasserre, Victor Magron, and Edouard Pauwels. Semialgebraic optimization for Lipschitz constants of ReLU networks. pages 19189–19200, 2020.

[113] Patricia Pauli, Anne Koch, Julian Berberich, Paul Kohler, and Frank Allgöwer. Training robust neural networks using Lipschitz bounds. In *IEEE Cont. Syst. Lett.*, volume 6, pages 121–126, 2021.

[114] Patrick L. Combettes and Jean-Christophe Pesquet. Fixed point strategies in data science. *IEEE Trans. Sig. Proc.*, 69:3878–3905, 2021.

[115] Nikos Komodakis and Jean-Christophe Pesquet. Playing with duality: An overview of recent primal-dual approaches for solving large-scale optimization problems. *IEEE Signal Process. Mag.*, 32(6):31–54, 2015.

[116] Patrick L Combettes, Dinh Dung, and Bang Cong Vu. Dualization of signal recovery problems. *Set-Valued Var. Anal.*, 18(3–4):373–404, 2010.

[117] Ulysse Côté-Allard, Cheikh Latyr Fall, Alexandre Drouin, Alexandre Campeau-Lecours, Clément Gosselin, Kyrre Glette, François Laviolette, and Benoit Gosselin. Deep learning for electromyographic hand gesture signal classification using transfer learning. *IEEE Trans. Neural Syst. Rehabilitation Eng.*, 27(4):760–771, 2019.

[118] Stefano Pizzolato, Luca Tagliapietra, Matteo Cognolato, Monica Reggiani, Henning Müller, and Manfredo Atzori. Comparison of six electromyography acquisition setups on hand movement classification tasks. *PLOS ONE*, 12(10), 10 2017.

[119] Manfredo Atzori, Arjan Gijsberts, Ilja Kuzborskij, Simone Elsig, Anne-Gabrielle Mittaz Hager, Olivier Deriaz, Claudio Castellini, Henning Müller, and Barbara Caputo. Characterization of a benchmark database for myoelectric movement classification. *IEEE Trans. on Neural Syst. and Rehabilitation Eng.*, 23(1):73–83, 2014.

[120] Cristina Andronache, Marian Negru, Ioana Bădiţoiu, George Cioroiu, Ana Neacsu, and Corneliu Burileanu. Automatic gesture recognition framework based on forearm emg activity. In *Proc. IEEE Int. Conf. Telecomm. Signal Porcess.*, pages 284–288, 2022.

[121] Xiangrui Wang, Lu Tang, Qibin Zheng, Xilin Yang, and Zhiyuan Lu. IRDC-Net: An inception network with a residual module and dilated convolution for sign language recognition based on surface electromyography. *Sensors*, 23(13), 2023.

[122] Kavya Gupta, Beatrice Pesquet-Popescu, Fateh Kaakai, and Jean-Christophe Pesquet. A quantitative analysis of the robustness of neural networks for tabular data. In *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, pages 8057–8061, 2021.

[123] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. There is no free lunch in adversarial robustness (but there are unexpected benefits). In *arXiv:1805.12152*, 2018.

[124] Jonas Geiping, Liam H Fowl, W Ronny Huang, Wojciech Czaja, Gavin Taylor, Michael Moeller, and Tom Goldstein. Witches' Brew: Industrial scale data poisoning via Gradient Matching. In *Int. Conf. Learn. Represent.*, 2020.

[125] HH Bauschke and PL Combettes. Convex analysis and monotone operator theory in Hilbert spaces, 2019.

[126] Patrick L Combettes and Valérie R Wajs. Signal recovery by proximal forward-backward splitting. *SIAM Multiscale Model. Simul.*, 4(4):1168–1200, 2005.

[127] Antonin Chambolle and Ch Dossal. On the convergence of the iterates of the fast iterative shrinkage/thresholding algorithm. *Springer J. Optim. Theory Appl.*, 166(3):968–982, 2015.

