Titre: Title:	Stabilizing and Enhancing Learning for Deep Complex and Real Neural Networks
Auteur: Author:	Chiheb Trabelsi
Date:	2019
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Trabelsi, C. (2019). Stabilizing and Enhancing Learning for Deep Complex and Real Neural Networks [Thèse de doctorat, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/3945/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/3945/
Directeurs de recherche: Advisors:	Christopher J. Pal
Programme: Program:	Génie informatique

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Stabilizing and Enhancing Learning for Deep Complex and Real Neural Networks

CHIHEB TRABELSI

Département de génie informatique et génie logiciel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*

Génie informatique

Juillet 2019

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Stabilizing and Enhancing Learning for Deep Complex and Real Neural Networks

présentée par **Chiheb TRABELSI**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Michel GAGNON, président
Christopher J. PAL, membre et directeur de recherche
Giovanni BELTRAME, membre
Roger GROSSE, membre externe

DEDICATION

Dedicated to the memory of
Ommi Chedhlia, Bibi Aichoucha,
my aunt Nabiha and my uncle Slim.

ACKNOWLEDGEMENTS

First of all, I would like to express my deepest gratitude to my thesis's advisor, Chris Pal. Chris has been an excellent academic supervisor who has guided me through l'Ecole Polytechnique de Montréal and the research thesis with knowledgeable generosity and professional insight, making me a better researcher. In addition he has been a caring friend and mentor.

This is the opportunity to testify how delighted I was to be able to work with you Chris for all these years. Your knowledge and perspicacity impressed me, but I will always remember that verve, passion and desire to share, which enormously inspired me far beyond all the difficulties encountered during my Ph.D. journey. Thank you so much.

I feel fortunate to have been offered the opportunity to become member of the vivacious Quebec Artificial Intelligence Institute (Mila) and Element AI. This allowed me to collaborate and share my research orientations and findings with leading experts in the fascinating field of deep learning. I have learned a lot from them and I warmly thank all of them.

Thanks also to the many great friends I have come across throughout my doctoral studies. They have seen me through the nastiest and finest, and have provided help through fruitful long discussions, and insightful comments. Some special mention goes to Olexa, Sandeep, Eugene, Alex, Anirudh and Valentin.

I am overwhelmed by the love and gratitude of my family, who has given me unwavering support throughout my journey, despite the ups and downs. I express my gratitude to my loved ones : my dad Abdelwahed, my mum Monia, and my brother Amine for their prayers, and their endless love.

RÉSUMÉ

Dans cette thèse nous proposons un ensemble de contributions originales sous la forme de trois articles relatifs aux réseaux de neurones profonds réels et complexes. Nous abordons à la fois des problèmes théoriques et pratiques liés à leur apprentissage. Les trois articles traitent des méthodes conçues pour apporter des solutions aux problèmes de l'instabilité observée au cours de l'entrainement des réseaux, notamment le problème notoire de dilution et d'explosion des gradients ou «vanishing and exploding gradients » lors de l'entrainement des réseaux de neurones profonds.

Nous proposons dans un premier temps la conception de modules d'entrainement appropriés, désignés par «building blocks», pour les réseaux de neurones profonds à valeurs complexes. Notre proposition comporte des méthodes d'initialisation et de normalisation ainsi que des fonctions d'activation des unités neuronales. Les modules conçus sont par la suite utilisés pour la spécification d'architectures profondes à valeurs complexes dédiées à accomplir diverses tâches. Ceci comprend des tâches de vision par ordinateur, de transcription musicale, de prédiction du spectre de la parole, d'extraction des signaux et de séparation des sources audio. Finalement nous procédons à une analyse détaillée de l'utilité de l'hypothèse contraignante d'orthogonalité généralement adoptée pour le paramétrage de la matrice de transition à travers les couches des réseaux de neurones réels récurrents.

Tout au long de ce manuscrit, nous montrons l'utilité des méthodes que nous proposons dans le contexte des défis et objectifs annoncés. Nous donnons ci-dessous un résumé de chaque article séparément.

Actuellement, la grande majorité des «building blocks» qui contiennent des techniques et des architectures destinées à l'apprentissage des réseaux neuronaux profonds reposent essentiellement sur des calculs et des représentations à valeurs réelles. Cependant, les représentations basées sur des nombres complexes ont commencé récemment à faire l'objet d'une attention particulière. Malgré leurs propriétés probantes, les réseaux profonds à valeurs complexes ont été négligés pendant un certain temps en raison de l'absence de «building blocks» nécessaires à leur entrainement. L'absence d'un tel cadre indique une lacune notable dans l'outillage de l'apprentissage profond.

Le premier article vise à combler cette lacune en fournissant une nouvelle représentation des réseaux de neurones complexes profonds et des méthodes d'apprentissage qui vont bien au-delà d'une simple généralisation théorique du cas réel. Plus spécifiquement, les éléments clés des réseaux de neurones profonds à valeurs complexes sont développés et appliqués aux réseaux à retransmission «feed-forward» convolutionnels et aux réseaux à mémoires courte et longue «LSTM» convolutionnels. Des convolutions à valeurs complexes et de nouveaux algorithmes pour la méthode de normalisation complexe par batch «Batch Normalization» et des stratégies d'initialisation de poids complexes constituent le fondement du cadre proposé. Les résultats d'une expérimentation intensive comportant des schémas d'entrainement de bout en bout, consolident notre cadre théorique. En effet, dans les cas étudiés, ces «Building Blocks» ont permis d'éviter des problèmes numériques ardus. Par ailleurs ils ont permis lors de l'entrainement d'apprendre des tâches multiples correspondant à des inputs aussi bien réels qu'intrinsèquement complexes.

Nous avons testé la performance des modèles profonds pour l'apprentissage des tâches de vision par ordinateur, de transcription musicale à l'aide du jeu de données MusicNet et de prédiction du spectre de la parole à l'aide du jeu de données TIMIT. Nous avons enregistré des résultats compétitifs aux réseaux réels pour les tâches de vision et réalisé l'état-de-l' art pour les tâches audio.

Le second article est une extension qui s'appuie sur les développements étayés dans le premier article et propose un nouveau pipeline «Deep Separator ou Deep Pitcher» pour l'extraction des signaux et la séparation automatique des sources cachées dans des environnements brouillés. Les représentations utilisent le domaine fréquentiel. Le «Deep Separator» est une nouvelle méthode de masquage basée sur une version complexe d'une modulation linéaire des inputs «Feature-wise Linear Modulation ou FiLM». Cette nouvelle méthode de masquage est la clé de voûte de notre architecture pour la récupération de l'information et la séparation des sources. Nous proposons également une nouvelle fonction de perte explicitement liée à la phase, qui est invariante en amplitude et par translation, prenant en compte les composantes complexes du spectrogramme. Nous la comparons par la suite à plusieurs autres fonctions de pertes définies dans les domaines temporel et fréquentiel. Les résultats expérimentaux sur les données du corpus standard de «Wall Street Journal» démontrent l'efficacité des séparateurs complexes profonds et illustrent le potentiel des réseaux complexes profonds à améliorer la performance lors de l'exécution des tâches de séparation des sources audio.

Le troisième article est une étude des problèmes liés à la nature des matrices orthogonales souvent employées pour faciliter l'apprentissage des réseaux de neurones réels récurrents et profonds. Les matrices de transition unitaires et les contraintes d'orthogonalité constituent une solution appropriée au problème de «vanishing gradients» observé notamment dans les réseaux neuronaux récurrents profonds. Néanmoins, l'application de contraintes strictes implique des coûts prohibitifs au niveau computationnel. Ces contraintes s'avèrent même obsolètes pour des tâches comme celles relatives au traitement du langage naturel.

Notre troisième article vise à évaluer l'utilité de la contrainte d'orthogonalité en contrôlant l'ampleur de l'écart de la matrice de transition par rapport à la variété de transformations orthogonales, et en mesurant la performance de généralisation du réseau de neurones récurrent qui lui est associé. Nous avons constaté que les contraintes strictes d'orthogonalité limitent la capacité de représentation du modèle entretenu, détériorent sa performance et ralentissent la vitesse de convergence du processus d'optimisation. Une stratégie de paramétrisation et de factorisation de la matrice de transition est proposée afin de majorer et minorer sa norme. Ceci permet de surveiller le degré d'expansion et de contraction de la norme des gradients lors de la propgation des activations et de la retro-propagation des gradients. En combinant la décomposition en valeurs singulières de la matrice de transition et l'algorithme de descente de gradient géodésique, nous exprimons explicitement l'ampleur de l'écart par rapport à l'orthogonalité en définissant une marge pour les valeurs singulières. Cette marge peut être apprise et optimisée. La stratégie ainsi définie nous permet d'observer et d'analyser les cas où la déviation de l'orthogonalité pourrait être bénéfique ou néfaste.

ABSTRACT

This thesis presents a set of original contributions in the form of three chapters on real and complex-valued deep neural networks. We address both theoretical issues and practical challenges related to the training of both real and complex-valued neural networks. First, we investigate the design of appropriate building blocks for deep complex-valued neural networks, such as initialization methods, normalization techniques and elementwise activation functions. We apply our theoretical insights to design building blocks for the construction of deep complex-valued architectures. We use them to perform various tasks in computer vision, music transcription, speech spectrum prediction, signal retrieval and audio source separation. We also perform an analysis of the usefulness of orthogonality for the hidden transition matrix in a real-valued recurrent neural network. Each of the three chapters are dedicated to dealing with methods designed to provide solutions to problems causing training instability, among them, the notorious problem of vanishing and exploding gradients during the training of deep neural networks.

Throughout this manuscript we show the usefulness of the methods we propose in the context of well known challenges and clearly identifiable objectives. We provide below a summary of the contributions within each chapter.

At present, the vast majority of building blocks, techniques, and architectures for training deep neural networks are based on real-valued computations and representations. However, representations based on complex numbers have started to receive increased attention. Despite their compelling properties complex-valued deep neural networks have been neglected due in part to the absence of the building blocks required to design and train this type of network. The lack of such a framework represents a noticeable gap in deep learning tooling.

The first research article presented in this thesis aims to fill this gap by providing new methods that go far beyond a simple theoretical generalization of real-valued neural networks. More specifically, we develop some key atomic components for complex-valued deep neural networks and apply them to convolutional feed-forward networks and Convolutional Long Short-Term Memory (Convolutional LSTM). Complex convolutions and new algorithms for complex batch-normalization and complex weight initialization strategies form the bedrock of the proposed framework. Results from intensive experimentation, with end-to-end training schemes, underpinned our theoretic framework. Indeed these building blocks have been shown to avoid numerical problems during training and thereby enable the use of complex-valued representations for learning various tasks with real and intrinsically complex inputs. We

show that such deep models are competitive with their real-valued counterparts. We test deep complex models on several computer vision tasks, on music transcription using the MusicNet dataset and on Speech Spectrum Prediction using the TIMIT dataset. We achieve state-of-the-art performance on these audio-related tasks.

The second article is an extension that builds on the results obtained in the first article to provide a novel fully complex-valued pipeline for automatic signal retrieval and signal separation in the frequency domain. We call this model a Deep Complex Separator. It is a novel masking-based method founded on a complex-valued version of Feature-wise Linear Modulation (FiLM). This new masking method is the key approach underlying our proposed speech separation architecture. We also propose a new explicitly phase-aware loss, which is amplitude and shift-invariant, taking into account the complex-valued components of the spectrogram. We compare it to several other phase-aware losses operating in both time and frequency domains. Experimental results on the standard Wall Street Journal Dataset demonstrate the effectiveness of the complex-valued deep separators, highlighting the potential of deep complex networks to improve performance on audio source separation tasks.

We provide a re-written version and extensions to a **third article**, which consists of a study that examines issues related to the nature of orthogonal matrices employed to train recurrent neural networks. Unitary transition matrices and orthogonal constraints are shown to provide an appropriate solution to the vanishing and exploding gradients problem in recurrent neural networks. However, imposing hard constraints has been shown to be computationally expensive and needless for some real-world tasks. This third chapter aims at assessing the utility of different types of soft and hard orthogonality constraints by controlling the extent of the deviation of the corresponding transition weight matrix from the manifold of orthogonal transformations and measuring the generalization performance of the associated recurrent neural network. Hard constraint of orthogonality are found to limit the model's representational power, worsens its performance, and slows down optimization convergence speed. A weight matrix factorization and parameterization strategy is proposed to bound matrix norms and therein monitor the degrees of expansion and contraction of the activations norm during the forward propagation and the gradients norm during the backpropagation. By combining Singular Value Decomposition of the hidden transition matrix and the geodesic gradient descent algorithm, we explicitly express the amount of deviation from orthogonality by setting a learnable margin for the singular values and optimize it. This allows us to observe and analyze cases where deviating from orthogonality could be either beneficial or detrimental.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
GPU	Graphics Processing Units
DNN	Deep Neural Network
NLP	Natural Language Processing
ASR	Automatic Speech Recognition
ANN	Artificial Neural Network
RV-NN	Real-Valued Neural Network
CV-NN	Complex-Valued Neural Network
RV-DNN	Real-Valued Deep Neural Network
CV-DNN	Complex-Valued Deep Neural Network
RNN	Recurrent Neural Network
CNN	Convolutional Neural Network
Resnet	Residual Network
GAN	Generative Adversarial Network
FFNN	Feed-Forward Neural Network
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
BN	Batch Normalization
WN	Weight Normalization
LN	Layer Normalization
CBN	Complex Batch Normalization
CLN	Complex Layer Normalization
FiLM	Feature-wise Linear Modulation
STFT	Short Time Fourier Transform
FFT	Fast Fourier Transform
MRI	Magnetic Reasoning Image
SAR	Synthetic Aperture Radar
VLBI	Very-Long-Baseline Interferometry
AD	Antenna Design
RI	Radar Imaging
ASP	Acoustic Signal Processing

UI	Ultrasonic Imaging
CSP	Communication Signal Processing
TPS	Traffic and Power Systems
ECG	Electrocardiogram
EEG	Electroencephalogram
LTI	Linear Time-Invariant
LSI	Linear Shift-Invariant
uRNN	unitary Recurrent Neural network
FC-uRNN	Full Capacity unitary Recurrent Neural Network
EURNN	Efficient Unitary RNN
KRU	Kronecker Recurrent Unit
FRU	Fourier Recurrent Unit
oRNN	Orthogonal Recurrent Neural Network
GORU	Gated Orthogonal Recurrent Unit
HRR	Holographic Reduced Representation
cgRNN	complex-valued gated Recurrent Neural Network
SNR	Signal-to-Noise Ratio
SDR	Signal-to-Distortion Ratio
cRM	complex Ratio Mask
cIRM	complex Ideal Ratio Mask
BPTT	BackPropagation Through Time
TBPTT	Trucated BackPropagation Through Time
T-F	Time-Frequency
VQA	Visual Question Answering
ReLU	Rectified Linear Unit
SELU	Scaled Exponential Linear Unit
ICS	Internal Covariate Shift

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

1.1 Context and Overview

1.1.1 Deep Neural Networks and Deep Social Transformations

Artificial Intelligence (AI) is a field that bring together a number of theories and technologies including conventional reasoning and rules-based methods. These technologies endow computers and automata to perform problem solving tasks in ways that mimic human thinking, at least in principle. Within the broader field we find a smaller domain known as machine learning (ML). The latter is a complete toolbox of important mathematical methods and algorithms allowing computers to improve their ability to perform many pattern recognition tasks such as computer vision and speech processing through learning. Lastly, Deep Learning (DL), constitutes a more specialized field within machine learning (Goodfellow et al., 2016). It should be noted that in this manuscript we use the terms “deep learning” and “deep neural networks” as equivalent synonyms unless otherwise indicated.

Recently, deep neural networks (DNNs) have made headlines in news for the general public. The significant progress and the unprecedented success of deep learning in various areas, such as natural language processing (NLP), image and speech recognition, and computer vision, constitute an important part of the “AI revolution”. Speech recognition advances have been made in leaps and bounds, as exemplified by products such as Amazon’s Alexa. Google has continued to train very deep neural networks to enhance the fluency and accuracy of its translation services. The increased ability of computers to identify images shows great promise to positively shape the near future of medical imaging diagnosis. Deep neural networks are expected to strengthen this growth in medical image analysis (X-rays, MRIs, and CT scans). Further, the health sector is expected to grow at a steady pace as DNN technology expands translational bioinformatics and sensor analysis. Enhanced image recognition is likely to be key to advances in robotics, autonomous drones and cars.

While AI is not new, AI methods based on deep learning have been making significant progress such that deep learning-based solutions are now widely used in industry. Fast improvements in information storage capacity, high computing power and parallelization (e.g. the preeminence of Graphics Processing Units (GPUs)) have contributed, to a large extent, to the swift uptake of deep learning technology in established industries such as search engines based around computational advertising as well as emerging industries such as autonomous vehicles. Businesses are now able to leverage the power of deep neural networks through using high level libraries and software packages. Software such as TensorFlow (Abadi et al., 2016) and Pytorch (Paszke et al., 2017) are now regularly applied to massive amounts of data allowing valuable insights to be extracted, to improve decision-making processes, to offer original products, and to model customer preferences and create better services. Deep learning technologies have attracted a significant amount of investment - nearly 60% of all non-industry investment in 2016 according to McKinsey (Chui, 2017).

Deep learning techniques have essentially made speech recognition into a usable technology. Consequently, it is now much more feasible to interact with computers by just talking to them and many companies now have products based around this interaction paradigm.

1.1.2 Real-Valued Deep Neural Networks

A popular primitive form of artificial neural network was first introduced more than half a century ago with Rosenblatt's Perceptron (Rosenblatt, 1958, 1961). However, the limitations of such simple models limited their interest in the research community. Neural networks regained interest within the research community, first in the mid-80s with the introduction of backpropagation by Rumelhart et al. (1986), followed by another moderate surge in interest in the second half of the 2000s with early advances in learning deeper models such as the contributions of Hinton et al. (2006), Bengio et al. (2007) and Poultney et al. (2007). A much more significant progress was achieved in the second decade of the 21st century. Deep Neural Networks or DNNs have represented one of the main vectors for research advances and growth in machine learning. Progress made by Krizhevsky et al. (2012), Goodfellow et al. (2014) and Karras et al. (2017) in computer vision, by Mnih et al. (2013) and Silver et al. (2017) in the area of reinforcement learning, and, by Bahdanau et al. (2014) and Vaswani et al. (2017) in the field of natural language processing, have made DNNs inescapable models for many learning tasks.

A real-valued neural network (RV-NN), or more specifically, a conventional (shallow) artificial neural network (ANN) is a set of nonlinear real-valued computing units (neurons) structured into a particular architecture to learn from experience and predict unseen events in terms of a hierarchy of concepts. It is a layered design with each layer composed of many neurons that make decisions based on input data. For shallow ANNs the number of layers or the (depth) is restricted to be small. Fully RV-NNs representations assume that the input data and the output, as well as the connection weights are real. The computations performed by the neural network are then based on real numbers. This is somewhat restrictive because there are learning tasks for which input and output are, by design, complex numbers. For such tasks, we wish here to use the full representational power of complex arithmetic, its richness and versatility. We are interested in incorporating fully complex computations into a DNNs' structure. In situations where the input and the output to a neural network are both complex values, we argue that fully complex computations are particularly appropriate, as will be described in later sections (see sections 1.4 and 3.2).

ANNs are loosely inspired by the neural networks of the human brain and the biological mechanisms that process information in the cortex. In the case of a computer, a task is performed by executing instructions contained in an algorithm written by a programmer. It turns out that the simplest tasks that humans can perform, and seem instinctive to them, such as the recognition of a person's face or the recognition of his voice, cannot indeed, be instructed to the computer in terms of basic and explicit algorithmic instructions. Neural networks can learn things using data alone, by mapping input features to outputs. There is no need for a human to program all the knowledge that the computer needs. The hierarchical structure with many layers allows the computer to acquire intricate patterns by building them out of simpler ones.

For the purpose of illustration, let's take the popular example from the literature where a neural network is assigned the task to learn how to recognize images of dogs. Initially, we input a mixture of images into the neural net as data points representing dogs and not dogs. By propagating the natural images in question from the input layer to the output layer, the first layers of the network detect fine-grained details of the image and the final ones detect more abstract structures. The deeper the information is processed through the network, the more general features are captured. But at this point, each neuron output is random (dog or not a dog). The network may or may not deliver the correct image. It is in fact no more than a simple game of tossing a coin (50% chance the network deliver the correct answer). In fact, the output is random because the network is initialized with random connection weights. But here we are describing approximately what will happen when we continue the feeding process. The weights are updated iteratively by using the well-known "backpropagation" algorithm that computes the error between the output guessed by the network and what the output should have been (dog or not a dog). The error is sent back through all layers of the system via this algorithm. The updating of the weights obtained by feeding the network with large numbers of images allows the system to improve over time at discerning between what an image of a dog is and what isn't. This type of learning is referred to as supervised learning.

Real-valued deep neural networks (RV-DNNs) use the composition of a large number of layers and nonlinear activation functions to capture the implicit nonlinear dependencies between input features and outputs. The difference between shallow RV-NNs and RV-DNNs is the depth of the nets i.e., the number of layers. Deep learning (DL) is an expression used for various types of neural networks, (to cite a few, e.g. recurrent nets (RNNs), convolutional networks (CNNs), and more recently, residual networks (He et al., 2016a) and Generative Adversarial Nets (Goodfellow et al., 2014, GANs)).

1.1.3 Building Blocks for Training Real-Valued Deep Neural Networks

Real-valued feed-forward neural networks (FFNNs) such as CNNs and real-valued RNNs have shown to excel in a wide variety of applications and learning tasks. However, their training is a challenging computational task. For instance, recurrent neural networks are characterized by a severe problem when training is carried out with gradient-based optimization algorithm, particularly when trying to capture long-term dependencies. This difficulty stems from the well-studied vanishing and exploding gradient problem (see section 5.2.3 for mathematical description and more details of the problem in the context of RNNs). Training very deep neural networks with stochastic gradient descent is hard because the gradients tend to decay as they are backpropagated through multiple levels of non linearity. Bengio et al. (1994) and Hochreiter (1991) report that the volatility of the gradients over a sequence of layers can prevent algorithm from convergence and impede learning. This problem is particularly intense for recurrent networks, since repeated applications of the recurrent weight matrix can magnify instability. Poor conditioning of recurrent matrices leads to exponential exploding or vanishing of gradients over the time horizon (see equation (5.4)). This problem prevents RNN from capturing long-term dependencies. Many heuristics have been initially devised to reduce some of the optimization difficulties. These include gradient clipping (Pascanu et al., 2013), initialization of the recurrence weight matrix by means of orthogonal matrices (Saxe et al., 2013; Le et al., 2015), use of long short-term memory (LSTM) recurrent networks (Hochreiter and Schmidhuber, 1997), and finally gated recurrent units (Cho et al., 2014, GRUs). Moreover, new architectures have been introduced to ensure smooth propagation of information through the network, and hence avoid the gradient dilemma. The most prominent are residual networks (Resnets), which directly transmit information from preceding layers up in a feed-forward network (FFNN) (He et al., 2016a). The attention mechanism (Bahdanau et al., 2014; Mnih et al., 2014), gives a recurrent network the possibility to have access to prior activations.

Beside the previous heuristics, a quite different approach relies on initialization (see section B) and/or normalization techniques. The latter introduce a particular type of transformations on the computations performed in the forward pass of the neural net. More precisely all normalization procedures for training RV-DNNs explicitly normalize some component (activations or weights) through dividing activations or weights by some real number computed from its statistics and/or subtracting some real number activation statistics (typically the mean) from the activations. For instance batch normalization uses the summed input to a neuron over a mini-batch of training observations to compute a sample mean and a sample variance which are then used to normalize the summed input to that neuron on each training case. Initializations (Glorot and Bengio, 2010; He et al., 2015) and normalization mechanisms, such as batch normalization (BN) (Ioffe and Szegedy, 2015), Layer normalization (LN) (Ba et al., 2016) and weight normalization (WN) (Salimans and Kingma, 2016) have been shown to empower training of very deep neural networks. They are widely understood to lessen training variability, support higher learning rate, speed-up convergence and enhance generalization, yet the causes for their successes are still the focus of an ongoing active research (Balduzzi et al., 2017; Santurkar et al., 2018; Kohler et al., 2018; Yang et al., 2019a). Initialization and normalization techniques constitute currently essential building blocks of the most successful RV-DNNs. For instance in addition to image classification (He et al., 2016a; Huang et al., 2017), a number of standardization techniques have excelled in achieving high performance on various tasks, such as object detection and segmentation (Wu and He, 2018) and fast stylization and style transfer (Ulyanov et al., 2016; Dumoulin et al.).

1.2 Complex-Valued Neural Networks

Complex-valued neural networks involve the use of complex valued inputs. A complex input may be characterized by its amplitude and phase which are one of the most fundamental concepts in signal processing (Hirose, 2012). CV-NNs process complex-valued information by means of complex-valued parameters and variables. They are particularly rich in diversity. CV-NNs are similar in architecture to their real-valued counterparts RV-NNs. The difference lies in the nature of their parametrization (weighting). The former comprise a set of non-linear computational units with complex-valued, instead of real-valued, weights and activation functions. A peculiarity of the CV-NNs that we leverage in the manuscript is that the analyticity property of their activation functions may be derogated (see section 3.6.2).

1.3 Problem Statement, Challenges and Objectives

From a research perspective, deep learning is still an active area where real-valued deep neural networks have proven to be powerful substitutes for conventional machine learning methods. They continue to set standards for a wide range of computer vision, automatic speech recognition and natural language processing problems. At present, the vast majority of building blocks, techniques, and architectures for training deep neural networks are founded on real-valued operations and representations. However, representations based on complex numbers have started to receive increased attention. Many articles have explored the implementation of complex numbers in deep learning architectures. Recent work on recurrent neural networks (Arjovsky et al., 2016; Wisdom et al., 2016; Danihelka et al., 2016) and older work (Plate, 1991, 1995) suggest that there may be “real” advantages of using complex instead of real arithmetic to parametrize deep neural networks. Despite their compelling properties, complex-valued deep neural networks have been somewhat neglected because of the absence of the building blocks required to design and train this type of nets. The lack of such a framework represents a noticeable gap in deep learning tooling.

This thesis presents a set of original contributions in the form of three essays on real and complex-valued deep neural networks. It addresses both theoretical and practical challenging issues related to the improvement of the training building blocks for both real and complex-valued deep neural networks including architectures, training methods (initialization, normalization, etc.) and learning of various tasks in computer vision, music transcription, speech spectrum prediction, signal retrieval and audio source separation. The three articles are dedicated to dealing with methods designed to provide solutions to the problems caused by the vanishing and exploding gradients during the training of deep neural networks.

1.3.1 Thesis Objectives and Connections Between Chapters Themes

The objectives of the thesis are threefold. First of all, we address the problem of exploding and vanishing gradient in deep complex and real-valued neural networks. This is achieved in both chapters 3 and 5 by investigating the utility of orthogonality for the hidden transition matrix in a Recurrent Neural Network (RNN) and leveraging the norm preservation property of unitary matrices to design initialization methods for deep complex-valued networks. The second objective is to investigate training stability issues that go beyond the vanishing and exploding gradient problem and that are specific to deep complex-valued neural networks. We do not limit our efforts to the investigation of such problems. In fact, we design appropriate building blocks that cope with the issues in question and allow us to design deep complex networks. Our third and final objective is to illustrate the utility of complex representation in tasks where both amplitude and phase components are relevant and highlight the effectiveness of deep complex networks to learn adequate hidden and output complex-valued representations for the task at hand. Therefore, we provide a solution in chapter 4 for the challenging problem of signal retrieval and signal extraction in the frequency domain. More specifically, we consider as a case study the task of monoaural audio source separation.