[128] Laifeng Huang, Xingxin He, Leyuan Fang, Hossein Rabbani, and Xiangdong Chen. Automatic classification of retinal optical coherence tomography images with layer guided convolutional neural network. *IEEE Signal Process. Lett.*, 26(7):1026–1030, 2019.

[129] Yushi Chen, Kaiqiang Zhu, Lin Zhu, Xin He, Pedram Ghamisi, and Jón Atli Benediktsson. Automatic design of convolutional neural network for hyperspectral image classification. *IEEE Geosci. Rem. Sens. Lett.*, 57(9):7048–7066, 2019.

[130] Saparudin Saparudin, Reza Firsandaya Malik, Erwin Erwin, M Fachrurrozi, Sukemi Sukemi, Puji Rachmawati, Ari Susanto, and Bambang Suprihatin. Convolutional neural networks for realtime multi-faces verification with occlusion. In *Proc. Int. Conf. Electr. Eng. Comp. Sci.*, pages 235–240, 2019.

[131] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *Proc. Eur. Conf. Comput. Vis.*, pages 850–865, 2016.

[132] Se Rim Park and Jinwon Lee. A fully convolutional neural network for speech enhancement. *Proc. Interspeech*, pages 1993–1997, 2017.

[133] Zhiheng Ouyang, Hongjiang Yu, Wei-Ping Zhu, and Benoit Champagne. A fully convolutional neural network for complex spectrogram processing in speech enhancement. In *Proc. Int. Conf. Acoust. Speech Signal Process.*, pages 5756–5760, 2019.

[134] Gregor Urban, Krzysztof J Geras, Samira Ebrahimi Kahou, Ozlem Aslan, Shengjie Wang, Rich Caruana, Abdelrahman Mohamed, Matthai Philipose, and Matt Richardson. Do deep convolutional nets really need to be deep and convolutional? *Proc. Int. Conf. Learning Represent.*, 2017.

[135] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *Proc. Int. Conf. Learning Represent.*, 2017.

[136] Zhouhan Lin, Roland Memisevic, and Kishore Konda. How far can we go without convolution: Improving fully-connected networks. *Proc. Int. Conf. Learning Represent.*, 2016.

[137] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Proc. Int. Conf. Learning Represent.*, 2015.

[138] Joshua Bassey, Lijun Qian, and Xianfang Li. A survey of complex-valued neural networks. *arXiv:2101.12249*, 2021.

[139] Akira Hirose. Applications of complex-valued neural networks to coherent optical computing using phase-sensitive detection scheme. *Inf. Sci.-Applications*, 2(2):103–117, 1994.

[140] Taehwan Kim and Tülay Adalı. Approximation by fully complex multilayer perceptrons. *Neural computation*, 15(7):1641–1666, 2003.

[141] Igor Aizenberg. *Complex-valued neural networks with multi-valued neurons*, volume 353. Springer, 2011.

[142] J. A. Barrachina, C. Ren, C. Morisseau, G. Vieillard, and J.-P. Ovarlez. Complex-valued vs. real-valued neural networks for classification perspectives: An example on non-circular data. In *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, pages 2990–2994, 2021.

[143] E Cole, J Cheng, J Pauly, and S Vasanawala. Analysis of deep complex-valued convolutional neural networks for MRI reconstruction and phase-focused applications. *Magn. Reson. Med.*, 86(2):1093–1109, 2021.

[144] Othmane-Latif Ouabi, Radmila Pribic, and Sorin Olaru. Stochastic Complex-valued Neural Networks for Radar. In *Proc. IEEE European Signal Process. Conf.*, pages 1442–1446, 2021.

[145] Jesper Sören Dramsch, Mikael Lüthje, and Anders Nymark Christensen. Complex-valued neural networks for machine learning on non-stationary physical data. *Computers & Geosci.*, 146:104643, 2021.

[146] Xinming Cheng, Jiangnan He, Jianbiao He, and Honglei Xu. Cv-CapsNet: Complex-valued Capsule Network. *IEEE Access*, 7:85492–85499, 2019.