Below we provide an overview of each article objective, motivation, and contributions. Section

1.4 is devoted to a more thorough presentation of the specific contributions.

The first article aims to fill the gap identified hitherto by providing a new complex-valued deep neural network representation and training methods that go far beyond a simple theoretical generalization of the real-valued case. More precisely the key atomic components for complex-valued deep neural networks are developed and applied to convolutional feed-forward networks and convolutional LSTMs. Complex convolutions and new algorithms for complex batch-normalization and complex weight initialization strategies form the bedrock of the proposed framework. These are then used in intensive experiments with end-to-end training schemes.

The second article is an extension that builds on the results obtained in the first article to provide a novel fully complex-valued pipeline for automatic signal retrieval and separation in the frequency domain. We call our model a Deep Complex Separator. It is a novel masking-based method founded on a complex-valued version of Feature-wise Linear Modulation (FiLM). This new masking method is the keystone of our proposed speech separation architecture. We also propose a new explicitly phase-aware loss function, which is amplitude and shift-invariant, taking into account the complex-valued components of the spectrogram.

The third article is a study that discusses issues related to the nature of orthogonal transition hidden matrices employed to train recurrent neural networks. Unitary transition matrices and orthogonal constraints are shown to provide an appropriate solution to the vanishing and exploding gradients in recurrent neural networks. However, imposing hard constraints is shown to be computationally expensive and needless for some real-world tasks. This paper aims at assessing the utility of the orthogonality constraint by controlling the extent of the deviation of the corresponding transition weight matrix from the manifold of orthogonal transformations and measuring the generalization performance of the associated RNN. Hard constraint of orthogonality are found to limit the model's representational power, worsens its performance, and slows down optimization convergence speed. A weight matrix factorization and parameterization strategy is proposed to bound matrix norms and therein monitor the degrees of expansion and contraction of the activations norm during the forward propagation and the gradients norm during the backpropagation. We suspect that a hard constraint of orthogonality limits the model's representational power, worsens its performance, and slows down optimization convergence speed. By combining Singular Value Decomposition of the hidden transition matrix and the geodesic gradient descent algorithm, we can explicitly express the amount of deviation from orthogonality by setting a learnable margin for the singular values and optimize it. This allows us to observe and analyze cases where deviating from orthogonality could be either beneficial or detrimental.

1.4 Complex-Valued Neural Networks : Motivations and Recent Work

In this section, we first present the motivations and benefits of using a neural network formulation based on complex numbers rather than real numbers, then we briefly discuss recent articles content on deep complex networks. The suggested use of complex arithmetic in deep neural networks is justified by the many benefits it can provide. The latter are representational, computational and biological, as will be emphasized in the next subsections.

1.4.1 Representational Motivation

Complex arithmetic has long been the preferred tool for digital signal processing (Hirose and Yoshida, 2012). The complex formulation is beneficial not only for wave-originating information, but also for a broad-spectrum of processing with time-frequency treatment using Short Time Fourier transform (STFT). A variety of other transformations can be applied to convert real-valued input to complex-valued input. For instance work on the theory of scattering networks (Bruna and Mallat, 2013), and subsequent works, have also provided sound mathematical grounds for understanding deep learning neural networks on the basis of complex wavelet settings and non-linear operators. A complex signal does not merely increase the size of the signal (real + imaginary). Instead, the depictive richness of complex signals is embedded in its practical interpretability and rigorous mathematical theory of complex analysis. A CV-NN is not similar to a two dimensional real-valued neural network as it may be thought at first glance. It has its own subtleties and appropriate features such as generalization (Hirose and Yoshida, 2012). CV-NNs are shown to be valuable in processing amplitude information and signal related phenomena. This is a critical point in applications in various domains.

Further, there are learning tasks where the use of CV-NNs is inevitably required or greatly effective. Inputs to a network may be naturally complex-valued (i.e. complex by design, because they have phase and magnitude components that are statistically correlated), or converted to a complex form. Complex-valued signals occur in many areas and are thus of fundamental interest. Examples of such inputs include, Magnetic Reasoning Images (MRI), Synthetic Aperture Radar Data (SAR) and Very-Long-Baseline Interferometry (VLBI), Antenna Design (AD), Radar Imaging (RI), Acoustic Signal Processing (ASP) and Ultrasonic Imaging (UI), Communications Signal Processing (CSP), Traffic and Power Systems (TPS) etc (Hirose and Yoshida, 2012).

It is worth noting that, even in the analysis of real-valued inputs (e.g. electrocardiogram (ECG), electroencephalogram (EEG), images or audio signals), one of the most accurate approaches is to use frequency domain analysis, which directly involves complex numbers. In general the frequency domain representation of a real-valued input is obtained using the short time Fourier transformation which is a mapping from the set of real numbers to the set of complex numbers. From a representational perspective, this is advantageous as it provides a fine-grained 2D representation of a 1D signal without any loss of information. Moreover, signals such as ECG, EEG and audio are seen as a superposition of different oscillations. The representation in terms of frequency components seems then to be another natural way of representing such inputs. By considering as input a Fourier-based encoding, a prior knowledge about the nature of the signal and its structure is passed to the neural network. The transmitted input signal is described by its magnitude and phase which carry different information contents about the nature of that signal. Several studies suggest that phase is important for perceptual quality, leading to consider magnitude and phase. Phase has proved to be critical in image processing. It is much more informative and important than magnitude for image understanding and interpretation, and for image recognition (Oppenheim and Lim, 1981). In the Fourier representation of an input, spectral magnitude and phase tend to behave differently, and in some circumstances most of the main constituents of the input signal are well-preserved if the phase alone is retained. In addition, under different settings, for example when a signal is of finite length, the phase content alone is sufficient to fully reproduce a signal within a scale factor (Oppenheim and Lim, 1981; Paliwal et al., 2011; Williamson et al., 2016).

Dating back to the 1980s literature (Oppenheim and Lim, 1981), it has been shown that complex phase is useful for capturing non-stationary or quasi-stationary (locally-stationary) behavior, discontinuity (edges and jumps) and oscillatory movements (textures). Oppenheim and Lim (1981) have shown that the information related to all edges, shapes and all entities in an image is completely embedded in phase. Paliwal and Alsteris (2005) have shown that the phase spectrum of a speech signal can contribute to speech intelligibility as much as the magnitude spectrum. A complex-valued feature, taken as a whole figure, is more appropriate and efficient at capturing local behavior than a pair of real and imaginary coordinates, calculated independently, and, integrating invariances with respect to translation or rotation. Complex-valued neural nets take into account the existing correlation between the real and the imaginary part and learn representations for the task at hand accordingly.

1.4.2 Computational Motivation

The previous discussion highlights the representational power of complex arithmetic, its richness and versatility which can be incorporated into deep neural networks' structures. Other benefits and motivating proprieties relate to computational aspects as we report below.

The Short Time Fourier transform (STFT) time-frequency encoding provides important added advantages : 1- the reduction of time dimension of the representation for an underlying signal, and 2- computational efficiency. For instance, training Recurrent Neural Networks (RNNs) or Convolutional Neural Networks (CNNs) on long sequences is known to be a challenging problem due, not only to the unstable vanishing and exploding gradient problems, but also to the computational requirements of backpropagation through time (BPTT) (Hochreiter, 1991; Bengio et al., 1994). The use of STFT embedding can considerably reduce the temporal dimension of the representation. Besides, STFT is computationally efficient as, in practice, it uses the Fast Fourier Transform (FFT) whose computational complexity is $\mathcal{O}(N \log(N))$.

Many input and output signals are modeled as linear time-invariant or shift-invariant (LTI and LSI) systems (e.g. Audio, speech, ...). These signals can be represented equally in the time domain or in the frequency domain. A signal $\mathbf{s}$ is the result of convolving an input $\mathbf{x}$ with a system's transfer function $\mathbf{g}$ such that $\mathbf{x} * \mathbf{g} = \mathbf{s}$, where $*$ denotes the convolution operation. According to the convolution theorem, the Fourier Transform of convolved signals in the time domain is the elementwise multiplication of their Fourier Transforms and vice versa (see section A). This means that $\mathbf{F}(\mathbf{s}) = \mathbf{F}(\mathbf{x} * \mathbf{g}) = \mathbf{F}(\mathbf{x}) \odot \mathbf{F}(\mathbf{g})$, where $\mathbf{F}$ denotes the Fourier Transform and $\odot$ denotes the elementwise complex multiplication. Now if we need to recover $\mathbf{x}$, deconvolution in the frequency domain is usually advised by computing $\mathbf{F}(\mathbf{s})$. In the case of absence of noise it is merely $\mathbf{F}(\mathbf{x}) = \mathbf{F}(\mathbf{s}) \, / \, \mathbf{F}(\mathbf{g})$. Finally the inverse Fourier Transform $\mathbf{F}^{-1}$ is applied to $\mathbf{F}(\mathbf{x})$ to estimate the deconvolved signal $\mathbf{x}$.

1.4.3 Biological Motivation

Neuroscience provides another source of motivation for complex computations. Reichert and Serre (2013) argue that neural network formulations relying on complex-valued neuronal units are not only computationally efficient but also biologically meaningful and more plausible than their real valued counterpart. The authors explain that in the human brain an action potential, taking place in the cortex, is characterized by distinct pulse (spike) patterns, and the time separating pulses may be different. This suggests that it is appropriate to introduce complex numbers representing phase and amplitude into artificial deep neural networks. In such representations units (neurons) in each layer are described by a pair of attributes, a firing rate and a phase. The phase embodies several characteristics, believed to be linked to neuronal synchrony. These include gating of information processing and binding of distributed object representations. Focusing on the latter, Reichert and Serre (2013) show that neural synchrony could be flexible enough to ensure multiple roles in deep neural networks. For instance the role of temporal coordination of a neuronal output or more specifically synchronization of neuronal firing is a relevant computational element of how the gating mechanism can be exploited in propagating information through deep networks.

Synchrony has been postulated in various studies as a key element in the processing of sensory information by the cortex (Von Der Malsburg, 1994; Crick, 1984; Singer and Gray, 1995; Fries, 2005; Uhlhaas et al., 2009; Stanley, 2013). The order of magnitude of synchrony of neuronal spikes has been shown to have an impact on the output of downstream neurons. For this reason, synchrony is assumed to play an important role in the gating of information transmission between neurons (Fries, 2005; Benchenane et al., 2011). It is important to mention that this is related to the gating mechanism existing in deep feed-forward neural networks (Srivastava et al., 2015; Greff et al., 2016), deep autoregressive models (van den Oord et al., 2016a,b) and also recurrent neural networks (Hochreiter and Schmidhuber, 1997; Cho et al., 2014). In the context of deep gating-based networks, synchronization means the propagation of information whose controlling gates simultaneously hold high values. These controlling gates are usually the activations of a sigmoid function. In fact, in the case of a deep feed-forward network, the gating mechanism allows one to propagate inputs of a given layer to the subsequent ones. In the case of a recurrent neural network, it allows the holding of information from one time step to another.

There are other advantages coming from computational neuroscience. For instance in their description of the processing of sound by the human hearing system, Terhardt (1985) and Zwicker and Fastl (2013) explain that the auditory system decomposes sound according to its frequency bands, with the cochlea playing a role similar to that of the Fourier transform in the frequency decomposition task, apart from the fact that computations are carried out on a logarithmic basis (Chalupper and Fastl, 2000; McDermott, 2018). We conjecture that when a deep neural network is fed with a complex input, it may learn the frequency-based structures and features that are most informative to the task being learned.

1.4.4 Recent Related Work on Complex-Valued Representations in Recurrent Neural Networks

As we pointed out earlier, for the majority of published work, it is common to shorten complex-valued spectra by omitting the phase components of frequency domain signals, which results in important losses of information. Complex-valued spectra have been often disregarded as the deep learning community has not developed reliable methods for training complex deep neural networks. Recently, a quite few articles have explored the implementation of complex numbers in deep learning architectures. The tardy implementation of deep complex networks could be explained by the followings facts : 1) The lack of work investigating the difficulty of training deep complex DNNs, and 2) The lack of open libraries allowing the training of such models and supporting complex numbers. In this section, we present the few closely related studies that have focused on the particular case of recurrent neural networks using complex arithmetic.

Arjovsky et al. (2016) introduced the idea of applying a complex-valued unitary recurrent weight matrix to guard against the vanishing and exploding gradients. The advised unitary recurrent neural network (uRNN) is a complex-valued RNN, with a unitary recurrent weight matrix and complex-valued ReLU-based activation function called modReLU. A unitary recurrent weight matrix is a norm preserving mapping. The use of such unitary mapping prevents the gradients from exploding or vanishing, and allows for the capture of long-term dependencies. The proposed uRNN is computationally efficient as it spans a reduced subset of unitary matrices. The authors provide empirical evidence suggesting that uRNNs perform well at propagating gradient information over lengthy sequences and are not affected by saturating hidden states as much as LSTMs. Unfortunately, there is no free lunch. Restricting uRNNs to a reduced subset of unitary matrices decreases their expressive power (Wisdom et al., 2016).

To overcome the problem Wisdom et al. (2016) proposed to parameterize the recurrent matrix with a full dimensional unitary matrix at the expense of computational efficiency. The training of full capacity unitary recurrent networks (FC-uRNNs) requires a very greedy projection step in terms of computation time to maintain the unitary propriety of the recurrent matrix at each iteration of the stochastic optimization process. The FC-uRNNs outperform by a large margin, their restrictive analogous uRNNs that use a restricted-capacity recurrence matrix.

Jing et al. (2016) proposed a new parameterization of the recurrent weight matrices using unitary linear operators, called Efficient Unitary RNN (EURNN). EURNNs are supposed to allow a rather fine control of the number of parameters of a recurrent neural network and thereby overcome the problems of excessive parameterization and bad conditioning. Although the idea of setting up recurrent weight matrices with a strict unitary matrix operator is appealing, it raises a number of delicate issues. First, imposing strict unitary constraints greatly limits the search space. In addition, strict unitary constraints make it difficult to forget irrelevant information.

Jose et al. (2017) address two important concerns with recurrent neural networks : over-parametrization, and ill-conditioning of the recurrent weight matrix. The first problem increases the complexity of learning and the training time. The second leads to the vanishing and exploding gradient problem. The authors propose a flexible recurrent neural network architecture referred to as Kronecker Recurrent Units (KRU) which improves parameter efficiency in RNNs through a Kronecker factored recurrent matrix. It also circumvent the ill-conditioning of the recurrent matrix by imposing soft unitary restrictions on the factors. Jose et al. (2017) report that the KRU approach outperform classical methods which uses $\mathcal{O}(N^2)$ parameters in the recurrent matrix. Yet, their method still suffers from balance between computational efficiency and sample complexity.

Mhammedi et al. (2016) in their orthogonal RNN (oRNN) bypass the costly projection step in FC-uRNN by parametrizing the orthogonal matrices using Householder reflection vectors. The latter allows a fine-grained control over the number of parameters by choosing the number of Householder reflection vectors. However their continuous real-valued parametrization fails to fulfil the necessary condition of the continuity of unitary space to ensure an adjustable continuous parametrization going from subspace to full unitary space.

Danihelka et al. (2016) highlighted two main limitations related to computational aspects of traditional LSTMs. First, the number of cells depends on the dimension of the recurrent weight matrices, which implies that an LSTM with N_h memory cells uses a recurrent weight matrix with $\mathcal{O}((N_h)^2)$ entries. In addition, LSTM is limited in learning to represent common data structures such as arrays due to the lack of a mechanism for indexing its memory during writing and reading. Combining Long Short-Term Memory networks and Holographic Reduced Representations (HRRs) (Plate, 1991, 1995), Danihelka et al. (2016) introduced a new method to enhance recurrent neural networks capacity with additional memory without increasing the number of network parameters. The proposed combined scheme has an associative memory array founded on complex-valued vectors. The authors report experimental evidence of faster learning on numerous memorization tasks.

Jing et al. (2019) have built a novel RNN, named Gated Orthogonal Recurrent Units (GORU), that conveys an explicit forgetting mechanism to the class of unitary/orthogonal RNNs. GORU combines the remembering ability of orthogonal matrices (uRNNs) and the ability of gated architectures to efficiently omit recurring or irrelevant past information. The authors argue that their robust forgetting mechanism outperforms short and long-run neural networks such as gated recurrent unit GRU and traditional LSTM in several benchmark tasks. In spite of their robust forgetting capabilities, GORU still suffer from perfect balance between computational efficiency and sample complexity. Jing et al. (2019) also attempted to use complex-valued representation by having unitary weights instead of orthogonal ones. The authors report that they encountered training instabilities while working with complex-valued representations, hence, they decided to consider only the real-valued case.

More recently, Wolter and Yao (2018a) built on the previous works related to the parametrization of the hidden transition matrix with a unitary transformation. They present a complex-valued gated recurrent network (cgRNN) in which weights are parameterized by full-dimensional unitary matrices. The cgRNN is a hybrid cell resulting from the combination of complex-valued norm-preserving state transitions with a gating mechanism.

To summarize, we can conclude from the previous discussion that there are indeed many “real” benefits of using complex-valued representation. The reviewed research papers indicate smoother flowing and more robust transmittal of gradient information across layers when unitary transition matrices are used. Higher memory capacity and noise-robust memory retrieval mechanisms are reported in the context of associative memory. A significant reduction in the number of parameters and a faster convergence are also reported when complex-valued models are used. These are too many advantages that cannot be simply overlooked. If we are willing to admit that the current Deep learning constitutes an open opportunity for further research contributions, then perhaps we should make use of complex analysis where there is a greater variety of recipes to leverage.

1.4.5 Contributions

This manuscript provides several contributions that we describe for each article separately, with the mention that our approach is the first, in its kind, to provide a general framework for building blocks that can be used in training complex-valued deep neural networks. Our finding are impactful since they have been the subject of further exploration, adoption and extensions in recent works (in sections 6.1.1 and 6.1.3).

Contributions in Article 1

Article 1 is the object of the developments detailed in Chapter 3. By leveraging the advantages offered by complex representations, we propose a general formulation for the building components of complex-valued deep neural networks and apply it to the context of feed-forward convolutional networks and convolutional LSTMs. Our contributions in this article are :

1. A specification of a new complex valued activation function and a comparison of its performance to that of various complex-based activation functions. See sections 3.3.4 and 3.4.1 ;
2. A formulation of complex batch normalization, which is described in Section 3.3.5 ;
3. Complex weight initialization, which is presented in Section 3.3.6 ;

4. A state-of-the-art result on the MusicNet multi-instrument music transcription dataset, presented in Section 3.4.2 ;
5. A state-of-the-art result in the Speech Spectrum Prediction task on the TIMIT dataset, presented in Section 3.4.3.

Contributions in Article 2

Article 2 is the subject of the developments included in chapter 4. It builds on the concepts depicted in the first article to provide a novel fully complex-valued pipeline for automatic signal retrieval and signal separation in the frequency domain. As a case study, we consider speech separation. In audio source speech separation, the task consists of isolating the speech of multiple speakers into separate signals. Our contributions are summarized as follows :

1. A novel masking method presented in section 4.3.3. It is based on Feature-wise Linear Modulation (FiLM) (Perez et al., 2018) to create multiple separated candidates for the audio associated with a given speaker. These candidates are averaged in order to obtain the final separated speech for the speaker in question. Our experiments demonstrate the efficacy of our proposed masking method, and show its regularizing effect ;
2. A new frequency-domain loss presented in section 4.4. The loss function is taking explicitly into account both magnitude and phase of the signals. The characteristic of our loss is that it is amplitude and shift invariant. Our comparative analysis relates to different phase-aware losses defined in time and frequency domains, demonstrates the superiority of our proposed loss.

Contributions in Article 3

One method to prevent gradients from vanishing and exploding is to employ either soft or hard constraints on the transition weight matrices in order to enforce orthogonality. Orthogonal matrices have the compelling propriety of preserving gradient norm during backpropagation and therefore are tractable as they provide potential solutions for the vanishing and exploding gradient problem. Here our paper explores issues related to optimization convergence, speed and gradient stability when encouraging or enforcing orthogonality. The main contributions are :

1. A weight matrix factorization and parameterization strategy is proposed. The latter is employed to bound matrix norms and therein control the degrees of expansion and contraction of the activations norm during the forward propagation and the gradients

norm during the backpropagation ;

2. A sensitivity analysis to identify the cases where deviating from orthogonality could be either beneficial or detrimental is proposed. The results of the analysis suggest that a strict orthogonality or an uncontrolled deviation from the orthogonal space of parameters is generally detrimental to the training process and that a compromise of the amount of deviation must be considered in order to ensure a faster and more efficient convergence and a good performance of the model.

1.5 Outline

The chapters of this manuscript are based on published works reported in Trabelsi et al. (2017), Vorontsov et al. (2017) and a work under review presented in chapter 4 introducing “Deep Complex Separators”. Chapter 2 is a review of closely related literature with more focus on recent works. In this chapter we first present a detailed review of the main approaches related to training difficulties observed when training real-valued deep neural networks. Our review includes methods for parameter initialization and normalization of activations, gating-based mechanisms and heuristics designed to deal with the problem of vanishing and exploding gradients in deep feed-forward neural networks. In a second stage we review recent work on complex-valued representation in deep neural networks. Finally we discuss studies on signal retrieval and speech separation. Chapter 3 is the object of the article entitled “Deep Complex Networks”. It contains the technical details and the experimentation related to the proposed complex-valued “building blocks” for training deep neural networks. It includes new initialization and normalization methods for training various tasks with real and naturally complex-valued inputs. Chapter 4 is the object of the novel theoretical and empirical developments for signal retrieval and speech separation proposed in the article entitled “Deep Complex Separators”. Chapter 5 is devoted to the “Study of Orthogonality for the Hidden Transition Matrix in Recurrent Neural Networks”. Chapter 6 concludes the thesis and opens up new research perspectives that can be explored in the future.

Basic concepts related to complex numbers and signal processing are presented in Appendix A. In this thesis, we make use of two initialization techniques, widely used in deep learning research, the Glorot and He initializations. Given the lack of technical details, found in subsequent research work related to these methods, we have found it useful that we include, in Appendix B, an in-depth mathematical derivation of the missing details. Notice that in this thesis we build on these initializations to introduce new ones for the complex-valued neural network architectures that we propose in the next chapters.

CHAPTER 2 LITERATURE REVIEW

Deep Learning techniques have recently become standard tools for achieving state-of-the art results in machine learning and pattern recognition tasks. This is the result of successive research work that has shown the potential of deep learning models. A number of contributions in the field have successfully dealt with the problem of the vanishing and exploding gradient through the design of appropriate “building blocks” including architectures, techniques and learning strategies.

In this chapter we first present a detailed review of the main building blocks used in real-valued deep neural networks. Our discussion includes methods for parameter initialization and normalization of activations, gating-based mechanisms and heuristics designed to deal with the aforementioned problem in feed-forward and recurrent neural networks. In a second stage we provide a summary on recent work dealing with complex-valued deep neural networks.

2.1 Initialization Techniques

The goal of a parameter initialization technique is to ensure that during the first iterations of training that the input signal can propagate to the output layer of the network. It also ensures the return of a correction feed-back signal from the output layer to the input layer. The correction feed-back signal is also called the error signal. It consists of the gradient of the cost function with respect to the parameter being updated. The propagation of the input signal as well as the back-propagation of the correction term has to be performed through the layers of the network without any severe vanishing or exploding of the signals amplitude. The control of the signals amplitudes can be achieved by bounding their respective variances.

The two major works on initialization techniques are those of Glorot and Bengio (2010) and He et al. (2015). Glorot and Bengio (2010) have investigated parameter initialization for neural networks using tanh and sigmoid activation functions, while He et al. (2015) have used linear rectifiers. He et al. (2015) present a more robust initialization method than Glorot and Bengio (2010) as they make more appropriate assumptions about the distribution of the activations and their derivatives by taking into account the sparsity induced by the use of rectifiers and their linear behavior, except at the origin. This comes from the simple fact that in the case of the rectifiers null activations have null derivatives and positive activations have derivatives equal to 1. Therefore for rectifiers a distribution assumption can be easily satisfied at initialization time for both the activations and their derivatives. With regard to the derivatives of tanh and sigmoid this is not the case. Glorot assumes that the derivative is always equal to 1 at initialization time, regardless of the activation value. This is obviously incorrect as the maximal values of the tanh and sigmoid derivatives are obtained for null activations and correspond respectively to 1 and $1/4$.

It is worth mentioning that Glorot et al. (2011) introduced the use of rectifiers for supervised learning which was an essential step in deep learning research. The findings in both Glorot and Bengio (2010) and Glorot et al. (2011) have led to the discarding of unsupervised pre-training techniques as a means to perform weight initialization.

2.2 Normalization Techniques

Besides parameter initialization, normalization techniques have been widely adopted to enable faster convergence and stable training. The first approach that has been introduced to normalize neural representations is Batch Normalization (Ioffe and Szegedy, 2015). Batch Normalization is also the most commonly used method among normalizing techniques in the deep learning literature. It is especially adopted for vision tasks and non-sequential data.

Normalizing the input representation of a machine learning model is not a recent practice. The min-max normalization is probably the simplest normalization technique that use to be widely adopted in the machine learning community. It consists of rescaling the features of the input either in the $[0, 1]$ or the $[-1, 1]$ intervals (Juszczak et al., 2002). The standardization of the input features is another normalization technique consisting of subtracting the sample mean and the dividing by the sample standard deviation so that the transformed data has zero mean and unit variance. Whitening transformations such as Principal Component Analysis whitening (PCA whitening), Mahalanobis or ZCA whitening and Cholesky whitening of the input features, are different methods consisting of linearly decorrelating the input components. Whitening is still a common practice in machine learning (LeCun et al., 2012). It is a generalization of standardization techniques. If the features are normally distributed, performing a whitening transformation on the input features allows one to feed the model with independent factors that have the same sample mean and the same unit variance. In other words, given that the features are normally distributed, a whitening transformation maps the input space to a space where the features are independently and identically distributed.

Batch Normalization (BN) is applied at each neural layer instead of just performing normalization on the input features. The first transformation that Batch Normalization performs is the standardization of the previous layer features. During the training process, the standardization is performed using the corresponding sample mean and sample standard deviation of each feature. These estimates are computed across a minibatch of instances. This dependence on the minibatch is not desirable during inference time when using held-out data. It yields a stochastic behaviour of the output while the latter should depend on the input deterministically. It is possible then to estimate the mean and the standard deviation using the whole training set instead of just a random minibatch. This is expensive especially if one wants to track the validation loss during training. A more efficient alternative is to update the mean and standard deviation estimates at each training iteration using exponentially weighted moving averages of the estimates. Libraries such as Pytorch (Paszke et al., 2017) and Tensorflow (Abadi et al., 2016) implement the exponentially weighted moving average method to estimate the mean and the standard deviation of the features.

After performing standardization, an affine transformation is performed on the features using a scaling and a shifting parameter. These features scaling and shifting parameters are learned by backpropagation. The final formula for BN can then be expressed as :

$$x_{bn} = \gamma \, \frac{x - \hat{\mu_x}}{\hat{\sigma_x}} + \beta, \tag{2.1}$$

where $\hat{\mu_x}$ and $\hat{\sigma_x}$ are respectively the estimates of the mean and the standard deviation of the feature x. γ is the scaling parameter and β is the shifting one.