[147] Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. Dynamic routing between capsules. In *Proc. Ann. Conf. Neur. Inform. Proc. Syst.*, volume 30, pages 3856–3866, 2017.

[148] Patrick L Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer-Verlag, New York, 2011.

[149] H H Bauschke and P L Combettes. Convex Analysis and Monotone Operator Theory in Hilbert spaces. *2nd ed., corrected printing. New York: Springer*, 2019.

[150] G.M. Georgiou and C. Koutsougeras. Complex domain backpropagation. *IEEE Tran. on Circ. and Syst. II: Analog and Digital Signal Process.*, 39(5):330–334, 1992.

[151] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. In *Nature*, volume 401, pages 788–791, 1999.

[152] Jan Chorowski and Jacek M. Zurada. Learning understandable neural networks with nonnegative weight constraints. In *IEEE Trans. Neural Net. and Learn. Syst.*, volume 26, pages 62–69, 2015.

[153] Qingyang Wang, Michael A Powell, Ali Geisa, Eric Bridgeford, and Joshua T Vogelstein. Why do networks have inhibitory/negative connections? *arXiv:2208.03211*, 2022.

[154] Babajide O. Ayinde and Jacek M. Zurada. Deep Learning of constrained autoencoders for enhanced understanding of data. In *IEEE Trans. Neural Net. and Learn. Syst.*, volume 29, pages 3969–3979, 2018.

[155] Ilias Diakonikolas, Daniel M Kane, Vasilis Kontonis, and Nikos Zarifis. Algorithms and SQ lower bounds for PAC learning one-hidden-layer ReLU networks. In *Proc. Conf. Learn. Theory*, pages 1514–1539, 2020.

[156] Ana Neacșu, Jean-Christophe Pesquet, and Corneliu Burileanu. EMG-based automatic gesture recognition using robust neural networks (extended version). In *Preprint submitted to ACM*, 2022.

[157] Brandon Amos, Lei Xu, and J. Zico Kolter. Input convex neural networks. In *Proc. Int. Conf. Machine Learn.*, volume 70, pages 146–155, 2017.

[158] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. In *Proc. Ann. Conf. Neur. Inform. Proc. Syst.*, volume 6, pages 737–744, 1993.

[159] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, pages 1701–1708, 2014.

[160] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *Proc. Int. Conf. Machine Learn.*, volume 2, 2015.

[161] Anfeng He, Chong Luo, Xinmei Tian, and Wenjun Zeng. A twofold siamese network for real-time object tracking. In *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, pages 4834–4843, 2018.

[162] Martin Gleize, Eyal Shnarch, Leshem Choshen, Lena Dankin, Guy Moshkowich, Ranit Aharonov, and Noam Slonim. Are you convinced? Choosing the more convincing evidence with a Siamese Network. In *Proc. Ann. Meeting of the Assoc. for Comp. Linguistics*, pages 967–976, 2019.

[163] Kyriakos D. Apostolidis and George A. Papakostas. A survey on adversarial deep learning robustness in medical image analysis. In *Electronics*, volume 10, 2021.

[164] Sebastian Neumayer, Alexis Goujon, Pakshal Bohra, and Michael Unser. Approximation of Lipschitz functions using deep spline neural networks. In *J. Math. Data Sci.*, volume 5, pages 306–322, 2023.

[165] Aman Sinha, Hongseok Namkoong, and John Duchi. Certifiable distributional robustness with principled adversarial training. In *Proc. Int. Conf. Learn. Represent.*, 2018.

[166] Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In *Proc. Int. Conf. Learn. Represent.*, 2020.

[167] Zhizhong Li and Derek Hoiem. Learning without forgetting. In *IEEE Trans. Pattern Anal. Machine Intell.*, volume 40, pages 2935–2947, 2017.

[168] Ali Shafahi, Parsa Saadatpanah, Chen Zhu, Amin Ghiasi, Christoph Studer, David Jacobs, and Tom Goldstein. Adversarially robust transfer learning. In *Proc. Int. Conf. Learn. Represent.*, 2020.