The standardization $\frac{x-\hat{\mu_x}}{\hat{\sigma_x}}$ implemented in Batch Normalization ensures the non-explosion and non-vanishing of both the sample mean and sample variance of the activations and their feed-back gradient signal. Therefore, it achieves the goal for which initialization methods have been introduced, which is avoiding vanishing and exploding variances of the activations and their corresponding gradient through the layers of the network. This allows to maintain stable training not only in the early stages of learning but through the whole training process.

Numerous other normalization techniques have been proposed in the deep learning setting. Among those, Layer Normalization (Ba et al., 2016, LN) and Weight Normalization (Salimans and Kingma, 2016, WN) are probably the most common methods that are used as alternatives to Batch Normalization. Layer Normalization consists of performing the exact same standardization, scaling and shifting operations as the ones implemented in Batch Normalization. The only difference is that the sample mean and the sample standard deviation are computed over the features of the layer instead of the instances of the minibatch. This provides a flexibility compared to the Batch version as the normalization becomes independent of the minibatch size. Layer Normalization is mostly used in the context of sequential data such as natural language texts, speech signals, time series data, etc. These temporal and sequential data are mostly processed by autoregressive models taking into account temporal dependencies. Examples of such models are Recurrent Neural Networks. It has been shown empirically that Layer Normalization is more effective than Batch Normalization on sequential data (Ba et al., 2016).

Weight Normalization (Salimans and Kingma, 2016) is another technique which consists of rescaling the norm of the weights at each layer and where the rescaling factor is learned. This allows to avoid the co-adaptation of the weights with the norm of the activation during learning by decoupling the norm of the weights from their direction. Like Layer Normalization, the weight version is independent from the minibatch size. It is even independent of any data statistics. The formula for WN is given by :

$$\mathbf{w}_{wn} = \gamma \, \frac{\mathbf{w}}{||\mathbf{w}||}, \tag{2.2}$$

where $||\mathbf{w}||$ is the euclidean norm of the weight vector $\mathbf{w}$ and γ is a scalar expressing the norm of $\mathbf{w}_{wn}$.

Self-Normalized Neural Networks (Klambauer et al., 2017) implement the Scaled Exponential Linear Unit (SELU) activation which implicitly performs normalization as it ensures that the activations at each layer converge towards a zero mean and a unit variance. Klambauer et al. (2017) proved that the variance is also bounded which eliminates the risk of vanishing and exploding gradient. The authors have also observed that the normalization induced by SELUs provides a stable training and a strong regularization effect.

The common observations made when normalization techniques have been used are related to their positive impact on the training process as they provide faster convergence, better performance and a stable training. The authors in Ioffe and Szegedy (2015) speculate that the preservation of the activations mean and variance in Batch Normalization reduces what is called Internal Covariate Shift (ICS) and so, accelerates training. ICS is the change of the layer's features distribution through the depth of the network. The authors explanation lacks theoretical clarity. It is true that, if the joint distribution of the neural units at each layer is Gaussian, then, preserving the mean and the covariance of the joint distribution of the features at each layer would result in its preservation. However, not only the distribution of the features is not necessarily Gaussian but the covariance between the distinct features is also not taken into account in Batch Normalization. The preservation of the features joint distribution is then not guaranteed when using Batch Normalization. Recently, the machine learning community has started to uncover more fundamental problems that batch normalization is helping to solve. Santurkar et al. (2018) revealed that, under some assumptions, Batch Normalization makes the loss landscape smooth which results in a more predictive and stable gradients of the loss with respect to the features. This allows the neural network to converge faster towards an optimal solution. Under the Gaussian and zero mean assumptions of the data, Kohler et al. (2018) have shown that Batch Normalization splits the optimization problem in the same way as Weight Normalization by optimizing the norm and the direction of the parameters separately.

2.3 Gating-Based Mechanism and Identity Skip Connections

In addition to the normalization and initialization techniques that have made it possible to design and train deep networks, gating-based architectures have made it easier to train recurrent neural networks, as well as to design new feed-forward networks that can reach more than 100 layers.

Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (Cho et al., 2014, GRU) are recurrent networks that use the gating-based mechanism. The gating-based mechanism implemented in LSTMs and GRUs makes it possible to take into account temporal dependencies between observations which are separated by short or long time intervals. The gating-based mechanism allows them to reduce the problem of vanishing and exploding gradient.

Highway networks (Srivastava et al., 2015; Greff et al., 2016) are feed-forward networks that implement the same gating-based mechanism. To the best of our knowledge, the design of highway networks allowed, for the first time, to build very deep feed-forward networks (more than 100 layers). It is important to mention that the highway networks do not use any kind of normalization layers. This is due to the fact that the gating-based mechanism, by itself, implements a kind of normalization that limits the values of the activations between -1 and 1. The intermediate output $f(\mathbf{x}^l)$ is multiplied by a transformation gate $T(\mathbf{x}^l)$ and the result is added to the multiplication of the input by another carry gate $C(\mathbf{x}^l)$ as described by the following equation :

$$\mathbf{x}^{l+1} = T(\mathbf{x}^l) \cdot f(\mathbf{x}^l) + C(\mathbf{x}^l) \cdot \mathbf{x}^l. \tag{2.3}$$

$T(\mathbf{x}^l)$ and $C(\mathbf{x}^l)$ are sigmoidal gates that use the sigmoid activation function. $f(\mathbf{x}^l)$ uses the tanh activation function. Srivastava et al. (2015) have noticed that when the carry gate is expressed in terms the transformation gate such that $C(\mathbf{x}^l) = 1 - T(\mathbf{x}^l)$, it reduces the number of parameters and it makes sure that the sum of the contributions of the input $\mathbf{x}^l$ and of the intermediate output $f(\mathbf{x}^l)$ to the activations is equal to 1. This is an implicit standardization implemented at each layer of the highway network.

Inspired by the highway networks, He et al. (2016a) developed the residual networks by taking advantage of the combination between batch normalization and the ReLU activation function. They have also changed the gating mechanism existing in the highway networks to create an architecture that implements a specific case of highway layers. This specific mechanism consists of computing an intermediate output $f(\mathbf{x}^l)$ and adding to it the input $\mathbf{x}^l$. This is equivalent to considering $T(\mathbf{x}^l) = \mathbf{1}$ and $C(\mathbf{x}^l) = \mathbf{1}$ for all layers l. The intermediate output $f(\mathbf{x}^l)$ is said to be connected to the input $\mathbf{x}^l$ by an identity skip connection, or shortcut, since $\mathbf{x}^l$ is multiplied by $C(\mathbf{x}^l) = \mathbf{1}$ and it skips the convolutional layers inside the residual modules that compute $f(\mathbf{x}^l)$. As the intermediate output $f(\mathbf{x}^l)$ could be expressed as $f(\mathbf{x}^l) = f(\mathbf{x}^{l+1}) - \mathbf{x}^l$, it could be interpreted as a residual, hence, the name of residual networks. Stacking the residual modules makes it possible to build very deep networks.

The combination of identity skip connection, batch normalization and ReLU activations allows for a better flow of the activations and of their gradients through the layers of the network. For highway networks, a trade-off in terms of the number of layers must be established to obtain an optimal performance of the model. Whereas, for the residual networks, the increase in the number of layers is generally beneficial.

Balduzzi et al. (2017) explained that the combination of identity skip connections, batch normalization and ReLU activations ensures that the gradients of the loss with respect to the units of the first layers are correlated. This is only observed for the residual networks and for very shallow networks of just one layer. For deep feed-forwards networks without batch normalization and identity skip connections, the gradients of the loss with respect to the first layer units are decorrelated and resemble white noise, hence the shattered gradient problem. The gradients correlation in residual networks decays sublinearly through the layers while in standard deep feed-forward networks, it decays exponentially.

2.4 Vanishing and exploding gradient problem in Recurrent Neural Networks

As we have seen in section B, care must be taken to ensure a forward and a backward propagation of the layers' activations and their gradients respectively. The vanishing and exploding gradient problem was discovered by Hochreiter (1991) and Bengio et al. (1994) separately. Both recurrent and deep feed-forward neural networks suffer from this shortcoming. Numerous attempts have been proposed in order to reduce the severity of this problem. One of the most notable attempts is the Long Short Term Memory (LSTM) Network (Hochreiter and Schmidhuber, 1997) where a memory cell is added to the architecture of a recurrent network. The memory cell is updated or completely reset at each time step. This allows the recurrent network mechanism to remember long-term dependency information if needed, as well as to forget them and retain only short term ones when needed. The feed-forward analogue of the LSTM is the highway network (Srivastava et al., 2015) which inspired He et al. (2016a) later to incorporate identity skip connections in deep feed-forward networks.

Pascanu et al. (2013) suggested the gradient norm clipping strategy to deal with exploding gradients which turned out to be extremely efficient in practice. This has the effect of bounding the spectral norm of the hidden transition matrix, i.e., bounding its maximal singular value, and so, limiting the maximal gain induced by the hidden transition matrix.

Other methods have been deployed in order to overcome the vanishing and the exploding gradient problem. Krueger and Memisevic (2015) applied a penalty on the differences between successive L_p norms of the hidden representation. This has the effect of penalizing vanishing or exploding gradients and encouraging norm-preserved representations through time. Orthogonal weight matrices provide a novel angle of attack on the well-known vanishing and exploding gradient problem in RNNs. Le et al. (2015) adopted an identity initialization for the hidden transition matrix which is a specific case of an orthogonal initialization. To the best of our knowledge Saxe et al. (2013) were the first to use an orthogonal initialization for the weights of a deep feed-forward network. Henaff et al. (2016) used a soft penalty constraint on the hidden transition matrix, of the form $\lambda||\mathbf{W}^T\mathbf{W}-\mathbf{I}||^2$ to encourage the weights to be close to the Stiefel Manifold, the space of orthogonal parameters. When this is combined with an L_2 penalty on the hidden transition matrix, it has the effect of applying a Gaussian prior with a mean equal to 1 on the singular values of the weights.

Arjovsky et al. (2016) designed a recurrent neural network where the transition hidden matrix is unitary. Unitary matrices are the complex generalization of orthogonal matrices. Compared to their orthogonal counterparts, unitary matrices provide a richer representation, for instance being capable of implementing the discrete Fourier transform, and thus of discovering spectral representations. In Arjovsky et al. (2016) the hidden transition matrix is constructed using the product of specific unitary transformations, more specifically using diagonal matrices, permutations, rotations, the Discrete Fourier Transform and its inverse. As a consequence, the resulting hidden transition matrix, does not cover the whole space of possible unitary matrices in $\mathbb{C}^{N_h \times N_h}$ where N_h is the number of complex hidden dimensions. Wisdom et al. (2016) have proposed to cover the set of all unitary matrices for the hidden transition matrix by adopting the geodesic gradient descent algorithm (Nishimori, 2005; Tagare, 2011) when optimizing the unitary weights. The geodesic gradient descent is expensive in terms of computational complexity as it is cubic, i.e $\mathcal{O}((N_h)^3)$. This is because the algorithm requires an inverse operation. Hyland and Rätsch (2017); Mhammedi et al. (2016); Lezcano-Casado and Martínez-Rubio (2019) have proposed the design of full capacity unitary parameterizations for the hidden transition matrix that are efficient in terms of complexity. Arjovsky et al. (2016) show the potential of this type of unitary RNNs on toy tasks. Wisdom et al. (2016)

provided a more general framework for learning unitary matrices and applied their method on toy tasks and on a real-world speech task.

2.5 Complex-Valued Representation in Deep Neural Networks

Complex-valued representations for deep neural networks have recently started to receive increased attention.

Danihelka et al. (2016) used complex-valued representations in the context of Holographic Reduced Representations (Plate, 1991, 1995, HRRs) to build an associative memory mechanism which is noise-robust. HRRs enable one to store key-value data. Retrieving a value from the memory data associated with a given key can be performed by convolving the whole memory with the key or by applying an inner product between these two. Plate (1991, 1995) leveraged the Convolution Theorem (see section A) by applying the Fast Fourier Transform (FFT) on the keys and the data. This allows one to perform an elementwise multiplication between their Fourier transforms and apply an inverse FFT to convert the result to the time domain. This is equivalent to performing convolution between the key and the data in the time domain and has the advantage of being less expensive as the computational complexity of each of the FFT and its inverse is $\mathcal{O}(N\, log(N))$. Danihelka et al. (2016) used an associative memory to augment the capacity of LSTMs and to increase their robustness to noise and interference. For that, they applied independent random permutations on the memory to create multiple copies of it. This enables one to obtain decorrelated noise in each of the permuted copies. A complex multiplication is then performed between the key and each of the copies. A signal averaging on the resulted multiplications reduces the decorrelated noise in them and strengthens the Signal-To-Noise Ratio (SNR) of the retrieved signal. Danihelka et al. (2016), however, have not relied on FFTs in order to convert the temporal signals to the frequency domain. In fact, they assumed that complex-valued multiplication between the key and the data is itself enough to perform retrieval, and they have assumed that for each input representation the first half is real and the second one is imaginary.

During this decade, interest in Fourier domain representations has started to grow in the machine learning community. Bruna et al. (2013) introduced a generalization of convolutions to graphs using the Graph Fourier Transform, which is in turn defined as the multiplication of a graph signal by the eigenvector matrix of the graph Laplacian. However, the computation of the eigenvector matrix is expensive. Recently, methods that are computationally more efficient have been introduced in Defferrard et al. (2016) and Kipf and Welling (2016) to avoid an explicit use of the Graph Fourier basis. In the context of Convolutional Neural Networks, Rippel et al. (2015) introduced spectral pooling, which allows one to perform pooling in the frequency domain. This enables maintaining the output spatial dimensionality, and thus retaining significantly more information than other pooling approaches. Rippel et al. (2015) have also observed that the parametrization of the convolution filters in the Fourier domain induces faster convergence during training. However, the authors avoid performing complex-valued convolutions, instead building from real-valued kernels in the spatial domain. To ensure that a complex parametrization in the spectral domain maps onto real-valued kernels, the authors impose a conjugate symmetry constraint on the spectral-domain weights, such that when the inverse Fourier transform is applied to them, it only yields real-valued kernels.

Arjovsky et al. (2016) designed a recurrent neural network (RNN) where the transition hidden matrix is unitary. More specifically, the hidden transition matrix is constructed using the product of specific unitary transformations including the Discrete Fourier Transform and its inverse. This allows preserving the norm of the hidden state, and as a consequence, avoids the problem of vanishing and exploding gradients. Wisdom et al. (2016) and Arjovsky et al. (2016) showed that using complex numbers in recurrent neural networks (RNNs) allows to have a richer representational capacity and unitary hidden transition parameters allows faster convergence properties.

Wolter and Yao (2018b) designed an RNN where the input is converted to the frequency domain using a Short Time Fourier Transform (STFT). The output is converted back to the time domain by applying an inverse STFT. Zhang et al. (2018a) proposed a Fourier Recurrent Unit (FRU) where they showed that FRU has gradient lower and upper bounds independent of the temporal dimension. They have also demonstrated the great expressivity of the sparse Fourier basis from which the FRU draws its power.

2.6 Audio Source Speech Separation

Real-world speech communication often takes place in overcrowded environments where there are multiple speakers. Human hearing systems have the potential to focus the attention on a particular sound source, while disabling all other surrounding voices and sounds. This is recognized as the cocktail party effect (Cherry, 1953). The latter remains one of the most pursued research issues in automatic speech recognition and source separation. The growing interest in a reliable speech processing system, in noisy environments, stems from its valuable application to varied human-interaction systems, such as smartphones, personal computers, televisions, etc. A reliable speech processing system that is aimed to work in such noisy environments needs to be endowed with the ability to separate speech of different talkers. This task, which is natural for humans, has shown to be very difficult to model. Indeed the decomposition of an audio signal into its individual constituents is a hard task for the following reasons : multiple sounds may overlap and create partial obscurities, the origins of sounds may not be identified a priori even if the sources may come from the same class and share comparable features, and finally a number of different interfering sources may be present. Recent deep learning systems have made significant progress toward solving this problem.

Audio source separation aims at decomposing an input audio signal, which is the aggregation (fusion) of many sources, so as to recover these multiple independent target audio sources (i.e., speakers). Speech separation is one of the fundamental problems in audio processing as it refers to the instance where the background is highly non-stationary and may hide difficult sources such as singing voices, or other types of speech waveforms.

Initially it takes as input a time-domain waveform which is a raw mixture of audio signals, and then applies the Short Time Fourier transform (STFT) to obtain a time-frequency (T-F) encoding of the mixture sound (i.e., Spectrograms). Finally the T-F bins that match each source are then isolated, and used for the decoding (synthesis) of the hidden source waveforms using the inverse STFT. Complex matrix representation of the spectrograms integrates magnitude and phase components. For an effective audio speech separation, accurate estimation and integration of both components is highly recommended. In this context a major difficulty of dissociating phase and magnitude has been witnessed in previous work. The phase of the clean speech has been often omitted due to the difficulty of its estimation. This has led to the state where most approaches often proceed to estimating the magnitude of the spectrum as a time-frequency mask for each source, and then synthesizing using the masked magnitude spectrogram and reusing the phase of the corrupted sound mixture. (Huang et al., 2014a; Xu et al., 2014; Grais et al., 2016; Nugraha et al., 2016; Takahashi et al., 2018). Reusing phase of the original corrupted speech has a negative impact on the quality of the inference of the clean speech. The impact is amplified when noisy conditions deteriorate, i.e., when Signal-to-Noise Ratio (SNR) is low.

Extensive studies have been devoted to this subject for a quite some time. Wang and Chen (2018) provide a broad overview of recent audio-only methods based on deep learning that tackle both speech denoising (Erdogan et al., 2015; Weninger et al., 2015) and speech separation tasks. Speech separation had long been considered as a signal extraction problem. Lately speech separation has been cast as a supervised learning problem where the discriminative features of speech, speakers and background perturbations are learned from training data. Recently, we have witnessed an important increase in the volume of the papers where supervised separation of algorithms are put forward. The contribution of deep learning to supervised speech separation has drastically accelerated progress and boosted performance. Wang and Chen (2018) provide a comprehensive overview of the recent research of deep learning applied to speech separation. The algorithms designed for audio source separation relate to monaural methods which include speech enhancement (binary discrimination between speech and non speech), speaker separation, speech dereverberation and finally, multi-mircophones techniques. Our discussion here only focuses of selective overview of closely related work on monaural speaker separation methods.

Early efforts at untangling multiple speakers in an audio source have usually involved either pure audio inputs that are separated from any type of corrupting noise or particular microphone configurations for robust supervision (Duong et al., 2010). Succeeding research work in this area assumed in some cases monophonic audio signals (Wang and Plumbley, 2006; Virtanen, 2007; Smaragdis et al., 2007; Spiertz, 2009; Huang et al., 2014a) on which some familiar matrix decomposition procedures were applied such as Independent Component Analysis (Hyvärinen and Oja, 2000), Sparse Decomposition (Zibulevsky and Pearlmutter, 2001), Non-negative Matrix Factorization (Févotte et al., 2009; Févotte and Idier, 2010; Liutkus et al., 2014), and Probabilistic Latent Variables models (Smaragdis et al., 2006). For the source separation task, matrix decomposition approaches appear to have some limits. First, most of them operate on the frequency domain without taking into account the phase component of the signals. In addition, for large data sets, performing composition can be computationally expensive and the decomposition task can be sensitive to the fixed number of spectral bases chosen to encode the signal in hand. In addition as Hershey et al. (2015) argue, model assumptions barely ever match data and inference is intractable because of the difficulty to discriminatively train.

More recently, there has been growing interest in leveraging deep learning techniques (Huang et al., 2014a; Hershey et al., 2015; Gao et al., 2018; Ephrat et al., 2018) to tackle the problem of speech separation. The methods proposed so far can be grouped into two categories : audio-video and audio-only speech separation methods. The first category exploit visual information in video in order to carry out the task of audio source separation. In this category a number of independent and parallel works have recently emerged. They address the problem of audio-visual sound source separation using deep neural network representations.

Two recent examples in this category include Gao et al. (2018) and Ephrat et al. (2018). Gao et al. (2018) addressed the closely connected problem of sorting out the sound of multiple on-screen objects (e.g. musical instruments). The authors perform two-step modeling procedure. Their first step consists of performing a non-negative matrix factorization on the audio channel in order to discover latent sound representations for each physical object detected. The subsequent step is a classification task achieved by training a multi-instance multi-label neural network to map the spectral audio bases to the distribution of detected visual objects. One shortcoming is that the objects in the videos are identified without guidance from the subsequent classification task, thus, some valuable information could be missed, as a result of overfitting. Ephrat et al. (2018) introduced a hybrid speaker independent audio-visual model, where they combine the strength of feed-forward convolutional networks and the short and long-run dynamics of bidirectional LSTM, to conjointly extract visual features of distinct speakers and their corresponding audio signals. While both Gao et al. (2018) and Ephrat et al. (2018) leverage visual and auditory signals as a means to achieve high separation quality, in our work, we rely however on audio source only.

For the audio-only speech separation category the use of deep learning techniques has also gained growing interest in recent years. Our work falls into this category of methods. Huang et al. (2014a) were the earliest to use a deep learning approach to modeling monaural speech separation. They combine of a feed-forward and a recurrent network that are jointly optimized with a soft masking function. Concomitantly, in a closely-related work, Du et al. (2014) proposed a neural network to estimate the log power spectrum of the target speakers. Hershey et al. (2015) proposed a deep clustering approach to speech separation. The basic idea is to learn high-dimensional embedding of the mixture signals ; then standard clustering techniques use the obtained embedding to separate the speech targets. The deep attractor network suggested by Chen et al. (2016) is an extension of the deep clustering approach. The network also creates the so-called "attractors" to better cluster time-frequency points dominated by different speakers. The aforementioned approaches estimate only the magnitude of the STFTs and reconstruct the time-domain signal. Similarly to our work, other papers have recently proposed to integrate the phase-information within a speech separation system. The work by Erdogan et al. (2015), for instance, proposes to train a deep neural network with a phase-sensitive loss. Another noteworthy attempt has been described in Wang et al. (2018), where the neural network still estimates the magnitude of the spectrum, but the time-domain speech signals are retrieved directly. Further, another trend, instead of explicitly integrating phase-information, performs speech separation in the time domain directly, as described in Venkataramani and Smaragdis (2018). Likewise, the TasNet architectures proposed in Luo and Mesgarani (2017) and Luo and Mesgarani (2018) accomplish speech separation using the mixed time signal as input. TasNet directly models the mixture waveform using an encoder-decoder framework, and performs the separation on the output of the encoder.

The studies by Lee et al. (2017); Hu and Wang (2004); Huang et al. (2014a) are more closely related to our work as they address the speech separation problem taking into account phase information. However, this was done without leveraging the recent advances in complex-valued deep learning. The authors in Lee et al. (2017) address the audio-source separation problem using a fully complex-valued deep neural network that learns the nonlinear relationship between an input sound and its distinct sources. Both the activations and weights of the network are complex-valued. For a more comprehensive review of most of these techniques, we refer the readers to Wang and Chen (2018).

Previous work has shown that one way to go around the decoupling of magnitude/phase problem is to estimate a masking function to be applied to the noisy spectrum based on appropriate objective function. In addition, better modeling of dynamics has shown to improve the quality of the speech separation process. In our work, we follow both of these tracks. Erdogan et al. (2015) are among the first to propose a masking based-approach by incorporating phase information (a phase-sensitive mask (PSM)). Their sensitive objective function is based on the signal-to-noise ratio (SNR) of the signal to be recreated. Because of reusing noisy phase the performance of PSM was shown to be limited (Wang et al., 2016). Subsequent work suggested the use of complex-valued ratio masks (cRM). Williamson et al. (2016) define the complex ideal ratio mask (cIRM) and train a deep neural network to jointly estimate real and imaginary components of the cIRM. The authors argue that by operating in the complex domain, the cIRM is able to overcome the decoupling phase/magnitude dilemma simultaneously and enhance both components. However, this method is found to be restricted because it uses only the error between the cIRM and its expected counterpart at the training stage, which often results in performance deterioration (Wang et al., 2014; Yu et al., 2017). In a more recent article, Ephrat et al. (2018) propose a cIRM in which each of the real and imaginary parts is a sigmoidal transformation of the corresponding real and imaginary components of the model output. Despite its advantages, the previous approach to cRM seems to have its main limitations (Choi et al., 2019). Choi et al. (2019) argue that, first the suggested loss function does not capture the cIRM distribution appropriately. Secondly, the masking method produces a cRM with delimited rotation range of 0 - 90 degrees, which prevents an accurate adjustment of the noisy phase. Lee et al. (2017) address the audio-source separation problem using a fully complex-valued deep neural network that learns the nonlinear relationship between an input sound and its distinct sources. Both the activations and weights of the network are complex-valued.

Except Choi et al. (2019), who used the building blocks that we propose in this thesis, previous approaches are incomplete in the sense that they essentially only replace real-valued architectures by complex-valued equivalents and leave out other important deep learning building blocks such as initialization, and the normalization methods known to be extremely important in a real-valued models.

Motivated by all the shortcomings and challenges we have highlighted in the previous sections, we aim, in our work, at taking advantage of the compelling properties of the complex-valued building blocks and framework that we have developed in the first article in order to break the magnitude/phase decoupling deadlock. More specifically we propose a new deep complex separator for audio source separation. Our Deep Separator is a novel masking-based method founded on a complex-valued version of Feature-wise Linear Modulation (FiLM) (Perez et al., 2018). This new masking method is the keystone of our proposed speech separation architecture. We also propose a new explicitly phase-aware loss, which is amplitude and shift-invariant, taking into account the complex-valued components of the spectrogram. We compare it to several other phase-aware losses operating in both time and frequency domains. Our experimental results on the standard Wall Street Journal Dataset demonstrate the effectiveness of the proposed model and loss, highlighting the potential of deep complex-valued networks to improve performance on audio source separation tasks.

CHAPTER 3 ARTICLE 1 : DEEP COMPLEX NETWORKS

This chapter is a verbatim copy of our accepted paper at the International Conference on Learning Representations (ICLR) 2019. The paper submission and reviews could be found in the following link https ://openreview.net/forum ?id=H1T2hmZAb . The authors of the paper are : Chiheb Trabelsi, Olexa Bilaniuk, Ying Zhang, Dmitriy Serdyuk, Sandeep Subramanian, João Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio and Christopher J Pal.

At present, the vast majority of building blocks, techniques, and architectures for deep learning are based on real-valued operations and representations. However, recent work on recurrent neural networks and older fundamental theoretical analysis suggests that complex numbers could have a richer representational capacity and could also facilitate noise-robust memory retrieval mechanisms. Despite their attractive properties and potential for opening up entirely new neural architectures, complex-valued deep neural networks have been marginalized due to the absence of the building blocks required to design such models. In this work, we provide the key atomic components for complex-valued deep neural networks and apply them to convolutional feed-forward networks and convolutional LSTMs. More precisely, we rely on complex convolutions and present algorithms for complex batch-normalization, complex weight initialization strategies for complex-valued neural nets and we use them in experiments with end-to-end training schemes. We demonstrate that such complex-valued models are competitive with their real-valued counterparts. We test deep complex models on several computer vision tasks, on music transcription using the MusicNet dataset and on Speech Spectrum Prediction using the TIMIT dataset. We achieve state-of-the-art performance on these audio-related tasks.

3.1 Introduction

Recent research advances have made significant progress in addressing the difficulties involved in learning deep neural network architectures. Key innovations include normalization techniques (Ioffe and Szegedy, 2015; Salimans and Kingma, 2016) and the emergence of gating-based feed-forward neural networks like Highway Networks (Srivastava et al., 2015). Residual networks (He et al., 2016a,b) have emerged as one of the most popular and effective strategies for training very deep convolutional neural networks (CNNs). Both highway networks and residual networks facilitate the training of deep networks by providing shortcut paths for easy gradient flow to lower network layers thereby diminishing the effects of vanishing gradients (Hochreiter, 1991). He et al. (2016b) show that learning explicit residuals of layers helps in avoiding the vanishing gradient problem and provides the network with an easier optimization problem. Batch normalization (Ioffe and Szegedy, 2015) demonstrates that standardizing the activations of intermediate layers in a network across a minibatch acts as a powerful regularizer as well as providing faster training and better convergence properties. Further, such techniques that standardize layer outputs become critical in deep architectures due to the vanishing and exploding gradient problems.

The role of representations based on complex numbers has started to receive increased attention, due to their potential to enable easier optimization (Nitta, 2002), better generalization characteristics (Hirose and Yoshida, 2012), faster learning (Arjovsky et al., 2016; Danihelka et al., 2016; Wisdom et al., 2016) and to allow for noise-robust memory mechanisms (Danihelka et al., 2016). Wisdom et al. (2016) and Arjovsky et al. (2016) show that using complex numbers in recurrent neural networks (RNNs) allows the network to have a richer representational capacity. Danihelka et al. (2016) present an LSTM (Hochreiter and Schmidhuber, 1997) architecture augmented with associative memory with complex-valued internal representations. Their work highlights the advantages of using complex-valued representations with respect to retrieval and insertion into an associative memory. In residual networks, the output of each block is added to the output history accumulated by summation until that point. An efficient retrieval mechanism could help to extract useful information and process it within the block.

In order to exploit the advantages offered by complex representations, we present a general formulation for the building components of complex-valued deep neural networks and apply it to the context of feed-forward convolutional networks and convolutional LSTMs. Our contributions in this paper are as follows :

1. A formulation of complex batch normalization, which is described in Section 3.3.5 ;
2. Complex weight initialization, which is presented in Section 3.3.6 ;
3. A comparison of different complex-valued ReLU-based activation functions presented in Section 3.4.1 ;
4. A state of the art result on the MusicNet multi-instrument music transcription dataset, presented in Section 3.4.2 ;
5. A state of the art result in the Speech Spectrum Prediction task on the TIMIT dataset, presented in Section 3.4.3.

We perform a sanity check of our deep complex network and demonstrate its effectiveness on standard image classification benchmarks, specifically, CIFAR-10, CIFAR-100. We also use a reduced-training set of SVHN that we call SVHN*. For audio-related tasks, we perform a music transcription task on the MusicNet dataset and a Speech Spectrum prediction task on TIMIT. The results obtained for vision classification tasks show that learning complex-valued representations results in performance that is competitive with the respective real-valued architectures. Our promising results in music transcription and speech spectrum prediction underscore the potential of deep complex-valued neural networks applied to acoustic related tasks [1] – We continue this paper with discussion of motivation for using complex operations and related work.

3.2 Motivation and Related Work

Using complex parameters has numerous advantages from computational, biological, and signal processing perspectives. From a computational point of view, Danihelka et al. (2016) has shown that Holographic Reduced Representations (Plate, 2003), which use complex numbers, are numerically efficient and stable in the context of information retrieval from an associative memory. Danihelka et al. (2016) insert key-value pairs in the associative memory by addition into a *memory trace*. Although not typically viewed as such, residual networks (He et al., 2016a,b) and Highway Networks (Srivastava et al., 2015) have a similar architecture to associative memories : each ResNet residual path computes a residual that is then inserted – by summing into the "memory" provided by the identity connection. Given residual networks' resounding success on several benchmarks and their functional similarity to associative memories, it seems interesting to marry both together. This motivates us to incorporate complex weights and activations in residual networks. Together, they offer a mechanism by which useful information may be retrieved, processed and inserted in each residual block.

Orthogonal weight matrices provide a novel angle of attack on the well-known vanishing and exploding gradient problems in RNNs. Unitary RNNs (Arjovsky et al., 2016) are based on unitary weight matrices, which are a complex generalization of orthogonal weight matrices. Compared to their orthogonal counterparts, unitary matrices provide a richer representation, for instance being capable of implementing the discrete Fourier transform, and thus of discovering spectral representations. Arjovsky et al. (2016) show the potential of this type of recurrent neural networks on toy tasks. Wisdom et al. (2016) provided a more general framework for learning unitary matrices and they applied their method on toy tasks and on a real-world speech task.

Using complex weights in neural networks also has biological motivation. Reichert and Serre (2013) have proposed a biologically plausible deep network that allows one to construct richer

[1] The source code is located at `http://github.com/ChihebTrabelsi/deep_complex_networks`

and more versatile representations using complex-valued neuronal units. The complex-valued formulation allows one to express the neuron's output in terms of its firing rate and the relative timing of its activity. The amplitude of the complex neuron represents the former and its phase the latter. Input neurons that have similar phases are called *synchronous* as they add constructively, whereas *asynchronous* neurons add destructively and thus interfere with each other. This is related to the gating mechanism used in both deep feed-forward neural networks (Srivastava et al., 2015; van den Oord et al., 2016b,a) and recurrent neural networks (Hochreiter and Schmidhuber, 1997; Cho et al., 2014; Zilly et al., 2016) as this mechanism learns to synchronize inputs that the network propagates at a given feed-forward layer or time step. In the context of deep gating-based networks, synchronization means the propagation of inputs whose controlling gates simultaneously hold high values. These controlling gates are usually the activations of a sigmoid function. This ability to take into account phase information might explain the effectiveness of incorporating complex-valued representations in the context of recurrent neural networks.

The phase component is not only important from a biological point of view but also from a signal processing perspective. It has been shown that the phase information in speech signals affects their intelligibility (Shi et al., 2006). Also Oppenheim and Lim (1981) show that the amount of information present in the phase of an image is sufficient to recover the majority of the information encoded in its magnitude. In fact, phase provides a detailed description of objects as it encodes shapes, edges, and orientations.

Recently, Rippel et al. (2015) leveraged the Fourier spectral representation for convolutional neural networks, providing a technique for parameterizing convolution kernel weights in the spectral domain, and performing pooling on the spectral representation of the signal. However, the authors avoid performing complex-valued convolutions, instead building from real-valued kernels in the spatial domain. In order to ensure that a complex parametrization in the spectral domain maps onto real-valued kernels, the authors impose a conjugate symmetry constraint on the spectral-domain weights, such that when the *inverse Fourier transform* is applied to them, it only yields real-valued kernels.

As pointed out in Reichert and Serre (2013), the use of complex-valued neural networks (Georgiou and Koutsougeras, 1992; Zemel et al., 1995; Kim and Adalı, 2003; Hirose, 2003; Nitta, 2004) has been investigated long before the earliest deep learning breakthroughs (Hinton et al., 2006; Bengio et al., 2007; Poultney et al., 2007). Recently Reichert and Serre (2013); Bruna et al. (2015); Arjovsky et al. (2016); Danihelka et al. (2016); Wisdom et al. (2016) have tried to bring more attention to the usefulness of deep complex neural networks by providing theoretical and mathematical motivation for using complex-valued deep networks. However, to the best of our knowledge, most of the recent works using complex valued networks have been applied on toy tasks, with the exception of some attempts. In fact, (Oyallon and Mallat, 2015; Tygert et al., 2015; Worrall et al., 2016) have used complex representation in vision tasks. Wisdom et al. (2016) have also performed a real-world speech task consisting of predicting the log magnitude of the future *short time Fourier transform* frames. In Natural Language Processing, (Trouillon et al., 2016; Trouillon and Nickel, 2017) have used complex-valued embeddings. Much remains to be done to develop proper tools and a general framework for training deep neural networks with complex-valued parameters.

Given the compelling reasons for using complex-valued representations, the absence of such frameworks represents a gap in machine learning tooling, which we fill by providing a set of building blocks for deep complex-valued neural networks that enable them to achieve competitive results with their real-valued counterparts on real-world tasks.

3.3 Complex Building Blocks

In this section, we present the core of our work, laying down the mathematical framework for implementing complex-valued building blocks of a deep neural network.

3.3.1 Representation of Complex Numbers

We start by outlining the way in which complex numbers are represented in our framework. A complex number $z = a + ib$ has a real component a and an imaginary component b. We represent the real part a and the imaginary part b of a complex number as logically distinct real valued entities and simulate complex arithmetic using real-valued arithmetic internally. Consider a typical real-valued $2D$ convolution layer that has N feature maps such that N is divisible by 2; to represent these as complex numbers, we allocate the *first* $N/2$ feature maps to represent the real components and the remaining $N/2$ to represent the imaginary ones. Thus, for a four dimensional weight tensor W that links N_{in} input feature maps to N_{out} output feature maps and whose kernel size is $m \times m$ we would have a weight tensor of size $\left(N_{out} \times N_{in} \times m \times m\right)/2$ complex weights.

3.3.2 Complex Convolution

In order to perform the equivalent of a traditional real-valued 2D convolution in the complex domain, we convolve a complex filter matrix $\mathbf{W} = \mathbf{A} + i\mathbf{B}$ by a complex vector $\mathbf{h} = \mathbf{x} + i\mathbf{y}$ where $\mathbf{A}$ and $\mathbf{B}$ are real matrices and $\mathbf{x}$ and $\mathbf{y}$ are real vectors since we are simulating complex arithmetic using real-valued entities. As the convolution operator is distributive, convolving the vector $\mathbf{h}$ by the filter $\mathbf{W}$ we obtain :

$$\mathbf{W} * \mathbf{h} = \ (\mathbf{A} * \mathbf{x} - \mathbf{B} * \mathbf{y}) + i\,(\mathbf{B} * \mathbf{x} + \mathbf{A} * \mathbf{y}). \tag{3.1}$$

As illustrated in Figure 3.1a, if we use matrix notation to represent real and imaginary parts of the convolution operation we have :

$$\begin{bmatrix} \Re(\mathbf{W} * \mathbf{h}) \\ \Im(\mathbf{W} * \mathbf{h}) \end{bmatrix} = \begin{bmatrix} \mathbf{A} & -\mathbf{B} \\ \mathbf{B} & \mathbf{A} \end{bmatrix} * \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}. \tag{3.2}$$

3.3.3 Complex Differentiability

In order to perform backpropagation in a complex-valued neural network, a sufficient condition is to have a cost function and activations that are *differentiable* with respect to the real and imaginary parts of each complex parameter in the network. See Section 3.6.3 in the Appendix for the complex chain rule.

By constraining activation functions to be *complex differentiable* or *holomorphic*, we restrict the use of possible activation functions for a complex valued neural networks (For further details about holomorphism please refer to Section 3.6.2 in the appendix). Hirose and Yoshida (2012) shows that it is unnecessarily restrictive to limit oneself only to holomorphic activation functions ; Those functions that are differentiable with respect to the real part and the imaginary part of each parameter are also compatible with backpropagation. (Arjovsky et al., 2016; Wisdom et al., 2016; Danihelka et al., 2016) have used non-holomorphic activation functions and optimized the network using regular, real-valued backpropagation to compute partial derivatives of the cost with respect to the real and imaginary parts.

Even though their use greatly restricts the set of potential activations, it is worth mentioning that holomorphic functions can be leveraged for computational efficiency purposes. As pointed out in Sarroff et al. (2015), using holomorphic functions allows one to share gradient values (because the activation satisfies the Cauchy-Riemann equations 3.11 and 3.12 in the appendix). So, instead of computing and backpropagating 4 different gradients, only 2 are required.

3.3.4 Complex-Valued Activations

ModReLU

Numerous activation functions have been proposed in the literature in order to deal with complex-valued representations. Arjovsky et al. (2016) have proposed modReLU, which is defined as follows :

$$\text{modReLU}(z) = \text{ReLU}(|z| + b)\, e^{i\theta_z} = \begin{cases} (|z| + b)\frac{z}{|z|} & \text{if } |z| + b \geq 0, \\ 0 & \text{otherwise,} \end{cases} \tag{3.3}$$

where $z \in \mathbb{C}$, θ_z is the phase of z, and $b \in \mathbb{R}$ is a learnable parameter. As $|z|$ is always positive, a bias b is introduced in order to create a *"dead zone"* of radius b around the origin 0 where the neuron is inactive, and outside of which it is active. The authors have used modReLU in the context of unitary RNNs. Their design of modReLU is motivated by the fact that applying separate ReLUs on both real and imaginary parts of a neuron performs poorly on toy tasks. The intuition behind the design of modReLU is to preserve the pre-activated phase θ_z, as altering it with an activation function severely impacts the complex-valued representation. modReLU does not satisfy the Cauchy-Riemann equations, and thus is not holomorphic. We have tested modReLU in deep feed-forward complex networks and the results are given in Table 3.1.

$\mathbb{C}$ReLU and zReLU

We call Complex ReLU (or $\mathbb{C}$ReLU) the complex activation that applies separate ReLUs on both of the real and the imaginary part of a neuron, i.e :

$$\mathbb{C}\text{ReLU}(z) = \text{ReLU}(\Re(z)) + i\,\text{ReLU}(\Im(z)). \tag{3.4}$$

$\mathbb{C}$ReLU satisfies the Cauchy-Riemann equations when both the real and imaginary parts are at the same time either strictly positive or strictly negative. This means that $\mathbb{C}$ReLU satisfies the Cauchy-Riemann equations when $\theta_z \in]0,\, \pi/2[$ or $\theta_z \in]\pi,\, 3\pi/2[$. We have tested $\mathbb{C}$ReLU in deep feed-forward neural networks and the results are given in Table 3.1.

It is also worthwhile to mention the work done by Guberman (2016) where a ReLU-based complex activation which satisfies the Cauchy-Riemann equations everywhere except for the set of points $\{\Re(z) > 0, \Im(z) = 0\} \cup \{\Re(z) = 0, \Im(z) > 0\}$ is used. The activation function has similarities to $\mathbb{C}$ReLU. We call Guberman (2016) activation as zReLU and is defined as follows :

$$z\text{ReLU}(z) = \begin{cases} z & \text{if } \theta_z \in [0, \pi/2], \\ 0 & \text{otherwise,} \end{cases} \tag{3.5}$$

We have tested zReLU in deep feed-forward complex networks and the results are given in Table 3.1.

3.3.5 Complex Batch Normalization

Deep networks generally rely upon Batch Normalization (Ioffe and Szegedy, 2015) to accelerate learning. In some cases batch normalization is essential to optimize the model. The standard formulation of Batch Normalization applies only to real values. In this section, we propose a batch normalization formulation that can be applied for complex values.

To standardize an array of complex numbers to the standard normal complex distribution, it is not sufficient to translate and scale them such that their mean is 0 and their variance 1. This type of normalization does not ensure equal variance in both the real and imaginary components, and the resulting distribution is not guaranteed to be circular ; It will be elliptical, potentially with high eccentricity.

We instead choose to treat this problem as one of whitening 2D vectors, which implies scaling the data by the inverse square root of their variances and canceling the covariance between the real and the imaginary part. This can be done by multiplying the $\mathbf{0}$-centered data $(\boldsymbol{x} - \mathbb{E}[\boldsymbol{x}])$ by the inverse square root of the 2×2 covariance matrix $\mathbf{V}$:

$$\tilde{\boldsymbol{x}} = (\mathbf{V})^{-\frac{1}{2}} (\boldsymbol{x} - \mathbb{E}[\boldsymbol{x}]) ,$$

where the covariance matrix $\mathbf{V}$ is

$$\mathbf{V} = \begin{pmatrix} V_{rr} & V_{ri} \\ V_{ir} & V_{ii} \end{pmatrix} = \begin{pmatrix} \text{Cov}(\Re\{\boldsymbol{x}\}, \Re\{\boldsymbol{x}\}) & \text{Cov}(\Re\{\boldsymbol{x}\}, \Im\{\boldsymbol{x}\}) \\ \text{Cov}(\Im\{\boldsymbol{x}\}, \Re\{\boldsymbol{x}\}) & \text{Cov}(\Im\{\boldsymbol{x}\}, \Im\{\boldsymbol{x}\}) \end{pmatrix} .$$

The square root and inverse of 2×2 matrices has an inexpensive, analytical solution, and its existence is guaranteed by the positive (semi-)definiteness of $\mathbf{V}$. Positive definiteness of $\mathbf{V}$ is ensured by the addition of $\epsilon \mathbf{I}$ to $\mathbf{V}$ (Tikhonov regularization). The mean subtraction and multiplication by the inverse square root of the variance ensures that $\tilde{\boldsymbol{x}}$ has standard complex distribution with mean $\mu = 0$, covariance $\Gamma = 1$ and pseudo-covariance (also called relation) $C = 0$. The mean, the covariance and the pseudo-covariance are given by :

$$\begin{aligned}
\mu &= \mathbb{E}\left[\tilde{\boldsymbol{x}}\right] \\
\Gamma &= \mathbb{E}\left[\left(\tilde{\boldsymbol{x}} - \mu\right)\left(\tilde{\boldsymbol{x}} - \mu\right)^{*}\right] = V_{rr} + V_{ii} + i\left(V_{ir} - V_{ri}\right) \\
C &= \mathbb{E}\left[\left(\tilde{\boldsymbol{x}} - \mu\right)\left(\tilde{\boldsymbol{x}} - \mu\right)\right] \ = V_{rr} - V_{ii} + i\left(V_{ir} + V_{ri}\right).
\end{aligned} \tag{3.6}$$

The normalization procedure allows one to decorrelate the imaginary and real parts of a unit. This has the advantage of avoiding co-adaptation between the two components which reduces the risk of overfitting (Cogswell et al., 2015; Srivastava et al., 2014).

Analogously to the real-valued batch normalization algorithm, we use two parameters, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$. The shift parameter $\boldsymbol{\beta}$ is a complex parameter with two learnable components (defining the new real and imaginary means). The scaling parameter $\boldsymbol{\gamma}$ is a 2×2 matrix with only three degrees of freedom, and thus only three learnable components. The matrix $(\mathbf{V})^{-\frac{1}{2}}$ normalizes the variance of the input to 1 along both of its original principal components. $\boldsymbol{\gamma}$ scales the input to achieve a new desired covariance. The scaling parameter $\boldsymbol{\gamma}$ is given by :

$$\boldsymbol{\gamma} = \begin{pmatrix} \gamma_{rr} & \gamma_{ri} \\ \gamma_{ri} & \gamma_{ii} \end{pmatrix}.$$

As the normalized input $\tilde{\boldsymbol{x}}$ has real and imaginary variance 1, we initialize both γ_{rr} and γ_{ii} to $1/\sqrt{2}$ in order to obtain a modulus of 1 for the variance of the normalized value. γ_{ri}, $\Re\{\boldsymbol{\beta}\}$ and $\Im\{\boldsymbol{\beta}\}$ are initialized to 0. The complex batch normalization is defined as :

$$\mathrm{BN}\left(\tilde{\boldsymbol{x}}\right) = \boldsymbol{\gamma}\, \tilde{\boldsymbol{x}} + \boldsymbol{\beta}. \tag{3.7}$$

We use running averages with momentum to maintain an estimate of the complex batch normalization statistics during training and testing. The moving averages of V_{ri} and $\boldsymbol{\beta}$ are initialized to 0. The moving averages of V_{rr} and V_{ii} are initialized to $1/\sqrt{2}$. The momentum for the moving averages is set to 0.9.

3.3.6 Complex Weight Initialization

In a general case, particularly when batch normalization is not performed, proper initialization is critical in reducing the risks of vanishing or exploding gradients. To do this, we follow the same steps as in Glorot and Bengio (2010) and He et al. (2015) to derive the variance of the complex weight parameters.

A complex weight has a polar form as well as a rectangular form

$$W = |W|e^{i\theta} = \Re\{W\} + i\ \Im\{W\}, \tag{3.8}$$

where θ and $|W|$ are respectively the argument (phase) and magnitude of W.

Variance is the difference between the *expectation of the squared magnitude* and the *square of the expectation* :

$$\text{Var}(W) = \mathbb{E}\left[WW^*\right] - (\mathbb{E}\left[W\right])^2 = \mathbb{E}\left[|W|^2\right] - (\mathbb{E}\left[W\right])^2,$$

which reduces, in the case of W symmetrically distributed around 0, to $\mathbb{E}\left[|W|^2\right]$. We do not know yet the value of $\text{Var}(W) = \mathbb{E}\left[|W|^2\right]$. However, we do know a related quantity, $\text{Var}(|W|)$, because the magnitude of complex normal values, $|W|$, follows the Rayleigh distribution (Chi-distributed with two degrees of freedom (DOFs)). This quantity is

$$\text{Var}(|W|) = \mathbb{E}\left[|W||W|^*\right] - (\mathbb{E}\left[|W|\right])^2 = \mathbb{E}\left[|W|^2\right] - (\mathbb{E}\left[|W|\right])^2. \tag{3.9}$$

Putting them together :

$$\text{Var}(|W|) = \text{Var}(W) - (\mathbb{E}\left[|W|\right])^2, \text{ and } \text{Var}(W) = \text{Var}(|W|) + (\mathbb{E}\left[|W|\right])^2.$$

We now have a formulation for the variance of W in terms of the variance and expectation of its magnitude, both properties analytically computable from the Rayleigh distribution's single parameter, σ, indicating the *mode*. These are :

$$\mathbb{E}\left[|W|\right] = \sigma\sqrt{\frac{\pi}{2}}, \quad \text{Var}(|W|) = \frac{4-\pi}{2}\sigma^2.$$

The variance of W can thus be expressed in terms of its generating Rayleigh distribution's single parameter, σ, thus :

$$\text{Var}(W) = \frac{4-\pi}{2}\sigma^2 + \left(\sigma\sqrt{\frac{\pi}{2}}\right)^2 = 2\sigma^2. \tag{3.10}$$

If we want to respect the Glorot and Bengio (2010) criterion which ensures that the variances of the input, the output and their gradients are the same, then we would have $\text{Var}(W) = 2/(n_{in} + n_{out})$, where n_{in} and n_{out} are the number of input and output units respectively. In such case, $\sigma = 1/\sqrt{n_{in} + n_{out}}$. If we want to respect the He et al. (2015) initialization that presents an initialization criterion that is specific to ReLUs, then $\text{Var}(W) = 2/n_{in}$ which $\sigma = 1/\sqrt{n_{in}}$.

The magnitude of the complex parameter W is then initialized using the Rayleigh distribution with the appropriate mode σ. We can see from equation 3.10, that the variance of W depends on on its magnitude and not on its phase. We then initialize the phase using the uniform distribution between $-\pi$ and π. By performing the multiplication of the magnitude by the phasor as is detailed in equation 3.8, we perform the complete initialization of the complex parameter.

In all the experiments that we report, we use variant of this initialization which leverages the independence property of unitary matrices. As it is stated in Cogswell et al. (2015), Srivastava et al. (2014), and Tompson et al. (2015), learning decorrelated features is beneficial for learning as it allows to perform better generalization and faster learning. This motivates us to achieve initialization by considering a (semi-)unitary matrix which is reshaped to the size of the weight tensor. Once this is done, the weight tensor is mutiplied by$\sqrt{He_{var}/\mathrm{Var}(W)}$ or$\sqrt{Glorot_{var}/\mathrm{Var}(W)}$ where $Glorot_{var}$ and He_{var} are respectively equal to $2/(n_{in}+n_{out})$ and $2/n_{in}$. In such a way we allow kernels to be independent from each other as much as possible while respecting the desired criterion. Note that we perform the analogous initialization for real-valued models by leveraging the independence property of orthogonal matrices in order to build kernels that are as much independent from each other as possible while respecting a given criterion.

3.3.7 Complex Convolutional Residual Network

A deep convolutional residual network of the nature presented in He et al. (2016a,b) consists of 3 *stages* within which feature maps maintain the same shape. At the end of a stage, the feature maps are downsampled by a factor of 2 and the number of convolution filters are doubled. The sizes of the convolution kernels are always set to 3 x 3. Within a stage, there are several *residual blocks* which comprise 2 convolution layers each. The contents of one such residual block in the real and complex setting is illustrated in Appendix Figure 3.1b.

In the complex valued setting, the majority of the architecture remains identical to the one presented in He et al. (2016b) with a few subtle differences. Since all datasets that we work with have real-valued inputs, we present a way to learn their imaginary components to let the rest of the network operate in the complex plane. We learn the initial imaginary component of our input by performing the operations present within a single real-valued residual block

$$BN \rightarrow ReLU \rightarrow Conv \rightarrow BN \rightarrow ReLU \rightarrow Conv$$

Using this learning block yielded better emprical results than assuming that the input image has a null imaginary part. The parameters of this real-valued residual block are trained by backpropagating errors from the task specific loss function. Secondly, we perform a $Conv \rightarrow BN \rightarrow Activation$ operation on the obtained complex input before feeding it to the first residual block. We also perform the same operation on the real-valued network input instead of $Conv \rightarrow Maxpooling$ as in He et al. (2016b). Inside, residual blocks, we subtly alter the

Table 3.1 Classification error on CIFAR-10, CIFAR-100 and SVHN* using different complex activations functions (zReLU, modReLU and $\mathbb{C}$ReLU). WS, DN and IB stand for the wide and shallow, deep and narrow and in-between models respectively. The prefixes R & C refer to the real and complex valued networks respectively. Performance differences between the real network and the complex network using $\mathbb{C}$ReLU are reported between their respective best models. All models are constructed to have roughly 1.7M parameters except the modReLU models which have roughly 2.5M parameters. modReLU and zReLU were largely outperformed by $\mathbb{C}$ReLU in the reported experiments. Due to limited resources, we haven't performed all possible experiments as the conducted ones are already conclusive. A "-" is filled in front of an unperformed experiment.

Arch	CIFAR-10			CIFAR-100			SVHN*		
	zReLU	modReLU	$\mathbb{C}$ReLU	zReLU	modReLU	$\mathbb{C}$ReLU	zReLU	modReLU	$\mathbb{C}$ReLU
CWS	11.71	23.42	6.17	-	50.38	**26.36**	80.41	7.43	3.70
CDN	9.50	22.49	6.73	-	50.64	28.22	80.41	-	3.72
CIB	11.36	23.63	5.59	-	48.10	28.64	4.98	-	3.62
	ReLU			ReLU			ReLU		
RWS	**5.42**			27.22			**3.42**		
RDN	6.29			27.84			3.52		
RIB	6.07			27.71			4.30		
DIFF	-0.17			+0.86			-0.20		

way in which we perform a projection at the end of a stage in our network. We concatenate the output of the last residual block with the output of a 1x1 convolution applied on it with the same number of filters used throughout the stage and subsample by a factor of 2. In contrast, He et al. (2016b) perform a similar 1x1 convolution with twice the number of feature filters in the current stage to both downsample the feature maps spatially and double them in number.

3.4 Experimental Results

In this section, we present empirical results from using our model to perform image, music classification and spectrum prediction. First, we present our model's architecture followed by the results we obtained on CIFAR-10, CIFAR-100, and SVHN* as well as the results on automatic music transcription on the MusicNet benchmark and speech spectrum prediction on TIMIT.