[169] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *Proc. Int. Conf. Machine Learn.*, pages 7472–7482, 2019.

[170] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. In *Proc. Int. Conf. Learn. Represent.*, 2018.

[171] Francesco Croce, Maksym Andriushchenko, and Matthias Hein. Provable robustness of ReLU networks via maximization of linear regions. In *Proc. Int. Conf. Art. Intell. and Statistics*, pages 2057–2066, 2019.

[172] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Timothy A Mann, and Pushmeet Kohli. A dual approach to verify and train deep networks. In *Proc. Int. Conf. Art. Intell.*, pages 6156–6160, 2019.

[173] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *Proc. Int. Conf. Machine Learn.*, pages 1310–1320, 2019.

[174] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *Proc. IEEE Symp. Security and Privacy*, pages 656–672, 2019.

[175] Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Certified adversarial robustness with additive noise. In *Proc. Ann. Conf. Neur. Inform. Proc. Syst.*, volume 32, 2019.

[176] Hadi Salman, Jerry Li, Ilya Razenshteyn, Pengchuan Zhang, Huan Zhang, Sebastien Bubeck, and Greg Yang. Provably robust deep learning via adversarially trained smoothed classifiers. In *Proc. Ann. Conf. Neur. Inform. Proc. Syst.*, volume 32, pages 11292–11303, 2019.

[177] Runtian Zhai, Chen Dan, Di He, Huan Zhang, Boqing Gong, Pradeep Ravikumar, Cho-Jui Hsieh, and Liwei Wang. MACER: Attack-free and scalable robust training via maximizing certified radius. In *Proc. Int. Conf. Learn. Represent.*, 2020.

[178] Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. In *Neural Networks*, volume 6, pages 861–867, 1993.

[179] Patrick Kidger and Terry Lyons. Universal approximation with deep narrow networks. In *Proc. Conf. Learn. Theory*, pages 2306–2327, 2020.

[180] P.P Vaidyanathan. Multirate Systems and Filter Banks. In *Prentice Hall (NJ)*, 1993.

[181] Laurence Moroney. Rock, paper, scissors dataset, 2019.

[182] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proc. IEEE Int. Conf. Comput. Vis.*, volume 4, pages 3730–3738, 2015.

[183] John R Silvester. Determinants of block matrices. *Gaz. Math.*, 84(501):460–467, 2000.

[184] P P Vaidyanathan. Multirate Systems and Filter Banks. *Prentice Hall (NJ)*, 1993.

[185] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11):2278–2324, 1998.

[186] Audrey Repetti, Matthieu Terris, Yves Wiaux, and Jean-Christophe Pesquet. Dual forward-backward unfolded network for flexible Plug-and-Play. In *Proc. European Signal Processing Conference*, pages 957–961, 2022.

[187] Simona Juvină, Ana Neacsu, Jean-Christophe Pesquet, and Corneliu Burileanu. Training graph neural network subject to a tight Lipschitz constraint. In *Submitted to Proc. Ann. Conf. Neur. Inform. Proc. Syst.*, 2023.

[188] Hyunjik Kim, George Papamakarios, and Andriy Mnih. The Lipschitz constant of self-attention. In *Int. Conf. on Machine Learn.*, pages 5562–5571, 2021.

[189] Xianbiao Qi, Jianan Wang, Yihao Chen, Yukai Shi, and Lei Zhang. Lipsformer: Introducing Lipschitz continuity to vision transformers. *arXiv:2304.09856*, 2023.

[190] Kavya Gupta. *Stability Quantification of Neural Networks*. PhD thesis, Université Paris-Saclay, 2023.

[191] Philipp Benz, Chaoning Zhang, and In So Kweon. Batch Normalization increases adversarial vulnerability and decreases adversarial transferability: A non-robust feature perspective. In *Proc IEEE Int. Conf. on Comp. Vision*, pages 7818–7827, 2021.