Table 3.2 Classification error on CIFAR-10, CIFAR-100 and SVHN* using different normalization strategies. NCBN, CBN and BN stand for a Naive variant of the complex batch-normalization, complex batch-normalization and regular batch normalization respectively. (R) & (C) refer to the use of the real- and complex-valued convolution respectively. The complex models use ℂReLU as activation. All models are constructed to have roughly 1.7M parameters. 5 out of 6 experiments using the naive variant of the complex batch normalization failed with the apparition of NaNs during training. As these experiments are already conclusive and due to limited resources, we haven't conducted other experiments for the NCBN model. A "-" is filled in front of an unperformed experiment.

Arch	CIFAR-10			CIFAR-100			SVHN*		
	NCBN(C)	CBN(R)	BN(C)	NCBN(C)	CBN(R)	BN(C)	NCBN(C)	CBN(R)	BN(C)
WS	-	**5.47**	6.32	27.29	**26.63**	27.89	NAN	3.80	**3.52**
DN	-	5.89	6.71	NAN	27.13	28.83	NAN	3.54	3.58
IB	-	5.66	6.83	NAN	26.99	29.89	NAN	3.74	3.56

3.4.1 Image Recognition

We adopt an architecture inspired by He et al. (2016b). The latter will also serve as a baseline to compare against. We train comparable real-valued Neural Networks using the standard ReLU activation function. We have tested our complex models with the ℂReLU, zReLU and modRelu activation functions. We use a cross entropy loss for both real and complex models. A global average pooling layer followed by a single fully connected layer with a softmax function is used to classify the input as belonging to one of 10 classes in the CIFAR-10 and SVHN datasets and 100 classes for CIFAR-100.

We consider architectures that trade-off model depth (number of residual blocks per stage) and width (number of convolutional filters in each layer) given a fixed parameter budget. Specifically, we build three different models - wide and shallow (WS), deep and narrow (DN) and in-between (IB). In a model that has roughly 1.7 million parameters, our WS architecture for a complex network starts with 12 complex filters (24 real filters) per convolution layer in the initial stage and 16 residual blocks per stage. The DN architecture starts with 10 complex filters and 23 blocks per stage while the IB variant starts with 11 complex filters and 19 blocks per stage. The real-valued counterpart has also 1.7 million parameters. Its WS architecture starts with 18 real filters per convolutional layer and 14 blocks per stage. The DN architecture starts with 14 real filters and 23 blocks per stage and the IB architecture starts with 16 real filters and 18 blocks per stage.

All models (real and complex) were trained using the backpropagation algorithm with Stochastic Gradient Descent with Nesterov momentum (Nesterov, 1983) set at 0.9. We also clip the norm of our gradients to 1. We tweaked the learning rate schedule used in He et al. (2016b) in both the real and complex residual networks to extract small performance improvements in both. We start our learning rate at 0.01 for the first 10 epochs to warm up the training and then set it at 0.1 from epoch 10-100 and then anneal the learning rates by a factor of 10 at epochs 120 and 150. We end the training at epoch 200.

Table 3.1 presents our results on performing image classification on CIFAR-10, CIFAR-100. In addition, we also consider a truncated version of the Street View House Numbers (SVHN) dataset which we call SVHN*. For computational reasons, we use the required 73,257 training images of Street View House Numbers (SVHN). We still test on all 26,032 images. For all the tasks and for both the real- and complex-valued models, The WS architecture has yielded the best performances. This is in concordance with Zagoruyko and Komodakis (2016) who observed that wider and shallower residual networks perform better than their deeper and narrower counterpart. On CIFAR-10 and SVHN*, the real-valued representation performs slightly better than its complex counterpart. On CIFAR-100, the complex representation outperforms the real one. In general, the obtained results for both representation are quite comparable. To understand the effect of using either real or complex representation for a given task, we designed hybrid models that combine both. Table 3.2 contains the results for hybrid models. We can observe in the Table 3.2 that in cases where complex representation outperformed the real one (wide and shallow on CIFAR-100), combining a real-valued convolutional filter with a complex batch normalization improves the accuracy of the real-valued convolutional model. However, the complex-valued one is still outperforming it. In cases, where real-valued representation outperformed the complex one (wide and shallow on CIFAR-10 and SVHN*), replacing a complex batch normalization by a regular one increased the accuracy of the complex convolutional model. Despite that replacement, the real-valued model performs better in terms of accuracy for such tasks. In general, these experiments show that the difference in efficiency between the real and complex models varies according to the dataset, to the task and to the architecture.

Ablation studies were performed in order to investigate the importance of the 2D whitening operation that occurs in the complex batch normalization. We replaced the complex batch normalization layers with a naive variant (NCBN) which, instead of left multiplying the centred unit by the inverse square root of its covariance matrix, just divides it by its complex variance. Here, this naive variant of CBN is Mimicking the regular BN by not taking into account correlation between the elements in the complex unit. The Naive variant of the Complex Batch Normalization performed very poorly ; In 5 out of 6 experiments, training failed with the appearance of NaNs (See Section 3.6.6 for the explanation). By way of contrast, all 6 complex-valued Batch Normalization experiments converged. Results are given in Table 3.2.

Another ablation study was undertaken to compare $\mathbb{C}$ReLU, modReLU and zRELU. Again the differences were stark : All $\mathbb{C}$ReLU experiments converged and outperformed both modReLU and zReLU, both of which variously failed to converge or fared substantially worse. We think that modReLU didn't perform as well as $\mathbb{C}$ReLU due to the fact that consecutive layers in a feed-forward net do not represent time-sequential patterns, and so, they might need to drop some phase information. Results are reported in Table 3.1. More discussion about phase information encoding is presented in section 3.6.7.

3.4.2 Automatic Music Transcription

In this section we present results for the automatic music transcription (AMT) task. The nature of an audio signal allows one to exploit complex operations as presented earlier in the paper. The experiments were performed on the MusicNet dataset (Thickstun et al., 2016). For computational efficiency we resampled the original input from 44.1kHz to 11kHz using the algorithm described in Smith (2002). This sampling rate is sufficient to recognize frequencies presented in the dataset while reducing computational cost dramatically. We modeled each of the 84 notes that are present in the dataset with independent sigmoids (due to the fact that notes can fire simultaneously). We initialized the bias of the last layer to the value of -5 to reflect the distribution of silent/non-silent notes. As in the baseline, we performed experiments on the raw signal and the frequency spectrum. For complex experiments with the raw signal, we considered its imaginary part equal to zero. When using the spectrum input we used its complex representation (instead of only the magnitudes, as usual for AMT) for both real and complex models. For the real model, we considered the real and imaginary components of the spectrum as separate channels. The model we used for raw signals is a shallow convolutional network similar to the model used in the baseline, with the size reduced by a factor of 4 (corresponding to the reduction of the sampling rate). The filter size was 512 samples (about 12ms) with a stride of 16. The model for the spectral input is similar to the VGG model (Simonyan and Zisserman, 2015). The first layer has filter with size of 7 and is followed by 5 convolutional layers with filters of size 3. The final convolution block is followed by a fully connected layer with 2048 units. The latter is followed, in its turn, by another fully connected layer with 84 sigmoidal units. In all of our experiments we use an input window of 4096 samples or its corresponding FFT (which corresponds to the 16,384 window used in the baseline) and predicted notes in the center of the window. All networks were optimized with Adam (Kingma and Ba, 2014). We start our learning rate at 10^{-3} for the first 10 epochs and then anneal it by a factor of 10 at each of the epochs 100, 120 and 150. We end the training at epoch 200. For the real-valued models, we have used ReLU as activation. $\mathbb{C}$ReLU has been used as activation for the complex-valued models.

The complex network was initialized using the unitary initialization scheme respecting the He criterion as described in Section 3.3.6. For the real-valued network, we have used the analogue initialization of the weight tensor. It consists of performing an orthogonal initialization with a gain of $\sqrt{2}$. The complex batch normalization was applied according to Section 3.3.5. Following Thickstun et al. (2016) we used recordings with ids '2303', '2382', '1819' as the test subset and additionally we created a validation subset using recording ids '2131', '2384', '1792', '2514', '2567', '1876' (randomly chosen from the training set). The validation subset was used for model selection and early stopping. The remaining 321 files were used for training. The results are summarized on Table 3.3. We achieve a performance comparable to the baseline with the shallow convolutional network. our VGG-based deep real-valued model reaches 69.6% average precision on the downsampled data. With significantly fewer parameters than its real counterpart, the VGG-based deep complex model, achieves 72.9% average precision which is the state of the art to the best of our knowledge. See Figures 3.3 and 3.4 in the Appendix for precision-recall curves and a sample of the output of the model.

3.4.3 Speech Spectrum Prediction

We apply both a real Convolutional LSTM (Xingjian et al., 2015) and a complex Convolutional LSTM on speech spectrum prediction task (See section 3.6.5 in the Appendix for the details of the real and complex Convolutional LSTMs). In this task, the model predicts the magnitude spectrum. It implicitly infers the real and imaginary components of the spectrum at time $t+1$, given all the spectrum (imaginary part and real components) up to time t. This is slightly different from (Wisdom et al., 2016). The real and imaginary components are considered as separate channels in both model. We evaluate the model with mean-square-error (MSE) on log-magnitude to compare with the others (Wisdom et al., 2016). The experiments are conducted on a downsampled (8kHz) version of the TIMIT dataset. By following the steps in Wisdom et al. (2016), raw audio waves are transformed into frequency domain via short-time Fourier transform (STFT) with a Hann analysis window of 256 samples and a window hop of 128 samples (50% overlap). We use a training set with 3690 utterances, a validation set with 400 utterances and a standard test set with 192 utterance.

To match the number of parameters for both model, the Convolutional LSTM has 84 feature maps while the complex model has 60 complex feature maps (120 feature maps in total). Adam (Kingma and Ba, 2014) with a fixed learning rate of 1e-4 is used in both experiments.

Table 3.3 MusicNet experiments. *FS* is the sampling rate. *Params* is the total number of parameters. We report the average precision (AP) metric that is the area under the precision-recall curve.

Architecture	FS	Params	AP, %
Shallow, Real	11kHz		66.1
Shallow, Complex	11kHz		66.0
Shallow, Thickstun et al. (2016)	44.1kHz	-	67.8
Deep, Real	11kHz	10.0M	69.6
Deep, Complex	11kHz	8.8M	**72.9**

We initialize the complex model with the unitary initialization scheme and the real model with orthogonal initialization respecting the Glorot criterion. The result is shown in Table 3.4 and the learning curve is shown in Figure 3.5. Our baseline model has achieved the state of the art and the complex convolutional LSTM model performs better over the baseline in terms of MSE and convergence.

3.5 Conclusions

We have presented key building blocks required to train complex valued neural networks, such as complex batch normalization and complex weight initialization. We have also explored a wide variety of complex convolutional network architectures, including some yielding competitive results for image classification and state of the art results for a music transcription task and speech spectrum prediction. We hope that our work will stimulate further investigation of complex valued networks for deep learning models and their application to more challenging tasks such as generative models for audio and images.

Acknowledgements

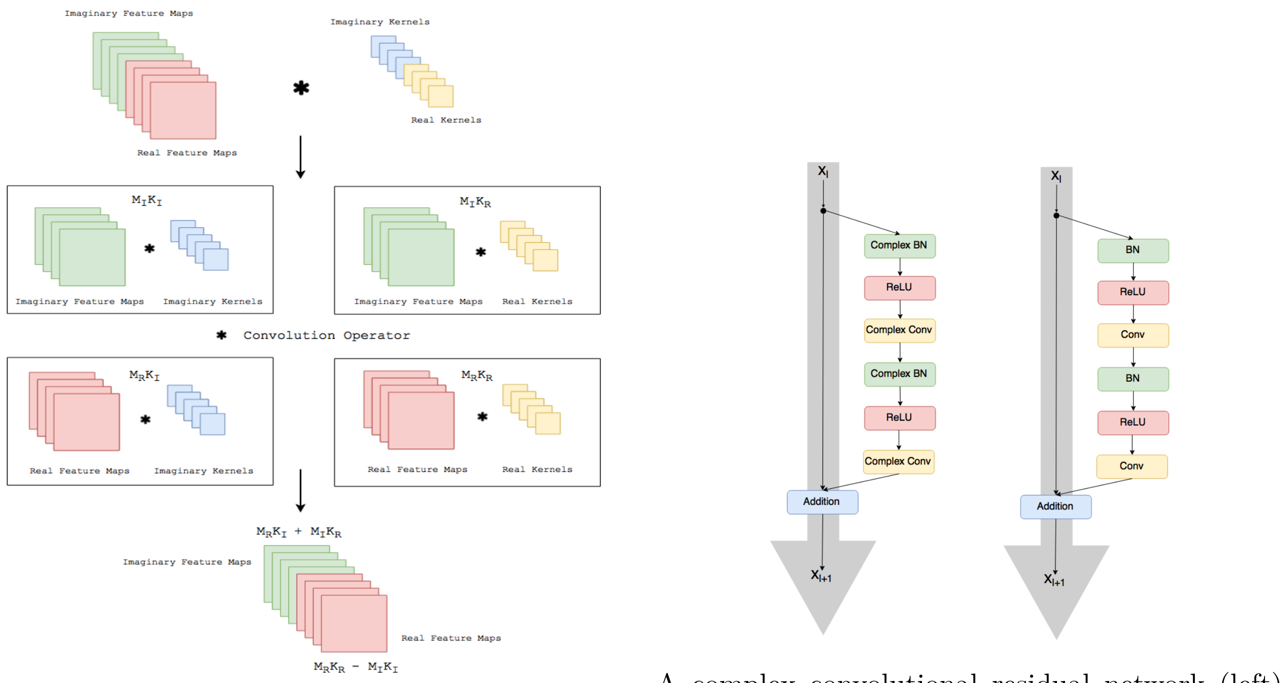
We are grateful to Roderick Murray-Smith, Jörn-Henrik Jacobsen, Jesse Engel and all the students at MILA, especially Jason Jo, Anna Huang and Akram Erraqabi for helpful feedback and discussions. We also thank the developers of Theano (Theano Development Team, 2016) and Keras (Chollet et al., 2015). We are grateful to Samsung and the Fonds de Recherche du Québec – Nature et Technologie for their financial support. We would also like to acknowledge NVIDIA for donating a DGX-1 computer used in this work.

Table 3.4 Speech Spectrum Prediction on TIMIT test set. CConv-LSTM denotes the Complex Convolutional LSTM.

Model	#params	MSE(validation)	MSE(test)
LSTM Wisdom et al. (2016)	$\approx$ 135k	16.59	16.98
Full-capacity uRNN Wisdom et al. (2016)	$\approx$ 135k	14.56	14.66
Conv-LSTM (our baseline)	$\approx$ 88k	11.10	12.18
CConv-LSTM (ours)	$\approx$ 88k	**10.78**	**11.90**

3.6 Appendix

In practice, the complex convolution operation is implemented as illustrated in Fig.3.1a where M_I, M_R refer to imaginary and real feature maps and K_I and K_R refer to imaginary and real kernels. $M_I K_I$ refers to result of a real-valued convolution between the imaginary kernels K_I and the imaginary feature maps M_I.



An illustration of the complex convolution operator.

A complex convolutional residual network (left) and an equivalent real-valued residual network (right).

Figure 3.2 Complex convolution and residual network implementation details.

3.6.1 MusicNet illustrations

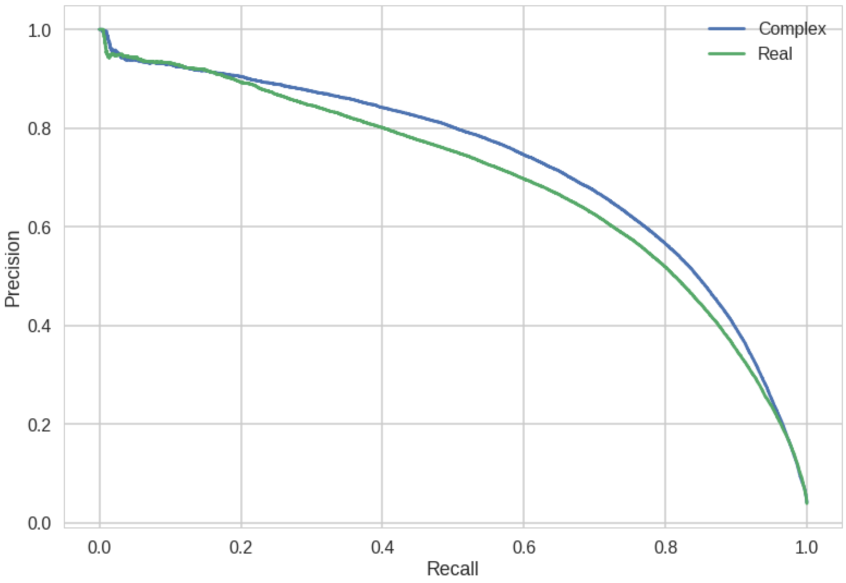


Figure 3.3 Precision-recall curve

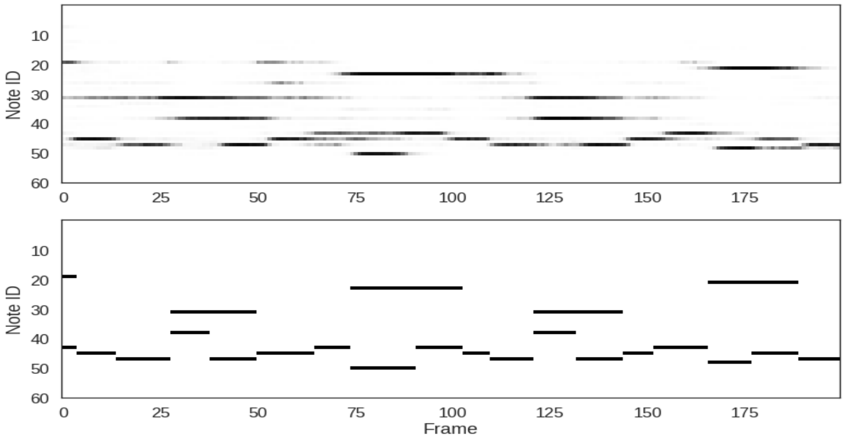


Figure 3.4 Predictions (Top) vs. ground truth (Bottom) for a music segment from the test set.

3.6.2 Holomorphism and Cauchy–Riemann Equations

Holomorphism, also called *analyticity*, ensures that a complex-valued function is *complex differentiable* in the neighborhood of every point in its domain. This means that the derivative, $f'(z_0) \equiv \lim_{\Delta z \to 0} \left[\frac{(f(z_0)+\Delta z)-f(z_0)}{\Delta z} \right]$ of f, exists at every point z_0 in the domain of f where f is a complex-valued function of a complex variable $z = x + i\, y$ such that $f(z) = u(x, y) + i\, v(x, y)$. u and v are real-valued functions. One possible way of expressing Δz is to have $\Delta z = \Delta x + i\, \Delta y$. Δz can approach 0 from multiple directions (along the real axis, imaginary axis or in-between). However, in order to be complex differentiable, $f'(z_0)$ must be the same complex quantity regardless of direction of approach. When Δz approaches 0 along the real axis, $f'(z_0)$ could be written as :

$$\begin{aligned} f'(z_0) &\equiv \lim_{\Delta z \to 0} \left[\frac{(f(z_0)+\Delta z)-f(z_0)}{\Delta z} \right] \\ &= \lim_{\Delta x \to 0} \lim_{\Delta y \to 0} \left[\frac{\Delta u(x_0, y_0) + i\, \Delta v(x_0, y_0)}{\Delta x + i\, \Delta y} \right] \\ &= \lim_{\Delta x \to 0} \left[\frac{\Delta u(x_0, y_0) + i\, \Delta v(x_0, y_0)}{\Delta x + i\, 0} \right]. \end{aligned} \tag{3.11}$$

When Δz approaches 0 along the imaginary axis, $f'(z_0)$ could be written as :

$$\begin{aligned} &= \lim_{\Delta y \to 0} \lim_{\Delta x \to 0} \left[\frac{\Delta u(x_0, y_0) + i\, \Delta v(x_0, y_0)}{\Delta x + i\, \Delta y} \right] \\ &= \lim_{\Delta y \to 0} \left[\frac{\Delta u(x_0, y_0) + i\, \Delta v(x_0, y_0)}{0 + i\, \Delta y} \right] \end{aligned} \tag{3.12}$$

Satisfying equations 3.11 and 3.12 is equivalent of having $\frac{\partial f}{\partial z} = \frac{\partial u}{\partial x} + i\, \frac{\partial v}{\partial x} = -i\, \frac{\partial u}{\partial y} + \frac{\partial v}{\partial y}$. So, in order to be complex differentiable, f should satisfy $\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}$ and $\frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x}$. These are called the Cauchy–Riemann equations and they give a necessary condition for f to be complex differentiable or "*holomorphic*". Given that u and v have continuous first partial derivatives, the Cauchy-Riemann equations become a sufficient condition for f to be holomorphic.

3.6.3 The Genralized Complex Chain Rule for a Real-Valued Loss Function

If L is a real-valued loss function and z is a complex variable such that $z = x + i\, y$ where $x, y \in \mathbb{R}$, then :

$$\nabla_L(z) = \frac{\partial L}{\partial z} = \frac{\partial L}{\partial x} + i \frac{\partial L}{\partial y} = \frac{\partial L}{\partial \Re(z)} + i \frac{\partial L}{\partial \Im(z)} = \Re(\nabla_L(z)) + i\Im\,(\nabla_L(z)). \tag{3.13}$$

Now if we have another complex variable $t = r + i\,s$ where z could be expressed in terms of t and $r, s \in \mathbb{R}$, we would then have :

$$\begin{aligned}
\nabla_L(t) &= \frac{\partial L}{\partial t} = \frac{\partial L}{\partial r} + i\frac{\partial L}{\partial s} \\
&= \frac{\partial L}{\partial x}\frac{\partial x}{\partial r} + \frac{\partial L}{\partial y}\frac{\partial y}{\partial r} + i\Big(\frac{\partial L}{\partial x}\frac{\partial x}{\partial s} + \frac{\partial L}{\partial y}\frac{\partial y}{\partial s}\Big) \\
&= \frac{\partial L}{\partial x}\Big(\frac{\partial x}{\partial r} + i\frac{\partial x}{\partial s}\Big) + \frac{\partial L}{\partial y}\Big(\frac{\partial y}{\partial r} + i\frac{\partial y}{\partial s}\Big) \qquad (3.14) \\
&= \frac{\partial L}{\partial \Re(z)}\Big(\frac{\partial x}{\partial r} + i\frac{\partial x}{\partial s}\Big) + \frac{\partial L}{\partial \Im(z)}\Big(\frac{\partial y}{\partial r} + i\frac{\partial y}{\partial s}\Big) \\
&= \Re(\nabla_L(z))\Big(\frac{\partial x}{\partial r} + i\frac{\partial x}{\partial s}\Big) + \Im(\nabla_L(z))\Big(\frac{\partial y}{\partial r} + i\frac{\partial y}{\partial s}\Big).
\end{aligned}$$

3.6.4 Computational Complexity and FLOPS

In terms of computational complexity, the convolutional operation and the complex batchnorm are of the same order as their real counterparts. However, as a complex multiplication is 4 times more expensive than its real counterpart, all complex convolutions are 4 times more expensive as well.

Additionally, the complex BatchNorm is not implemented in cuDNN and therefore had to be simulated with a sizeable sequence of elementwise operations. This leads to a ballooning of the number of nodes in the compute graph and to inefficiencies due to lack of effective operation fusion. A dedicated cuDNN kernel will, however, reduce the cost to little more than that of the real-valued BatchNorm.

Ignoring elementwise operations, which constitute a negligible fraction of the floating-point operations in the neural network, we find that for all architectures in 3.1 and for all of CIFAR10, CIFAR100 or SVHN, the inference cost in real FLOPS per example is roughly identical. It is ~ 265 MFLOPS for the $\mathbb{R}$-valued variant and ~ 1030 MFLOPS for the $\mathbb{C}$-valued variant of the architecture, approximately quadruple.

3.6.5 Convolutional LSTM

A Convolutional LSTM is similar to a fully connected LSTM. The only difference is that, instead of using matrix multiplications to perform computation, we use convolutional operations. The computation in a real-valued Convolutional LSTM is defined as follows :

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{xi} * \mathbf{x}_t + \mathbf{W}_{hi} * \mathbf{W}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_{xf} * \mathbf{x}_t + \mathbf{W}_{hf} * \mathbf{h}_{t-1} + \mathbf{b}_f) \\
\mathbf{c}_t &= \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tanh(\mathbf{W}_{xc} * \mathbf{x}_t + \mathbf{W}_{hc} * \mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{xo} * \mathbf{x}_t + \mathbf{W}_{ho} * \mathbf{h}_{t-1} + \mathbf{b}_o) \\
\mathbf{h}_t &= \mathbf{o}_t \circ \tanh(\mathbf{c}_t)
\end{aligned} \tag{3.15}$$

Where σ denotes the sigmoidal activation function, $\circ$ the elementwise multiplication and $*$ the real-valued convolution. $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$ represent the vector notation of the input, forget and output gates respectively. $\mathbf{c}_t$ and $\mathbf{h}_t$ represent the vector notation of the cell and hidden states respectively. the gates and states in a ConvLSTM are tensors whose last two dimensions are spatial dimensions. For each of the gates, $\mathbf{W}_{xgate}$ and $\mathbf{W}_{hgate}$ are respectively the input and hidden kernels.

For the Complex Convolutional LSTM, we just replace the real-valued convolutional operation by its complex counterpart. We maintain the real-valued elementwise multiplication. The sigmoid and tanh are both performed separately on the real and the imaginary parts.

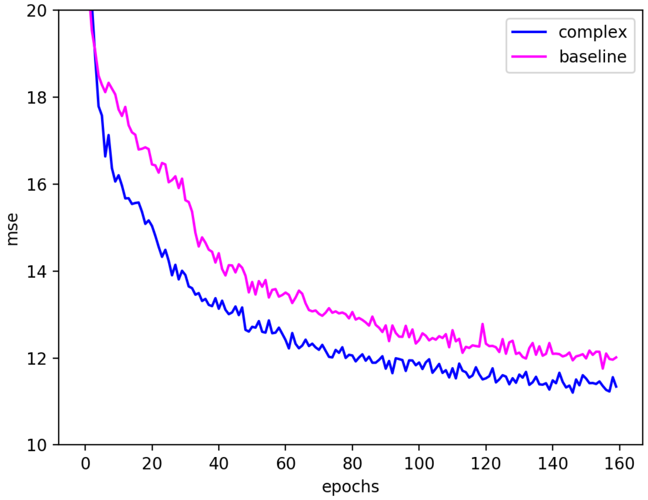


Figure 3.5 Learning curve for speech spectrum prediction from dev set.

3.6.6 Complex Standardization and Internal Covariate Shift

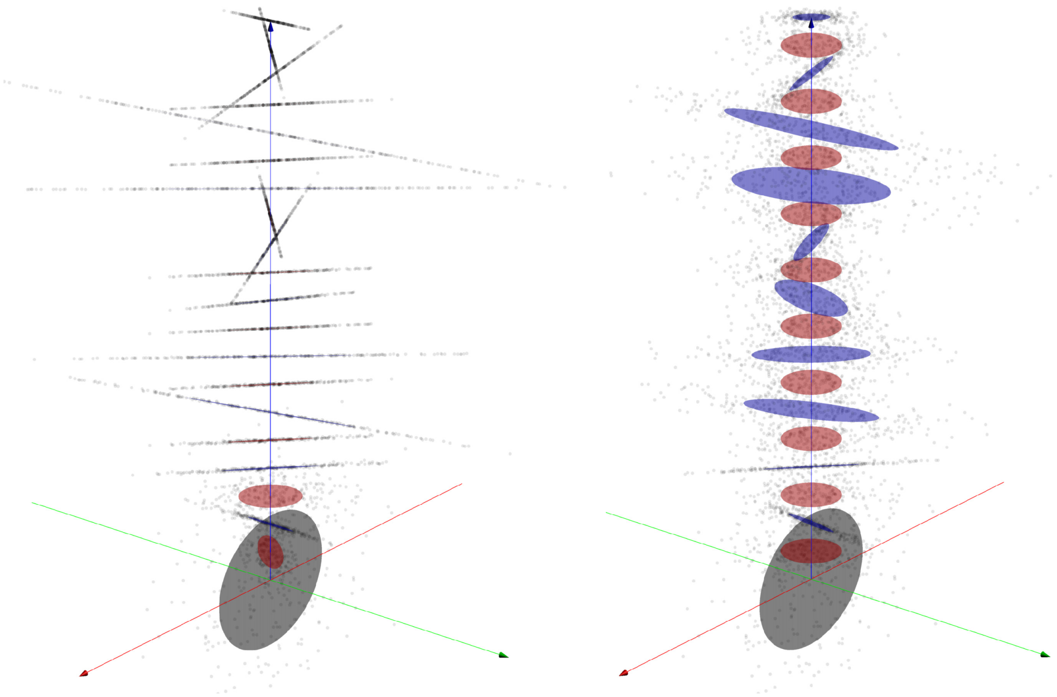
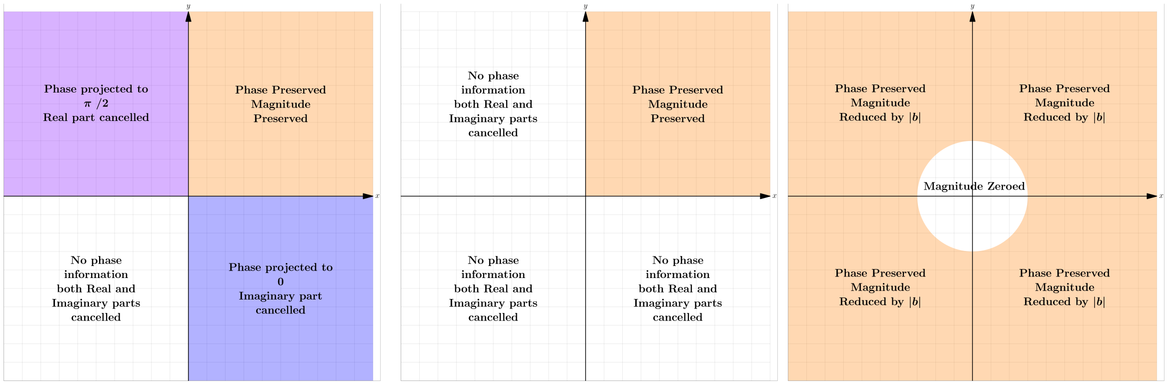


Figure 3.6 Depiction of Complex Standardization in Deep Complex Networks. *At left*, Naive Complex Standardization (division by complex standard deviation) ; *At right*, Complex Standardization (left-multiplication by inverse square root of covariance matrix between $\Re$ and $\Im$). The 250 input complex scalars are at the bottom, with $\Re(v)$ plotted on x (red axis) and $\Im(v)$ plotted on y (green axis). Deeper representations correspond to greater z (blue axis). *The gray ellipse* encloses the input scalars within 1 standard deviation of the mean. *Red ellipses* enclose all scalars within 1 standard deviation of the mean after "standardization". *Blue ellipses* enclose all scalars within 1 standard deviation of the mean after left-multiplying all the scalars by a random 2×2 linear transformation matrix. With the naive standardization, the distribution becomes progressively more elliptical with every layer, eventually collapsing to a line. This ill-conditioning manifests itself as NaNs in the forward pass or backward pass. With the complex standardization, the points' distribution is always successfully re-circularized.

3.6.7 Phase Information Encoding



$\mathbb{C}$ReLU zReLU modReLU

Figure 3.8 Phase information encoding for each of the activation functions tested for the Deep Complex Network. The x-axis represents the real part and the y-axis axis represents the imaginary part ; The figure on the right corresponds to the case where $b < 0$ for modReLU. The radius of the white circle is equal to $|b|$. In case where $b \geq 0$, the whole complex plane would be preserving both phase and magnitude information and the whole plane would have been colored with orange. Different colors represents different encoding of the complex information in the plane. We can see that for both zReLU and modReLU, the complex representation is discriminated into two regions, i.e, the one that preserves the whole complex information (colored in orange) and the one that cancels it (colored in white). However, $\mathbb{C}$ReLU discriminates the complex information into 4 regions where in two of which, phase information is projected and not canceled. This allows $\mathbb{C}$ReLU to discriminate information easier with respect to phase information than the other activation functions. It also allows to halve the variance of the pre-activations at the time of initialization (at beginnning of training process if we assume that pre-activations have a circular distribution around the origin). This is not the case neither for zReLU nor for modReLU as the former divides the variance by 4 at the time of initialization and modReLU preserves the variance of the preactivations if b is initialized to 0 (if modReLU acts as identity at the time of initialization). For both zReLU and modReLU, we can see that phase information may be preserved explicitly through a number of layers when these activation functions are operating in their linear regime, prior to a layer further up in a network where the phase of an input lies in a zero region. $\mathbb{C}$ReLU has more flexibility manipulating phase as it can either set it to zero or $\pi/2$, or even delete the phase information (when both real and imaginary parts are canceled) at a given level of depth in the network.

CHAPTER 4 DEEP COMPLEX SEPARATORS

Recent advances have made it possible to create deep complex-valued neural networks. Despite this progress, many challenging learning tasks have yet to leverage the power of complex representations. Building on recent advances, we propose a new deep complex-valued method for signal retrieval and separation in the frequency domain. As a case study, we perform audio source separation in the Fourier domain. Our new method takes advantage of the convolution theorem which states that the Fourier transform of two convolved signals is the elementwise product of their Fourier transforms. Our novel method is based on a complex-valued version of Feature-Wise Linear Modulation (FiLM) and serves as the keystone of our proposed signal separation architecture. We also introduce a new and explicit phase-aware loss, which is amplitude and shift invariant, taking into account the complex-valued components of the spectrogram. Using the Wall Street Journal Dataset, we compared our phase-aware loss to several others that operate both in the time and frequency domains and demonstrate the effectiveness of our proposed model and loss. Our results highlight the potential of deep complex-valued networks to improve performance on signal retrieval and separation tasks in the frequency domain.

4.1 Introduction

Complex-valued neural networks have been studied since long before the emergence of modern deep learning techniques (Georgiou and Koutsougeras, 1992; Zemel et al., 1995; Kim and Adalı, 2003; Hirose, 2003; Nitta, 2004). Nevertheless, deep complex-valued models have only just started to gain momentum (Reichert and Serre, 2013; Arjovsky et al., 2016; Danihelka et al., 2016; Trabelsi et al., 2017; Jose et al., 2017; Wolter and Yao, 2018a; Choi et al., 2019), with the great majority of models in deep learning still relying on real-valued representations. The motivation for using complex-valued representations for deep learning is twofold : On the one hand, biological nervous systems actively make use of synchronization effects to gate signals between neurons – a mechanism that can be recreated in artificial systems by taking into account phase differences (Reichert and Serre, 2013). On the other hand, complex-valued representations are better suited to certain types of data, particularly those that are naturally expressed in the frequency domain.

Other benefits provided by working with complex-valued inputs in the spectral (frequency) domain are computational. In particular, short-time Fourier transforms (STFTs) can be used to considerably reduce the temporal dimension of the representation for an underlying signal. This is a critical advantage, as training Recurrent Neural Networks (RNNs) or Convolutional Neural Networks (CNNs) on long sequences remains challenging due to unstable gradients and computational requirements of backpropagation through time (BPTT) (Hochreiter, 1991; Bengio et al., 1994). Applying the STFT on the raw signal, on the other hand, is computationally efficient, as in practice it is implemented with the Fast Fourier Transform (FFT) whose computational complexity is $\mathcal{O}(N\,log(N))$.

The aforementioned biological, representational and computational considerations provide compelling motivations for designing learning models for tasks where the complex-valued representation of the input and output data is more desirable than their real-counterpart.

Recent work has provided building blocks for deep complex-valued neural networks (Trabelsi et al., 2017). These building blocks have been shown, in many cases, to avoid numerical problems during training and thereby enable the use of complex-valued representations obtained through deep neural networks. These representations are well-suited for frequency domain signals, as they have the capacity to explicitly encode frequency magnitude and phase components. In particular, models built with these building blocks have excelled at tasks such as automatic music transcription, spectrum prediction (Trabelsi et al., 2017), speech enhancement (Choi et al., 2019), as well as MRI reconstruction (Dedmari et al., 2018).

Recently, Choi et al. (2019) have designed a deep complex U-Net for *speech enhancement*. This task consists of separating clean speech from noise when the noisy speech is given as input. A different but related task is that of separating the speech of multiple speakers into separate signals. This motivates us to build and improve upon the deep complex U-Net for the task of *speech separation*. From this work, our contributions are summarized as follows :

1. We present a novel masking method based on Feature-wise Linear Modulation (FiLM) (Perez et al., 2018) to create multiple separated candidates for each of the signals we aim to retrieve from a mixture of inputs. A signal averaging operation on the candidates is performed in order to increase the robustness of the signal to noise and interference. Before the averaging procedure, a dropout operation is implemented on the signal candidates in order to reduce the amount of interference and noise correlation existing between the different candidates. Our masking method could be seen as performing local ensembling. In the case of audio source separation, we aim to retrieve distinct audio signals associated with each speaker in the input mix. The candidates are averaged in order to obtain the final separated speech for each of the speakers in question. Our experiments demonstrate the efficacy of our proposed masking method, and show its regularizing effect.

2. We propose and explore a new frequency-domain loss taking **explicitly** into account the magnitude and phase of signals. A key characteristic of our loss is that it is amplitude- and shift-invariant. Our comparative analysis related to different phase-aware losses defined in time and frequency domains demonstrates the advantage of our proposed loss.

4.2 Related work

Extended related work sections on complex-valued representation in deep learning and speech separation can be respectively found in sections 2.5 and 2.6 [1]. We give here below a very concise review about recent developments on audio source separation.

4.2.1 Audio Speech Separation Methods

In this section, we discuss some neural speech separation methods that rely on audio information only. Our work falls into this category of methods. To the best of our knowledge, Huang et al. (2014b) were the first to explore the use of deep learning applied to monaural speech separation. Their system is based on a combination of a feed-forward and a recurrent network, that are jointly optimized with a soft masking function. A closely-related work has been concurrently proposed by Du et al. (2014), where a neural network is trained to estimate the log power spectrum of the target speakers.

Hershey et al. (2015) proposed a deep clustering approach to speech separation. The basic idea is to learn high-dimensional embeddings of the mixture signals, that is later exploited to separate the speech targets with standard clustering techniques. A recent attempt to extend deep clustering led to the deep attractor network proposed by Chen et al. (2016). Similarly to deep clustering, high dimensional embeddings are learned, but the network also creates the so-called “attractors" to better cluster time-frequency points dominated by different speakers.

The aforementioned approaches estimate only the magnitude of the STFTs and reconstruct the time-domain signal with the Griffin-Lim algorithm (Griffin and Lim, 1984) or other similar procedures (Sturmel and Daudet, 2006). Similarly to our work, other papers have recently proposed to integrate the phase-information within a speech separation system. The work by Erdogan et al. (2015), for instance, proposes to train a deep neural network with a phase-sensitive loss. Another noteworthy attempt has been described in Wang et al. (2018), where the neural network still estimates the magnitude of the spectrum, but the time-domain speech signals are retrieved by directly integrating the Griffin-Lim reconstruction into the neural layers. Instead of explicitly integrating phase-information, other recent work perform speech separation in the time domain directly, as described in Venkataramani and Smaragdis (2018). Likewise, The TasNet architectures proposed in Luo and Mesgarani (2017) and Luo and Mesgarani (2018) perform speech separation using the mixed time signal as input.

[1] **Important Note :** The goal of this paper is not to provide a state-of-the-art method in the speech separation task. In this work, we are interested in providing a method which improves signal retrieval and separation quality when the input data is complex-valued and represented in the Fourier domain. Recently, state-of-the-art in speech separation has been obtained by using Temporal Convolutional Networks (TCN) (Bai et al., 2018) on the time domain signal (Luo and Mesgarani, 2018) and by combining embedding and clustering methods with TCN on both magnitude spectrogram and time domain signal (Yang et al., 2019b).

The studies by (Lee et al., 2017; Huang et al., 2014b) are more closely related to our work as they address the speech separation problem taking into account phase information. However, this was done without leveraging the recent advances in complex-valued deep learning. The authors in Lee et al. (2017) address the audio-source separation problem using a fully complex-valued deep neural network that learns the nonlinear relationship between an input sound and its distinct sources. Both the activations and weights of the network are complex-valued. For a more comprehensive review of most of these techniques, we refer the readers to Wang and Chen (2018).

4.3 Deep Complex Speech Separation

We detail here a new deep complex architecture to perform speech separation. For this, we rely on the U-Net architecture proposed by Ronneberger et al. (2015) and the complex-valued building blocks proposed by Trabelsi et al. (2017). In our proposed architecture, we incorporated residual connections inside the U-Net blocks and we replaced the complex batch normalization with complex layer normalization, as the model was unable to learn with the former technique and yielded instabilities during training. The details of the complex layer normalization technique and the reasons of its outperformance compared to complex batchnorm are discussed in section 4.3.1. After that, we describe the steps of our novel complex masking method which is based on a complex-valued version of Feature-wise Linear Modulation (FiLM) (Perez et al., 2018), which we designed, and allows to perform local ensembling. Later, in section 4.4, we present our new frequency-domain loss which takes explicitly into account the magnitude and phase of signals.

4.3.1 Complex Layer Normalization

Just as in complex batch normalization, complex layer normalization consists in whitening 2D vectors by left-multiplying the $\mathbf{0}$-centered data $(\boldsymbol{x} - \mathbb{E}[\boldsymbol{x}])$ by the inverse square root of the 2×2 covariance matrix $\boldsymbol{V}$. $\tilde{\boldsymbol{x}} = (\boldsymbol{V})^{-\frac{1}{2}} (\boldsymbol{x} - \mathbb{E}[\boldsymbol{x}])$, where the covariance matrix $\boldsymbol{V}$ is

$$\begin{aligned}\boldsymbol{V} &= \begin{pmatrix} V_{rr} & V_{ri} \\ V_{ir} & V_{ii} \end{pmatrix} \\ &= \begin{pmatrix} \mathrm{Cov}(\Re\{\boldsymbol{x}\}, \Re\{\boldsymbol{x}\}) & \mathrm{Cov}(\Re\{\boldsymbol{x}\}, \Im\{\boldsymbol{x}\}) \\ \mathrm{Cov}(\Im\{\boldsymbol{x}\}, \Re\{\boldsymbol{x}\}) & \mathrm{Cov}(\Im\{\boldsymbol{x}\}, \Im\{\boldsymbol{x}\}) \end{pmatrix}.\end{aligned}$$

Complex layer normalization is distinguished from complex batch normalization by its computation of the mean and covariance statistics over the *layer* features instead of the *batch* instances. This allows us, as in the real-valued version, to avoid estimating batch statistics during training. An intuition for batch normalization's inappropriateness is related to the sparsity, in both time and frequency domains, of speech. This is reflected in the spectrograms. Speech is temporally halting and restarting, and spectrally consists of at most a few simultaneously-held fundamentals and their discrete overtones. Mixing few speakers does not significantly change this property.

In the light of this observation, it stands to reason that statistics computed across a batch's multiple utterance mixtures are almost meaningless. Speakers within and across utterance mixtures are not controlled for volume, nor can their pauses be meaningfully aligned. Batch statistics will therefore be inappropriately driven by the mixture with the most simultaneous speakers, the loudest speaker(s), or the speaker(s) with the "dirtiest" spectrogram. Finally, in the absence of any speech, batch statistics will inappropriately boost background noise to a standardized magnitude.

The above motivates the use of exclusively intra-sample normalization techniques like Layer Normalization for speech data. Batch normalization is more appropriate for natural images, which are spatially dense.

In addition to the fact that intra-sample normalization is more appropriate for speech signals, CLN ensures a more robust normalization of data when the number of feature maps is sufficiently large. In fact, according to the weak law of large numbers, as the sample size increases, the sample statistics approximate their expected values. Therefore, when the number of feature maps far exceeds the number of batch instances, we obtain more robust estimates because they converge, in probability, to the corresponding expected values.

4.3.2 Complex Residual U-Nets

Residual networks (He et al., 2016a) and identity connections (He et al., 2016b) have had a significant impact on image segmentation. These architectural elements have also been combined with U-Nets (Drozdzal et al., 2016) for image segmentation. In our case, we use simple basic complex residual blocks (Figure 4.1) inside each of the U-Net encoding and decoding paths (Figure 4.2). Figure 4.1 (Left) and (Middle) illustrate the basic structure of our Residual U-Net upsampling and downsampling blocks (U_i and D_i) used in Figure 4.2, while Figure 4.1 (Right) illustrates the structure of the complex residual blocks used in Figure 4.1 (Left) and Figure 4.1 (Middle). Each U-Net block begins with an upsampling block (in the decoding U-Net path) or a downsampling block (in the encoding U-Net path). It also contains a block that doubles the number of feature maps (in the encoding path), or halves them (in the decoding path). The upsampling, downsampling, doubling and halving blocks each apply successively a complex layer normalization, a $\mathbb{C}$ReLU and a complex convolution to their inputs. All complex convolutions have kernel size of 3×3 except for the case of a downsampling block, where the convolution layer has a kernel size of 1×1 and a stride of 2×2. In the case of upsampling, we use bilinear interpolation instead of transposed convolution because we found empirically that it yielded better results.

Immediately before and immediately after the doubling / halving blocks, we use $k = 1$ or $k = 2$ residual blocks. We have opted for this residual U-Net block architecture because of memory constraints and because residual connections are believed to perform inference through iterative refinement of representations (Greff et al., 2016; Jastrzebski et al., 2017).

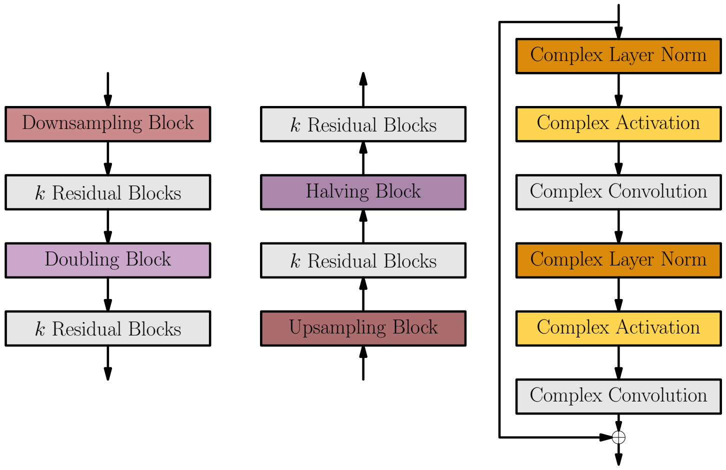


Figure 4.1 The basic structures of our U-Net downsampling block D_i (Left) and our U-Net upsampling block U_i (Middle) used respectively in the encoding and the decoding paths of Figure 4.2. The structure of a basic complex residual block (Right) in each of D_i and U_i.

4.3.3 Complex mask generation

In a environment involving multiple speakers and multiple sources of noise signals, a clean speech signal $s(t)$, of a particular speaker, is corrupted by the environment impulse response $r(t)$. It is also corrupted by an additive noise $a(t)$ (Zhang et al., 2018b). Then, the observed corrupted signal $y(t)$, can be written as : $y(t) = s(t) * r(t) + a(t)$, where $*$ denotes the convolution operator. We can then perform a Fourier Transform on the corrupted signal and leverage the Convolution Theorem along with the linearity property of the Fourier Transform. The Fourier Transform of the clean signal would then be written as :

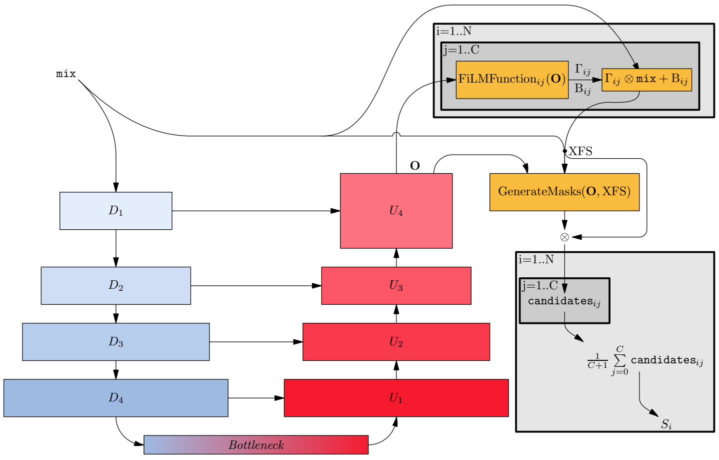
$$\begin{aligned}\mathcal{F}(y(t)) &= \mathcal{F}\left[s(t) * r(t) \quad + a(t)\right] \\ &= \mathcal{F}\left[s(t) * r(t)\right] + \mathcal{F}\left[a(t)\right] \\ &= \mathcal{F}\left[s(t)\right] \odot \mathcal{F}\left[r(t)\right] + \mathcal{F}\left[a(t)\right],\end{aligned} \tag{4.1}$$

where $\mathcal{F}$ is the Fourier Transform and $\otimes$ is the elementwise complex product. We can then express $\mathcal{F}\left[s(t)\right]$ as :

$$\mathcal{F}(s(t)) = \mathcal{F}\left[y(t)\right] \odot \frac{1}{\mathcal{F}\left[r(t)\right]} - \frac{\mathcal{F}\left[a(t)\right]}{\mathcal{F}\left[r(t)\right]}. \tag{4.2}$$

The problem of estimating $\mathcal{F}\left[s(t)\right]$ is reduced to estimating the scaling representation $\frac{1}{\mathcal{F}[r(t)]}$ and the shifting representation $-\frac{\mathcal{F}[a(t)]}{\mathcal{F}[r(t)]}$. For that, FiLM (Perez et al., 2018) can be used.

Featurewise Linear Modulation (FiLM) (Perez et al., 2018) technique has yielded impressive results in visual question answering (VQA). The FiLM approach applies an affine transformation to convolutional feature maps, given the embedding of the question. In our approach, we create



i=1..N
j=1..C
mix
$\mathrm{FiLMFunction}_{ij}(\mathbf{O})$
Γ_{ij}
B_{ij}
$\Gamma_{ij} \otimes \texttt{mix} + \mathrm{B}_{ij}$
XFS
O
$\mathrm{GenerateMasks}(\mathbf{O}, \mathrm{XFS})$
$\otimes$
D_1
D_2
D_3
D_4
U_4
U_3
U_2
U_1
Bottleneck
i=1..N
j=1..C
$\texttt{candidates}_{ij}$
$\frac{1}{C+1}\sum_{j=0}^{C} \texttt{candidates}_{ij}$
S_i

Figure 4.2 The architecture of our Deep Complex Separator that we call Deep Pitcher. It consists of a pipeline containing a U-Net and a Complex FiLMed Masking operator (see Algorithm 1). The Deep Pitcher takes as input the mixed speech signal which is fed to the U-Net. The downsampling blocks of the U-Net are denoted by D_i where in our case $i \in \{1, 2, 3, 4\}$ and the upsampling blocks are denoted by U_i where $i \in \{1, 2, 3, 4\}$. The output of the U-Net along with the input mix are then fed to the Complex FiLMed Masking operator in order to estimate the clean speech for each of the speakers.

multiple transformations of the complex input spectrogram using FiLM. The FiLM parameters are determined from the output of our U-Net. We then generate a complex mask for the original input spectrogram as well as for each of the FiLM-transformed spectrograms. This is accomplished by using a ResNet conditioned on the U-Net output, the spectrogram and its FiLM transformations. Each spectrogram is multiplied by its corresponding complex mask. This leads to multiple candidates for the separated speech of each speaker. The resulting outputs are averaged to produce the final estimated clean speech. This could be interpreted as a local ensembling procedure to estimate the clean speech of the different speakers.

More precisely, given the output of the last upsampling block of the U-Net, we generate scaling matrices Γ_j and shift matrices B_j, $j \in [1, C]$ of the same size as the input mix spectrogram. These parameters operate on the input mix as described by the following equation :

$$\mathrm{input\_transformation}_j = \Gamma_j \otimes \mathrm{inputmix} + \mathrm{B}_j, \tag{4.3}$$

where Γ_j and B_j are functions of the output of the last upsampling block in the U-Net, and $\otimes$ is the elementwise complex product. In our case, we used a simple complex convolution layer with a kernel of size 3 × 3 to generate Γ_j and B_j.

The original input mix and its C scaled and shifted transformations together form $C+1$ representations of the input mix. Given these $C+1$ complex representations, we generate $C+1$ corresponding complex masks, with which the representations are then multiplied. These masks are generated by a sequence of a complex convolution layer which kernel size is 3 × 3 followed by two residual blocks. Once we have performed the complex multiplication of the masks with their respective inputs, $C+1$ separated speech candidates are obtained for a given speaker. This procedure is repeated for the maximum number of speakers that could exist in an input mix.

The main motivation for this process is to increase the separation capability and reduce interference between the separated speakers. Each transformation can focus on a specific pattern in the representation ; Thereafter, as Danihelka et al. (2016) suggest, we can include and exclude candidates in order to keep specific patterns in the speech while removing unwanted ones. Each mask corresponding to a specific input transformation can be seen as a feature of the speaker embedding. Grouped together, the masks generated to retrieve the speech of a given speaker could be interpreted as an embedding identifying the speaker. The complex masking procedure is summarized in Algorithm 1.

Algorithm 1 Complex FiLMed Masking

```
Require: U-Net output :                     O
Require: Nb transformations (XFs) :         $C$
Require: Nb speakers :                      $N$
Require: Input Mix :                        mix
Ensure: Speakers separated speeches :       $S_1, ..., S_N$

function ℂ-FiLMed Masking(O, $C$, $N$, mix)
    for $i \leftarrow 1$ to $N$ do
        $\Gamma_{i1} ... \Gamma_{iC}$, $\mathrm{B}_{i1} ... \mathrm{B}_{iC} \leftarrow$ FilMFunction(O)
    end for
    XFS $\leftarrow$ [ ]
    for $i \leftarrow 1$ to $N$ do
        for $j \leftarrow 1$ to $C$ do
            $\mathrm{XF}_{ij} \leftarrow \Gamma_{ij} \otimes \mathrm{mix} + \mathrm{B}_{ij}$
            $\mathrm{XFS}_i$.append($\mathrm{XF}_{ij}$)
        end for
    end for
    XFS $\leftarrow$ concatenate($\mathrm{XFS}_{11}, ..., \mathrm{XFS}_{NC}$)
    masks $\leftarrow$ GenerateMasks(O, XFS)
    candidates $\leftarrow$ masks $\otimes$ XFS
    cleanspeeches $\leftarrow$ [ ]
    for $i \leftarrow 1$ to $N$ do
        $\mathrm{cleanspeech}_i \leftarrow$ average(candidates[$(C+1) \times (i-1) + 1 : (C+1) \times i$]))
        cleanspeeches.append($\mathrm{cleanspeech}_i$)
    end for
    return cleanspeeches
end function
```

4.4 Normalized Complex-Valued Inner Product Loss

In Choi et al. (2019) a weighted version of the SDR loss proposed in Vincent et al. (2006) is used in order to maximize the similarity between the reference and the estimated signal. The SDR loss proposed by Vincent et al. (2006) is nothing but the real-valued cosine similarity given by the following equation :

$$loss_{sdr}(\mathbf{y}, \mathbf{x}) = \frac{-\sum_i x_i \circ y_i}{||\mathbf{x}|| \cdot ||\mathbf{y}||}, \tag{4.4}$$

where $\circ$ denotes the element-wise real-valued multiplication operation. Both $\mathbf{y}$ and $\mathbf{x}$ are real-valued in the above equation as $\mathbf{y}$ is the target signal in the temporal domain and $\mathbf{x}$ is the estimated signal after performing an inverse STFT on the spectrogram. The phase is then taken **implicitly** into account as the real-valued target signal encodes inherently the phase of the spectrogram. As the task in Choi et al. (2019) is speech enhancement (which is different from ours as we are performing speech separation), the authors used a weighted version of the SDR loss to weight the part of the loss corresponding to the speech signal and also the complementary part corresponding to the noise signal. This weighting is performed according to their respective target energies.

Here, we suggest the use of a loss function which explicitly takes into account both magnitude and phase. This is accomplished by computing the inner product, between the reference signal and its estimate, in the complex plane. In fact computing the inner product in the frequency domain is equivalent to computing the cross correlation in the time domain followed by a weighted average. The inner product in the frequncy domain is then, shift invariant. The complex inner product between 2 signals is given by the following equation :

$$\langle \mathbf{x} | \mathbf{y} \rangle = \sum_j [\Re(x_j)\Re(y_j) + \Im(x_j)\Im(y_j)] + i\,[\Re(x_j)\Im(y_j) - \Im(x_j)\Re(y_j)]. \tag{4.5}$$

If $\mathbf{x}$ and $\mathbf{y}$ are identical, which is equivalent of having $||\mathbf{x}|| = ||\mathbf{y}||$ and $\angle\mathbf{x} = \angle\mathbf{y}$, then, $\langle \mathbf{x} | \mathbf{y} \rangle = ||\mathbf{y}||^2 + 0i$. If $\mathbf{x}$ and $\mathbf{y}$ are parallel, then $\frac{\langle \mathbf{x} | \mathbf{y} \rangle}{||\mathbf{x}|| \cdot ||\mathbf{y}||} = 1 + 0i = 1$. The inner product between the 2 signals normalized by the product of their amplitudes, is then amplitude and shift invariant. We chose a loss that maximizes the real part of that normalized inner product and minimizes the square of its imaginary part. Note that each of the real and imaginary parts of the normalized inner product lies between [-1, 1]. To understand more how the complex inner product is both amplitude and phase aware, how the real part of equation (4.5) is responsible of the amplitude similarity between the reference and estimate signals and how the imaginary part of the same equation is responsible for the phase matching between them, see Appendix 4.7.

Real-valued objective functions expressed in terms of complex variables have been used in optimization (Au, 2007; Sorber et al., 2012). We use a similar approach to express our new similarity loss. In order to approximate both the magnitude and phase in a proper way, we use independent penalization constants for the terms responsible for the amplitude and phase similarities. We define

Table 4.1 Speech separation experiments on two speakers using the standard setup with the Wall Street Journal corpus.
We explore different real and complex-valued model variants and report the test SDR. k is the the number of residual blocks used inside the residual U-Net block (See Figure 1). Start Fmaps is the number of feature maps in the first layer of the encoding path in the U-Net. The Start Fmaps defines the width of each of the successive layers in the model. We respectively double and half the size of the layers in each of the successive downsampling and upsampling stages. The effective number of feature maps for a complex feature map is equal to the number of reported feature maps × 2. This is due to the fact that it has a real and an imaginary part. The number of parameters is expressed in millions. The number of input mixture transformations is also reported. Test SDR scores for different time and spectral domain losses are inserted in the last column.

Model	k	Start Fmaps	Params	Transforms	Loss function	Test SDR
Real U-Net	1	64	8.45	0	$L2_{\text{freq}}$	4.59
Real U-Net	2	64	14.76	0	$L2_{\text{freq}}$	7.92
Complex U-Net	1	32	4.29	0	$L2_{\text{freq}}$	9.61
Complex U-Net	2	32	7.4	0	$L2_{\text{freq}}$	**9.70**
Complex U-Net	2	40	11.55	0	$L2_{\text{freq}}$	**10.30**
Complex U-Net	2	40	11.55	0	$\mathbb{C}$SimLoss	10.21
Complex U-Net	2	40	11.55	0	$L2_{\text{time}}$	9.31
Complex U-Net	2	40	11.55	0	$\cos_{\text{time}}$	9.34
Complex U-Net	2	40	11.57	5	$L2_{\text{freq}}$	10.58
Complex U-Net	2	40	11.57	5	$\mathbb{C}$SimLoss	**10.87**
Complex U-Net	2	40	11.57	5	$L2_{\text{time}}$	10.31
Complex U-Net	2	40	11.57	5	$\cos_{\text{time}}$	10.14
Complex U-Net	2	40	11.61	10	$L2_{\text{freq}}$	10.59
Complex U-Net	2	40	11.61	10	$\mathbb{C}$SimLoss	**10.90**
Complex U-Net	2	40	11.61	10	$L2_{\text{time}}$	10.86
Complex U-Net	2	40	11.61	10	$\cos_{\text{time}}$	10.47
Complex U-Net	2	40	11.67	15	$L2_{\text{freq}}$	**10.93**
Complex U-Net	2	40	11.67	15	$\mathbb{C}$SimLoss	10.91
Complex U-Net	2	40	11.67	15	$L2_{\text{time}}$	10.66
Complex U-Net	2	40	11.67	15	$\cos_{\text{time}}$	10.74

the following similarity loss denoted by $\mathbb{C}$SimLoss as :

$$\mathbb{C}\text{SimLoss}(\mathbf{x}, \mathbf{y}) = -\lambda_{real} \Re\left(\frac{\langle \mathbf{x}|\mathbf{y}\rangle}{||\mathbf{x}|| \cdot ||\mathbf{y}||}\right) + \lambda_{imag} \Im^2\left(\frac{\langle \mathbf{x}|\mathbf{y}\rangle}{||\mathbf{x}|| \cdot ||\mathbf{y}||}\right), \tag{4.6}$$

where λ_{real} and λ_{imag} are penalty constants. We fixed λ_{real} to 1 in all our experiments. We tried different values of $\lambda_{imag} \in \{10^2, 10^3, 10^4\}$. $\lambda_{imag} = 10^4$ worked the best. All the results are reported in Table 4.1 and Table 4.2 for $\mathbb{C}$SimLoss correpond to $\lambda_{imag} = 10^4$.

4.5 Experiments

4.5.1 Data Pre-processing and Training Details

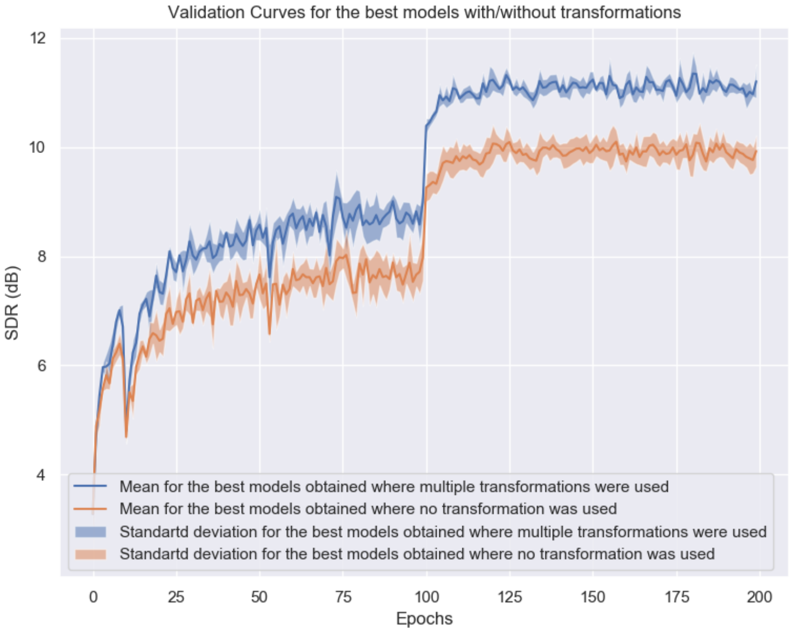


Figure 4.3 Validation curves of models with and without performing multiple input transformations.The plotted curves relate to models reported in Table 4.2. Models with multiple input transformations outperform those without transformations. The former achieved higher SDR scores, on average.

The speech mixtures are generated using the procedure adopted in Erdogan et al. (2015) and Wang et al. (2018). More precisely, the training set consists of 30 hours of two-speaker mixtures that were generated by randomly selecting sentences (uttered by different speakers) from the Wall Street Journal WSJ0 training set called `si_tr_s`. The signals are then mixed with different amplitude factors, leading signal-to-noise ratios (SNR) ranging between 0 dB and 5 dB. Using the same method, we also generated 10 hours of validation set. The test set is composed of 5 hours that were generated similarly using utterances from the different speakers belonging to the WSJ0 development set `si_dt_05`.

Table 4.1 and Table 4.2 contain the results for the experiments conducted using the Wall Street Journal dataset.

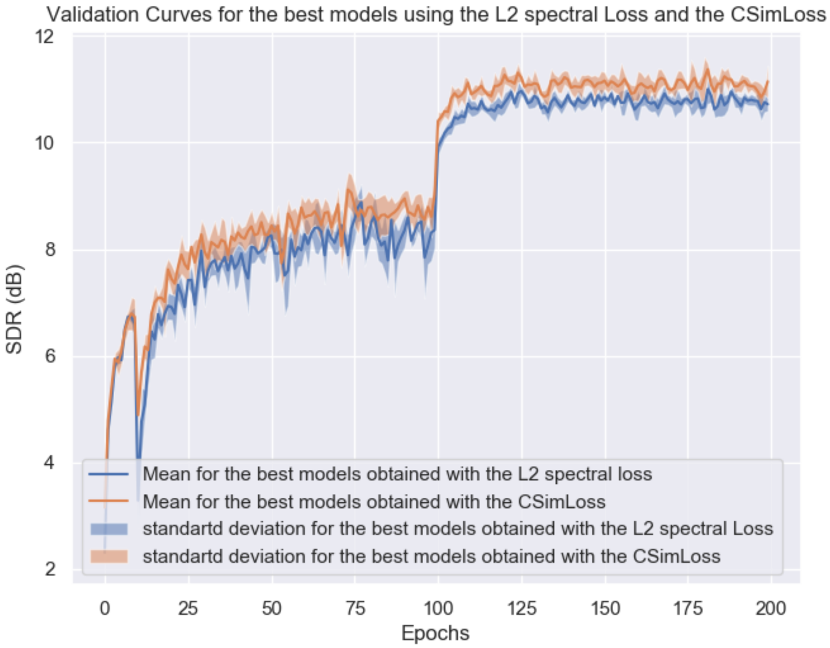


Figure 4.4 Validation curves of the models that yielded the highest SDRs using either the L2 spectral loss or our ℂSimLoss. The Drawn curves are related to models reported in Table 4.2.

All models in Tables 4.1 and 4.2 were trained using the backpropagation algorithm with Stochastic Gradient Descent with Nesterov momentum (Nesterov, 1983) set at 0.9. The gradient norm was clipped to 1. We used the learning rate schedule described in Trabelsi et al. (2017). In order to warm up the model during training, a constant learning rate of 0.01 was fixed for the first 10 epochs. From epoch 10 to 100, the learning rate was increased to 0.1. Later, an annealing of the learning rates by a factor of 10, at epochs, 120 and 150 was performed. We ended up the training at epoch 200. Models in Table 4.1 have been trained using a batch size of 40. Models in Table 4.2 have been trained using a batch size of 24 to fit in the GPU memory. All the models have been trained in parallel using 8 V100 GPUs. For all the tested losses, we used the Permutation Invariant Training criterion knows as PIT (Yu et al., 2017). The PIT criterion allows to take into account all possible assignments between the target signals and the estimated clean speeches. This is done by computing all possible permutations between the targets and the estimated clean speeches. During training, the assignment with the minimal loss is considered for backpropagation. This is due to the fact that for the synthetically mixed input, the order of the target speakers is randomly chosen and it doesn't satisfy a specific criterion. This random order in the target speakers causes the well-known label permutation problem (Hershey et al., 2015; Weng et al., 2015). The PIT criterion allows then to reduce significantly this problem by considering the output-target assignment yielding the minimal training loss. During inference, we assume that the model has learned to produce output that does

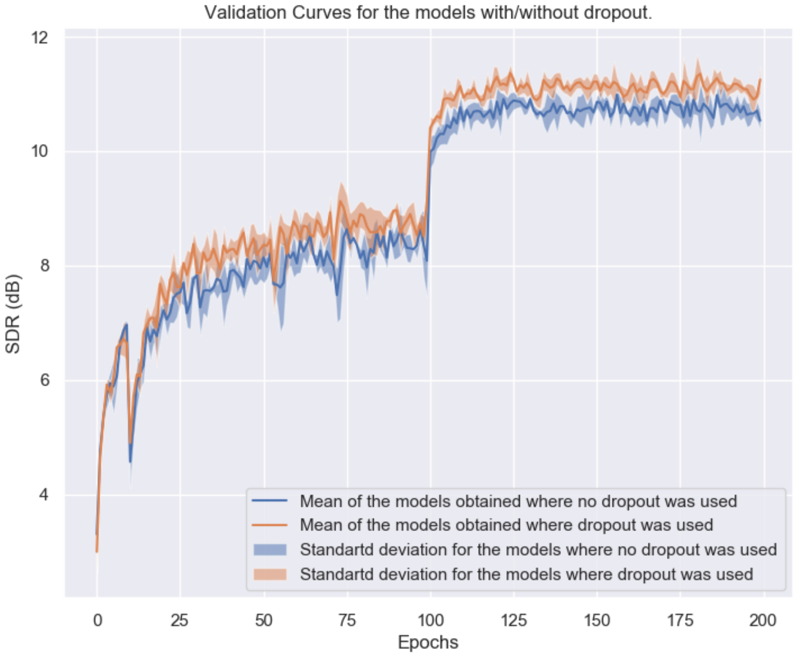


Figure 4.5 Validation curves of the models that yielded the highest SDRs for both cases where dropout on the input mixture transformations was used and where it was not. The Drawn curves are related to models reported in Table 4.2.

not permute speeches. Yu et al. (2017) mention that output-to-speaker assignment may change across time frames. This would have the effect of decreasing the Signal to Noise Ratio (SNR) and the Signal to Distortion Ratio (SDR) as it causes interference of speakers speeches.

At the end of each epoch, we assess our model on the validation set. Of all the models evaluated at the end of each epoch, the one which achieves the highest SDR validation score is then retained for testing.

We tried different configurations combining unitary and standard complex initializations. All of these initializations have been proposed by Trabelsi et al. (2017). It turned out that the best configuration is obtained when using a complex standard initialization for all layers, except for the convolutional layer, generating the FiLM parameters, and the first convolutional layer in the generating mask function which precedes the 2 residual blocks. For the above-mentioned convolutional layers a unitary initialization respecting the He criterion (He et al., 2015) was applied. This is not surprising as a unitary weight matrix $\in \mathbb{C}^{d\times d}$ constitutes a basis of $\mathbb{C}^d$. Therefore any complex-valued vector in $\mathbb{C}^d$, such as those representing the FiLM parameters or the masks, could be generated using a linear combination of the row vectors of that unitary matrix.

Table 4.2 Experiments on two speaker speech separation using the standard setup with the Wall Street Journal corpus. We explore different numbers of input mixture transformations and different dropout rates on the latter using the training losses defined in the spectral domain. The losses in questions are L2$_{\text{freq}}$ and $\mathbb{C}$SimLoss. The number of parameters is expressed in millions. All tested models contain 44 feature maps in the first downsampling layer of the U-Net instead of 40 in Table 4.1. The same number of $k = 2$ residual blocks is used inside the basic structure of the residual U-Net block. SDR scores are shown in the last column.

Params	Transforms	Dropout	Loss function	Test SDR
13.97	0	0	L2$_{\text{freq}}$	9.88
13.99	5	0	L2$_{\text{freq}}$	10.11
14.03	10	0	L2$_{\text{freq}}$	10.91
14.09	15	0	L2$_{\text{freq}}$	9.92
13.97	0	0	$\mathbb{C}$SimLoss	9.87
13.99	5	0	$\mathbb{C}$SimLoss	10.64
14.03	10	0	$\mathbb{C}$SimLoss	**11.05**
14.09	15	0	$\mathbb{C}$SimLoss	10.82
13.99	5	0.1	L2$_{\text{freq}}$	10.54
14.03	10	0.1	L2$_{\text{freq}}$	10.72
14.09	15	0.1	L2$_{\text{freq}}$	10.91
13.99	5	0.1	$\mathbb{C}$SimLoss	10.96
14.03	10	0.1	$\mathbb{C}$SimLoss	**11.34**
14.09	15	0.1	$\mathbb{C}$SimLoss	11.22
13.99	5	0.2	L2$_{\text{freq}}$	10.67
14.03	10	0.2	L2$_{\text{freq}}$	10.90
14.09	15	0.2	L2$_{\text{freq}}$	10.90
13.99	5	0.2	$\mathbb{C}$SimLoss	11.23
14.03	10	0.2	$\mathbb{C}$SimLoss	**11.29**
14.09	15	0.2	$\mathbb{C}$SimLoss	11.23
13.99	5	0.3	L2$_{\text{freq}}$	10.71
14.03	10	0.3	L2$_{\text{freq}}$	10.06
14.09	15	0.3	L2$_{\text{freq}}$	10.91
13.99	5	0.3	$\mathbb{C}$SimLoss	**11.21**
14.03	10	0.3	$\mathbb{C}$SimLoss	11.12
14.09	15	0.3	$\mathbb{C}$SimLoss	11.06
13.99	5	0.4	L2$_{\text{freq}}$	10.72
14.03	10	0.4	L2$_{\text{freq}}$	10.74
14.09	15	0.4	L2$_{\text{freq}}$	10.83
13.99	5	0.4	$\mathbb{C}$SimLoss	11.09
14.03	10	0.4	$\mathbb{C}$SimLoss	11.08
14.09	15	0.4	$\mathbb{C}$SimLoss	**11.12**

In Table 4.1 and 4.2 we experiment with architectures that use different number of mixture transformations. Adding mixture transformations does not significantly increase the number of parameters compared to the size of the whole model. In the case where 15 transformations are adopted, the number of parameters is increased by less than 1% of the total number.

The real-valued models in Table 4.1 contain significantly more parameters than their complex-valued counterparts. However they are, by far, the worst in terms of SDR scores. For example, the complex valued model, with 7.4 M parameters, beats its real-valued counterpart by a margin of 1.82 SDR. Yet the number of parameters of the real model is twice that of the complex model. It is clear that the complex-valued models are more suited for the audio separation task than their real-valued counterpart. Thus, we will no longer focus on real-valued models, but, instead, will concentrate on transformations and losses that are appropriate for complex-valued models.

Three major observations can be inferred from the numbers displayed in Table 4.1 : 1- Wider and deeper models improve the quality of separation in terms of SDRs ; 2- The increase in the number of input transformations has a positive impact on the task of separating audio sources, as additional input transformations achieve higher SDR scores ; 3- For a given number of input transformations, the best results are obtained with losses computed in the spectral domain. For all the experiments reported in Table 4.1, either the $\mathbb{C}$SimLoss or the $\mathrm{L2}_{\mathrm{freq}}$ achieve the highest SDR.

The scores reported in Table 4.1 show that the local ensembling procedure is beneficial to the task of speech separation. This rewarding impact is confirmed in all experiments of Table 4.2 (See also Figure 4.3). As mentioned in section 4.3.3, each mask could be seen as a feature of the speaker embedding and the generated masks together constitute the whole embedding. Performing dropout on the masks might then allow to perform regularization for the retrieval and separation mechanism. Dropping out a mask is equivalent to a dropout of input mixture transformations or clean speech candidates. Since spectral loss functions yielded higher SDRs than their time-domain counterparts, we adopted them to evaluate the effect of applying different dropout rates to the input transformations. Wider and deeper models with Start Fmaps = 44 and k=2 residual blocks are tested in the conducted experiments. Results are reported in Table 4.2.

In the absence of dropout and multiple transformations, we observe from the results displayed in Table 4.2, that wider models are not necessarily more beneficial to the separation task. The SDRs reported in the case of no mixture transformations are 9.88 and 9.87 for the wider model. These SDR scores correspond to the $\mathrm{L2}_{\mathrm{freq}}$ and $\mathbb{C}$SimLoss losses respectively. However, for the narrower models, SDRs of 10.30 and 10.21 were respectively reported for the same losses in Table 4.1. This means that wider models have the potential to overfit. On the other hand, if input transformations are taken into account, a jump in the SDR is observed. When 10 input transformations are introduced, SDR scores of 11.05 and 10.90 are recorded with the $\mathbb{C}$SimLoss and the $\mathrm{L2}_{\mathrm{freq}}$ losses, respectively. Lower SDR performances are recorded when ensembling is implemented with mixtures of 5 and 15 transformations, respectively. This means that the local ensembling procedure is acting as a regularizer. However, a tradeoff in terms of the number of input transformations (and so in terms of clean speech candidates) has to be made as increasing the number of input transformations might worsen the performance of the model and lead to overfitting. This could be explained by the fact that, for a given dropout rate, adding a significant number of transformations can potentially increase the correlation of the noise existing in the different candidates, hence, decreasing the Signal-To-Noise-Ratio of the signal and so, reduces the robustness of the retrieval method to noise and interference.

Dropping out the speech candidates using a small probability rate has a further regularization effect on the wider model. This could be inferred from the results reported in Table 4.2 (See also Figure 4.5). We employed different dropout rates varying from 0 to 0.4. A rate of 0.1 yielded the best result as it caused a jump of SDR score from 11.05 to 11.34. It is important to emphasize again the importance of having a compromise in terms of the number of transformations. For instance, for most of the dropout rates we experimented, a number of 10 mixture transformations yielded the highest SDRs. In all the experiments reported in Table 4.2, the $\mathbb{C}$SimLoss clearly outperformed the L2_{freq} (See Figure 4.4). In fact, regardless of the dropout rate and the number of input transformations employed, for wider models using the L2_{freq} training loss function, the SDR score did not cross the threshold of 10.91 dB. The highest SDR score obtained, when using the L2_{freq} loss function, is 10.93. This value corresponds to a narrower model with 15 input transformations (see Table 4.1).

4.6 Conclusion

In this work, we introduced a new complex-valued framework for signal retrieval and signal separation in the Fourier domain. As a case sudy, we considered audio source separation. We proposed a new masking method based on a complex-valued version of the Feature-wise Linear Modulation (FiLM) model, allowing to perform local ensembling and yielding a beneficial regularization effect. We also proposed a new phase-aware loss taking, explicitly, into account the magnitude and phase of the reference and estimated signals. In our study, phase proved to be an important factor that should be taken into account in order to improve the quality of the separation in terms of SDR. The phase-aware loss improves over other frequency and time-domain losses. Our deep separator draws its power from the compelling properties of complex-valued neural networks and the proposed masking method. Our finding might shed light on the deep complex-valued neural networks' tendency to solve challenging tasks where the data lie in the complex space and where it could be represented in the frequency domain. We view these results as an opportunity to pursue a more systematic investigation of the underpinning of complex-valued representation success. We believe that our proposed method could lead to new research directions where signal separation and signal retrieval are needed.

4.7 Appendix

We show here that solving the two-equation system, assimilating the real part of the inner product, between the two signals **x** and **y**, to the square of the amplitude of **y**, and canceling its imaginary part, amounts to canceling the differences in amplitude and phase between **x** and **y**, respectively (See equation 4.9). For this we will use the following trigonometric properties :

$$\begin{cases} \cos(\theta_x)\cos(\theta_y) & = \frac{1}{2}\cos(\theta_x - \theta_y) \; + \; \frac{1}{2}\cos(\theta_x + \theta_y) \\ \sin(\theta_x)\sin(\theta_y) & = \frac{1}{2}\cos(\theta_x - \theta_y) \; - \; \frac{1}{2}\cos(\theta_x + \theta_y) \\ \cos(\theta_x)\sin(\theta_y) & = \frac{1}{2}\sin(\theta_x + \theta_y) \; - \; \frac{1}{2}\sin(\theta_x - \theta_y) \\ \sin(\theta_x)\cos(\theta_y) & = \frac{1}{2}\sin(\theta_x + \theta_y) \; + \; \frac{1}{2}\sin(\theta_x - \theta_y), \end{cases} \tag{4.7}$$

where $\theta_x, \theta_y \; \in \mathbb{R}$. For simplicity of notation, we will denote a complex-valued target scalar as y and a its estimate as x instead of $\hat{y}$:

$$\begin{aligned} y = |y|e^{i\theta_y} = |y|\left[\cos(\theta_y) + i\sin(\theta_y)\right] \in \mathbb{C} \\ x = |y|e^{i\theta_x} = |x|\left[\cos(\theta_x) + i\sin(\theta_x)\right] \in \mathbb{C}. \end{aligned} \tag{4.8}$$

θ_y and θ_x are the corresponding phases of the reference y and its complex estimate x respectively.

Resolving the system of equations below is equivalent of having both magnitude and phase of the reference and estimation identical **OR** when y is 0.

$$\begin{aligned} & \begin{cases} \Re(x)\Re(y) + \Im(x)\Im(y) - |y|^2 \; = \; 0 \\ \Re(x)\Im(y) - \Re(y)\Im(x) \; = \; 0 \end{cases} \\ \Leftrightarrow & \begin{cases} \Re(x)\Re(y) + \Im(x)\Im(y) = \Re(y)^2 + \Im(y)^2 \\ \Re(x)\Im(y) = \Re(y)\Im(x) \end{cases} \\ \Leftrightarrow & \begin{cases} |x|\cos(\theta_x)\,|y|\cos(\theta_y) + |x|\sin(\theta_x)\,|y|\sin(\theta_y) = |y|^2 \\ |x|\cos(\theta_{\mathrm{x}})\,|y|\sin(\theta_y) = |y|\cos(\theta_y)\,|x|\sin(\theta_x) \end{cases} \\ \Leftrightarrow & \begin{cases} |x|\,|y|\left[\cos(\theta_x)\cos(\theta_y) + \; \sin(\theta_x)\sin(\theta_y)\right] = |y|^2 \\ |x|\,|y|\left[\cos(\theta_x)\sin(\theta_y) - \; \cos(\theta_y)\sin(\theta_x)\right] = 0 \end{cases} \\ \Leftrightarrow & \begin{cases} |y|\left[\,|x|\,(\cos(\theta_x)\cos(\theta_y) + \; \sin(\theta_x)\sin(\theta_y)) \; - \; |y|\,\right] = 0 \\ |x| = 0 \text{ OR } |y| = 0 \text{ OR } \left[\cos(\theta_x)\sin(\theta_y) - \; \cos(\theta_y)\sin(\theta_x)\right] = 0 \end{cases} \\ \Leftrightarrow & \begin{cases} |y| = 0 \text{ OR } |x|\,(\cos(\theta_x)\cos(\theta_y) + \; \sin(\theta_x)\sin(\theta_y)) \; = \; |y| \\ |x| = 0 \text{ OR } |y| = 0 \text{ OR } \cos(\theta_x)\sin(\theta_y) = \; \cos(\theta_y)\sin(\theta_x) \end{cases} \end{aligned}$$

$$\begin{aligned}
&\Leftrightarrow \begin{cases} |y| = 0 \text{ OR } |x| \left(\frac{1}{2}\cos(\theta_x - \theta_y) + \frac{1}{2}\cos(\theta_x + \theta_y) + \frac{1}{2}\cos(\theta_x - \theta_y) - \frac{1}{2}\cos(\theta_x + \theta_y)\right) = |y| \\ |x| = 0 \text{ OR } |y| = 0 \text{ OR } \frac{1}{2}\sin(\theta_x + \theta_y) - \frac{1}{2}\sin(\theta_x - \theta_y) = \frac{1}{2}\sin(\theta_x + \theta_y) + \frac{1}{2}\sin(\theta_x - \theta_y) \end{cases} \\
&\Leftrightarrow \begin{cases} |y| = 0 \text{ OR } |x| \cos(\theta_x - \theta_y) = |y| \\ |x| = 0 \text{ OR } |y| = 0 \text{ OR } -\sin(\theta_x - \theta_y) = \sin(\theta_x - \theta_y) \end{cases} \\
&\Leftrightarrow \begin{cases} |y| = 0 \text{ OR } |x| \cos(\theta_x - \theta_y) = |y| \\ |x| = 0 \text{ OR } |y| = 0 \text{ OR } \theta_x - \theta_y \equiv 0 \pmod{\pi} \end{cases} \\
&\Leftrightarrow \begin{cases} |y| = 0 \text{ OR } |x| \cos(\theta_x - \theta_y) = |y| \\ |x| = 0 \text{ OR } |y| = 0 \text{ OR } -\sin(\theta_x - \theta_y) = \sin(\theta_x - \theta_y) \end{cases} \\
&\Leftrightarrow \begin{cases} |y| = 0 \text{ OR } |x| \cos(k\pi) = |y|, \; k \in \mathbb{Z} \\ |x| = 0 \text{ OR } |y| = 0 \text{ OR } \theta_x = \theta_y + k\pi, \; k \in \mathbb{Z} \end{cases} \\
&\Leftrightarrow \begin{cases} |y| = 0 \text{ OR } |x| \cos(2k'\pi) = |y| \text{ OR } |x| \cos((2k'+1)\pi) = |y| = 0 \, (\text{because} \cos((2k'+1)\pi) = -1) \\ |x| = 0 \text{ OR } |y| = 0 \text{ OR } \theta_x = \theta_y + k'\pi, \; k' \in \mathbb{Z} \end{cases} \\
&\Leftrightarrow \begin{cases} |y| = 0 \text{ OR } |x| = |y| \text{ OR } |x| = |y| = 0 \\ |x| = 0 \text{ OR } |y| = 0 \text{ OR } \theta_x = \theta_y + 2k'\pi, \; k' \in \mathbb{Z} \end{cases} \\
&\Leftrightarrow \begin{cases} |y| = 0 \text{ OR } |x| = |y| \\ \theta_x = \theta_y + 2k'\pi, \; k' \in \mathbb{Z}. \end{cases}
\end{aligned} \tag{4.9}$$

Now, a solution corresponding to a null reference vector $\boldsymbol{y}$ could be problematic as it leads to an infinite number of choices for the estimated signal x. In fact, the authors in Choi et al. (2019) mentioned this issue and chose to work with a cosine similarity-based function in order to learn from noisy-only data. This is why it is more convenient to work with the normalized inner product loss.

CHAPTER 5 A STUDY OF ORTHOGONALITY FOR THE HIDDEN TRANSITION MATRIX IN RECURRENT NEURAL NETWORKS

5.1 Introduction

The vanishing and exploding gradient problem constitutes one of the main challenges faced when training a feed-forward or a recurrent neural network. Unitary transition matrices and soft orthogonality constraints have been adopted successfully to train recurrent neural networks for long-term dependency tasks. However, even though it seems that unitary and orthogonal constraints are ideal solutions to vanishing and exploding gradients, real-world tasks, such as language modeling, might not require such hard constraints on the transition matrices. Restricting the weights to be orthogonal might be too restrictive as it limits the search space of the parameters by constraining it to lie on the Stiefel manifold, the manifold of all orthogonal parameters. Moreover, hard and soft orthogonal constraints are computationally expensive. In this work, we study the usefulness of orthogonality for the hidden transition matrix in recurrent neural networks and we investigate the deviation of the parameters from the Stiefel manifold. We propose a parameterization of the singular values of the hidden transition matrix in order to control the deviation of orthogonality. We find that orthogonal initialization of the weights helps in several real-world tasks as well as carefully constructed synthetic datasets. Our experiments support the thesis that a hard orthogonal constraint is too restrictive and limits the representational power of recurrent networks.

5.2 Related Work and Motivation

5.2.1 Related Work

An extended related work section on the vanishing and exploding gradient problem can be found in 2.4. We detail our motivation for studying orthogonality of hidden transition matrices in RNNs in the next subsection.

5.2.2 Motivation

In this work, our goal is to know whether orthogonality is restrictive or provides advantages in terms of optimization convergence and performance. We focus on the optimization of real valued hidden transition matrices. We set the singular values within a configurable margin about the Stiefel manifold. We suspect that a hard constraint of orthogonality limits the model's representational power, worsens its performance, and slows down optimization convergence speed. By combining Singular Value Decomposition of the hidden transition matrix and the geodesic gradient descent algorithm, we can explicitly express the amount of deviation from orthogonality by setting a learnable margin for the singular values and optimize it. This allows us to observe and analyze cases where deviating from orthogonality could be either beneficial or detrimental.

5.2.3 Vanishing and Exploding Gradients in Recurrent Networks

As it can be seen in Appendix B, where the problem of the exploding and the vanishing variance of both the propagated and backpropagated signals can be illustrated via the composition of activations (equations (B.8) and (B.24)) and the derivative's chain rule (equations (B.5) and (B.20)) respectively, the issue of vanishing and exploding gradients in a recurrent neural network can be illustrated by expressing the gradient back-propagation chain through time steps.

A recurrent neural network with N time steps (can be also denoted by T instead of N) has activations :

$$\mathbf{h}_t = f(\widetilde{\mathbf{h}}_t) = f(\mathbf{W}_h\,\mathbf{h}_{t-1} + \mathbf{W}_x\,\mathbf{x}_t + \mathbf{b}), \quad t \in \{1, \cdots, N\}, \tag{5.1}$$

where f is an element-wise nonlinear activation function. $\widetilde{\mathbf{h}}_t \in \mathbb{R}^{N_h}$ is hidden pre-activation. $\mathbf{W}_h \in \mathbb{R}^{N_h \times N_h}$ is the hidden transition weight matrix propagating hidden information through all time steps. N_h denotes the number of hidden dimensions. $\mathbf{W}_x \in \mathbb{R}^{N_x \times N_h}$ is the input weight matrix projecting the input information to the hidden space. N_x is the number of input dimensions. $\mathbf{W}_x$ is also shared through all time steps. $\mathbf{b}$ is the shared bias through all time steps and it is a vector in $\mathbb{R}^{N_h}$.

We can see that for some loss function L_N computed after propagating the hidden information for all N time steps, the derivative with respect to hidden transition matrix $\mathbf{W}_h$ by backpropgating the loss signal to a given time step t is given by :

$$\frac{\partial L_N}{\partial \mathbf{W}_h} = \frac{\partial L_N}{\partial \mathbf{h}_t}\frac{\partial \mathbf{h}_t}{\partial \mathbf{W}_h}. \tag{5.2}$$

The partial derivatives for $\mathbf{h}_t$ with respect to $\mathbf{W}_h$ can be decomposed as follows :

$$\begin{aligned} \frac{\partial \mathbf{h}_t}{\partial \mathbf{W}_h} &= \frac{\partial \mathbf{h}_t}{\partial \widetilde{\mathbf{h}}_t}\frac{\partial \widetilde{\mathbf{h}}_t}{\partial \mathbf{h}_{t-1}}\frac{\partial \mathbf{h}_{t-1}}{\partial \mathbf{W}_h} \\ &= \mathbf{D}_t\mathbf{W}_h\frac{\partial \mathbf{h}_{t-1}}{\partial \mathbf{W}_h} \quad \Rightarrow \quad \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} = \mathbf{D}_t\mathbf{W}_h, \end{aligned} \tag{5.3}$$

where $\mathbf{D}_t$ is the Jacobian corresponding to the elementwise activation function, containing partial derivatives of the hidden units at time step t with respect to the pre-activation inputs. Typically, $\mathbf{D}_t$ is diagonal as the activation function is elementwise. Following the above, the gradient of the loss at the last time step N backpropgated to the 1^{st} time step can be fully decomposed into a recursive chain of matrix products :

$$\begin{aligned}
\frac{\partial L_N}{\partial \mathbf{W}_h} &= \frac{\partial L_N}{\partial \mathbf{h}_N} \frac{\partial \mathbf{h}_N}{\partial \mathbf{h}_{N-1}} \frac{\partial \mathbf{h}_{N-1}}{\partial \mathbf{h}_{N-2}} \cdots \frac{\partial \mathbf{h}_2}{\partial \mathbf{h}_1} \frac{\partial \mathbf{h}_1}{\partial \mathbf{W}_h} \\
&= \frac{\partial L_N}{\partial \mathbf{h}_N} \left[\prod_{i=0}^{N-2} \frac{\partial \mathbf{h}_{N-i}}{\partial \mathbf{h}_{N-(i+1)}} \right] \frac{\partial \mathbf{h}_1}{\partial \mathbf{W}_h} \\
&= \frac{\partial L_N}{\partial \mathbf{h}_N} \left[\prod_{i=0}^{N-2} \mathbf{D}_{N-i} \mathbf{W}_h \right] \frac{\partial \mathbf{h}_1}{\partial \mathbf{W}_h}.
\end{aligned} \tag{5.4}$$

In Pascanu et al. (2013), it is shown that the L_2 norm of $\dfrac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}}$ is bounded by the product of the norms of the nonlinearity's Jacobian and transition matrix at time t :

$$\begin{gathered}
\left|\left| \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} \right|\right| \leq ||\mathbf{D}_t|| \; ||\mathbf{W}_h|| \leq \lambda_{\mathbf{D}_t} \, \lambda_{\mathbf{W}_h} = \eta_t, \\
\lambda_{\mathbf{D}_t}, \lambda_{\mathbf{W}_h} \in \mathbb{R}.
\end{gathered} \tag{5.5}$$

where $\lambda_{\mathbf{D}_t}$ and $\lambda_{\mathbf{W}_t}$ are respectively the largest singular values of the nonlinearity's Jacobian $\mathbf{D}_t$ and the hidden transition matrix $\mathbf{W}_h$, i.e, their respective spectral norms.

It is extremely important to notice here that $\mathbf{W}_x$ **does suffer** from the exploding and vanishing gradient problem. A new input $\mathbf{x}_t$ is fed to the recurrent network and processed by $\mathbf{W}_x$ at each time step. This would allow us to compute the gradient of the loss with respect to $\mathbf{W}_x$ in terms of the sum of the gradients of the loss with respect to the hidden representation corresponding to each time step. The gradient of the loss function L_N with respect to $\mathbf{W}_x$ is given by :

$$\begin{aligned}
\frac{\partial L_N}{\partial \mathbf{W}_x} &= \sum_{i=0}^{N-2} \frac{\partial L_N}{\partial \mathbf{h}_{N-i}} \frac{\partial \mathbf{h}_{N-i}}{\partial \widetilde{\mathbf{h}}_{N-i}} \frac{\partial \widetilde{\mathbf{h}}_{N-i}}{\partial \mathbf{W}_x} = \sum_{i=0}^{N-2} \frac{\partial L}{\partial \mathbf{h}_{N-i}} \mathbf{D}_{N-i} \, \mathbf{x}_{N-i} \\
&= \sum_{i=0}^{N-2} \frac{\partial L_N}{\partial \mathbf{h}_N} \frac{\partial \mathbf{h}_N}{\partial \mathbf{h}_{N-1}} \frac{\partial \mathbf{h}_{N-1}}{\partial \mathbf{h}_{N-2}} \cdots \frac{\partial \mathbf{h}_{N+1-i}}{\partial \mathbf{h}_{N-i}} \mathbf{D}_{N-i} \, \mathbf{x}_{N-i} \\
&= \sum_{i=0}^{N-2} \frac{\partial L_N}{\partial \mathbf{h}_N} \left[\prod_{j=0}^{i-1} \frac{\partial \mathbf{h}_{N-j}}{\partial \mathbf{h}_{N-(j+1)}} \right] \mathbf{D}_{N-i} \, \mathbf{x}_{N-i} \\
&= \sum_{i=0}^{N-2} \frac{\partial L_N}{\partial \mathbf{h}_N} \left[\prod_{j=0}^{i-1} \mathbf{D}_{N-j} \, \mathbf{W}_h \right] \mathbf{D}_{N-i} \, \mathbf{x}_{N-i}.
\end{aligned} \tag{5.6}$$

We clearly see in equation (5.6) that $\frac{\partial L_N}{\partial \mathbf{W}_x}$ is expressed in terms of a sum of products. The same form of product is expressed in equation (5.4). Both products are expressed in terms of $\mathbf{W}_h$ and $\mathbf{D}_t$. It is very important to mention that the recurrence in the expression of $\mathbf{h}_t$ is the reason why the gradient is expressed in terms of a product, which causes it to vanish or to explode. The vanishing and exploding gradient problem might be less pronounced in equation (5.6) as the product is expressed inside a sum. It is also important to mention that for some tasks, like language modeling, a loss is computed at each time step instead of only computing one loss at the very end of the computation. In such case the gradient of each time step loss with respect to $\mathbf{W}_h$ is summed in order to obtain the gradient of the overall loss with respect to $\mathbf{W}_h$. The same is done to obtain the gradient with respect to $\mathbf{W}_x$. In this case, equations (5.4) and (5.6) would involve a sum over all time steps losses in the following way :

$$\frac{\partial L}{\partial \mathbf{W}_h} = \left[\sum_{i=0}^{N-2} \frac{\partial L_{N-i}}{\partial \mathbf{h}_{N-i}} \left(\prod_{j=0}^{i} \mathbf{D}_{N-j} \mathbf{W}_h \right) \right] \frac{\partial \mathbf{h}_1}{\partial \mathbf{W}_h}. \tag{5.7}$$

$$\frac{\partial L}{\partial \mathbf{W}_x} = \sum_{i=0}^{N-2} \frac{\partial L_{N-i}}{\partial \mathbf{h}_{N-i}} \sum_{j=0}^{i} \left[\prod_{k=0}^{j-1} \mathbf{D}_{N-k}\, \mathbf{W}_h \right] \mathbf{D}_{N-j}\, \mathbf{x}_{N-j}. \tag{5.8}$$

The gradient would be computed without problems for the first few terms involved in the sum expressed in (5.6), (5.7) and (5.8). These terms would be contributing the most in $\frac{\partial L}{\partial \mathbf{W}_x}$ if we have a vanishing gradient. Heuristics such as truncated backpropagation through time (TBPTT) (Williams and Peng, 1990) are generally deployed in order to cope with this issue. The heuristic consists of dividing the time dimension into different time slots where at each slot, the product expressed in equations (5.4), (5.6), (5.7) and (5.8) does not involve the terms outside the slot. Such a heuristic in practice is very efficient as it allows to avoid exploding and vanishing gradients. However, the estimation of the gradient is biased. In fact, for a given time slot, we might want to take into account hidden representations outside it as their contribution to the loss might be beneficial for the gradient estimation.

We can make an analogy with feed-forward networks for the processing of $\mathbf{x}_t$ as it is similar to the processing of an input information in a shallow feed-forward network with weight $\mathbf{W}_x$. The hidden signal $\mathbf{h}_t$ however, has been processed by the same hidden transition matrix $\mathbf{W}_h$ from time 1 to time $t-1$. If $\mathbf{W}_h$ does not preserve the norm, and if its spectral norm, i.e its maximal singular value, is less than 1, the hidden information would vanish through time. This will cause the loss of the previously processed hidden information when propagating the forward signal. The network would then mostly rely on the encoded information in $\mathbf{W}_x\, \mathbf{x}_t$. This is not desirable if the task at hand requires long term dependencies and if $\mathbf{x}_t$ is not inherently encoding past information. If the spectral norm of $\mathbf{W}_h$ is greater than 1, $||\mathbf{W}_h\, \mathbf{h}_{t-1}||$ would explode and this would cause the impossibility to train the neural network.

5.3 Deviation from Orthogonality Via Singular Values Parametrization

The amount of deviation from the space of orthogonal parameters can be controlled by parameterizing the singular values of the hidden transition matrix $\mathbf{W}_h$. When the singular values are close to 1, we ensure that the hidden transition matrix is close to the Stiefel Manifold, and so, close to the space of norm preserving transformations. Another way to ensure that the hidden transition matrix is not far from the space of orthogonal transformations is to perform a soft orthogonality constraint of the form $\lambda||\mathbf{W}_h^T\mathbf{W}_h - \mathbf{I}||^2$. As mentioned already in section 5.2, this form of soft orthogonality constraint has been already applied by Henaff et al. (2016).

As our goal is to perform an analysis of orthogonality by explicitly expressing the amount of contraction and expansion at each hidden dimension when $\mathbf{W}_h$ is applied on $\mathbf{h}_{t-1}$ (See equation (5.1)), we perform a singular value decomposition of $\mathbf{W}_h$, which consists of expressing $\mathbf{W}_h$ in terms of orthogonal basis matrices $\mathbf{U}$ and $\mathbf{V}$ with a diagonal spectral matrix $\mathbf{S}$ containing its singular values. We would then have :

$$\mathbf{W}_h = \mathbf{U}\mathbf{S}\mathbf{V}^T. \tag{5.9}$$

For each hidden dimension $i \in \{1, 2, \cdots N_h\}$, $s_i \in \{\text{diag}(\mathbf{S})\}$ controls the amount of contraction, expansion or norm preservation for the i^{th} dimension when $\mathbf{W}_h$ is applied. Contraction, expansion or norm preservation correspond respectively to $s_i > 1$, $s_i < 1$ and $s_i = 1$. If $s_i = 1 \, \forall \, i \in \{1, 2, \cdots N_h\}$, $\mathbf{W}_h$ would preserve the norms of $\mathbf{h}_{t-1}$, i.e, $||\mathbf{W}_h \; \mathbf{h}_{t-1}|| = ||\mathbf{h}_{t-1}||$. In this case $\mathbf{W}_h$ is orthogonal.

Whether $\mathbf{W}_h$ deviates from orthogonality or not, the bases $\mathbf{U}$ and $\mathbf{V}$ must be kept orthogonal during training. This is achieved via the Geodesic Gradient Descent algorithm. The Geodesic Gradient Descent allows one to optimize along the set of weights satisfying $\mathbf{U}^T\mathbf{U} = \mathbf{U}\mathbf{U}^T = \mathbf{I}$ and $\mathbf{V}^T\mathbf{V} = \mathbf{V}\mathbf{V}^T = \mathbf{I}$, i.e, along the Stiefel manifold. For an orthogonally-initialized matrix $\mathbf{W}$, the Cayley transformation of the update step onto the Stiefel manifold of orthogonal matrices is performed. This is done in Nishimori (2005) and Tagare (2011). Given an orthogonally-initialized parameter matrix $\mathbf{W}$ and its Jacobian $\mathbf{G}$ with respect to the objective function, we create a skew symmetric matrix $\mathbf{A}$ depending on $\mathbf{W}$ and $\mathbf{G}$ and project $\mathbf{W}$ by the Cayley transform of $\mathbf{A}$. The result obtained by The projection of $\mathbf{W}$ with the Cayley transform of $\mathbf{A}$ is orthogonal as both matrices involved in the product are orthogonal. More precisely. the updated $\mathbf{W}$ is obtained as follows :

$$\begin{aligned} &\mathbf{A} = \mathbf{G}\mathbf{W}^T - \mathbf{W}\mathbf{G}^T \\ &\mathbf{W}_{new} = (\mathbf{I} + \frac{\eta}{2}\mathbf{A})^{-1}(\mathbf{I} - \frac{\eta}{2}\mathbf{A})\mathbf{W}, \end{aligned} \tag{5.10}$$

where η is the learning rate.

In order to allow deviation from orthogonality, we employ a sigmoidal parameterization of the singular values allowing us to perform a hard constraint on the maximum and minimum amount of deviation from the Stiefel Manifold. Specifically, we define a margin m such as the singular values lie in $[1 - m, \; 1 + m]$. This is achieved in the following way :

$$s_i = 2m(\sigma(p_i) - 0.5) + 1, \qquad s_i \in \{\text{diag}(\mathbf{S})\}, \; m \in [0, \; 1]. \tag{5.11}$$

The underlying parameters p_i are updated freely via stochastic gradient descent. The parameterization of a singular value, expressed in equation (5.11) has an effect on the gradient of a given loss

L with respect to p_i. In fact :

$$\frac{dL}{dp_i} = \frac{dL}{ds_i}\frac{ds_i}{dp_i} = \frac{dL}{ds_i}\frac{d\sigma(p_i)}{dp_i}2m. \tag{5.12}$$

This means for example that the effective learning rate for the singular values is multiplied by $2m$. Consequently, we adjust the learning rate for the singular values to be independent of the margin by dividing it by $2m$.

5.4 Experiments

In this section, we compare the convergence speed and performances of recurrent models relying on orthogonal hidden transition matrices to the ones where the hidden transition matrix deviates from the orthogonality space. As discussed in section 5.3, deviation from orthogonality is done by either setting a spectral margin as described in equation (5.11) or by applying the same form of soft orthogonality than Henaff et al. (2016). We observe that orthogonal initialization is always helpful as mentioned in Henaff et al. (2016) and Saxe et al. (2013). We also observe that deviation from orthogonality helps converging faster while strict orthogonality stablizes gradient computation but slows down the convergence speed.

5.4.1 Datasets and Learning Tasks

The content of this section is the same as the beginning of sections 3 and 3.1 in Vorontsov et al. (2017). It contains fine details of the different tasks, their corresponding data pre-processing as well as the optimization algorithm and the hyperparameters used by the authors.

In order to perform our analysis, we performed five different tasks. To test the memorization capability of the recurrent network and its long-term dependency encoding robustness, we relied on the copy and adding tasks. These are toy synthetic tasks introduced in (Hochreiter and Schmidhuber, 1997). After that, We used more realistic, non-synthetic, data that require from the recurrent network to capture both short- and long-range dependencies. We first performed digit classification for sequential and permuted MNIST vectors (Le et al., 2015; LeCun et al., 1998), then we performed a character-level language modeling task using the Penn Treebank dataset (Marcus et al., 1993).

The copy task consists of an input sequence that must be remembered by the network, followed by a series of blank inputs terminated by a delimiter that denotes the point at which the network must begin to output a copy of the initial sequence. We use an input sequence of $T + 20$ elements that begins with a sub-sequence of 10 elements to copy, each containing a symbol $a_i \in \{a_1, ..., a_p\}$ out of $p = 8$ possible symbols. This sub-sequence is followed by $T - 1$ elements of the blank category a_0 which is terminated at step T by a delimiter symbol a_{p+1} and 10 more elements of the blank category. The network must learn to remember the initial 10 element sequence for T time steps and output it after receiving the delimiter symbol.

The goal of the adding task is to add two numbers together after a long delay. Each number is randomly picked at a unique position in a sequence of length T. The sequence is composed of T values sampled from a uniform distribution in the range $[0, 1)$, with each value paired with an indicator value that identifies the value as one of the two numbers to remember (marked 1) or as a value to ignore (marked 0). The two numbers are positioned randomly in the sequence, the first in the range $[0, \frac{T}{2} - 1]$ and the second in the range $[\frac{T}{2}, T - 1]$, where 0 marks the first element. The network must learn to identify and remember the two numbers and output their sum.

In the sequential MNIST task from Le et al. (2015), MNIST digits are flattened into vectors that can be read sequentially by a recurrent neural network. The goal is to classify the digit based on the sequential input of pixels at the last time step. The simple variant of this task is with a simple flattening of the image matrices ; the harder variant of this task includes a random permutation of the pixels in the input vector that is determined once for an experiment. The latter formulation introduces longer distance dependencies between pixels that must be interpreted by the classification model.

The English Penn Treebank (PTB) dataset from Marcus et al. (1993) is an annotated corpus of English sentences, commonly used for benchmarking language models. We employ a sequential character prediction task : given a sentence, a recurrent neural network must predict the next character at each step, from left to right. We use input sequences of variable length, with each sequence containing one sentence. We model 49 characters including lowercase letters (all strings are in lowercase), numbers, common punctuation, and an unknown character placeholder. We use two subsets of the data in our experiments : in the first, we first use 23% of the data with strings with up to 75 characters and in the second we include over 99% of the dataset, picking strings with up to 300 characters.

In all experiments, we employed RMSprop (Tieleman and Hinton, 2012) when not using geodesic gradient descent. We used minibatches of size 50 and for generated data (the copy and adding tasks), we assumed an epoch length of 100 minibatches. We cautiously introduced gradient clipping at magnitude 100 (unless stated otherwise) in all of our RNN experiments although it may not be required and we consistently applied a small weight decay of 0.0001. Unless otherwise specified, we trained all simple recurrent neural networks with the hidden to hidden matrix factorization as in (5.9) using geodesic gradient descent on the bases (learning rate 10^{-6}) and RMSprop on the other parameters (learning rate 0.0001), using a tanh transition nonlinearity, and clipping gradients of 100 magnitude. The neural network code was built on the Theano framework (Team et al., 2016) [1]. When parameterizing a matrix in factorized form, we apply the weight decay on the composite matrix rather than on the factors in order to be consistent across experiments. For MNIST and

[1] Source code for the model and experiments located at `https://github.com/veugene/spectre_release`

PTB, hyperparameter selection and early stopping were performed targeting the best validation set accuracy, with results reported on the test set.

5.4.2 Empirical Study on Orthogonality for The Copy and Adding Tasks

For different sequence lengths T of the copy and adding tasks (we can also denote the sequence length by N as described in equation (5.1)), we trained a factorized RNN with 128 hidden units and tested a wide range of spectral margins m. For the copy task, we used a simple recurrent neural network where the expression of hidden representation at a given time t is given by equation (5.1). The nonlinearity f used for the copy task is the identity function. We also investigated the use of nonlinearities.

For sequence lengths of 200, 500 and 1000, we can see from Figure 5.1 that deviating from orthogonality helps converging faster as for values of $m = 0.1$ and $m = 1$, the models converge to an optimal solution before 50 epochs. Recall that singular values lie in $[1 - m,\ 1 + m]$ and that $m \in [0, 1]$ An interesting phenomenon occurs for a much longer sequence length. For $T = 10000$, the only model which succeeded to resolve the copy task after 300 epochs is the one corresponding to a margin $m = 10^{-3}$ while all other models failed including the one corresponding to an orthogonal hidden transition matrix ($m = 0$) and the one with the highest margin ($m = 1$). This means that a sufficient condition to converge faster to an optimal solution is to deviate a little bit from the space of orthogonal weights while orthogonality does not ensure a faster convergence for long-term dependency tasks.

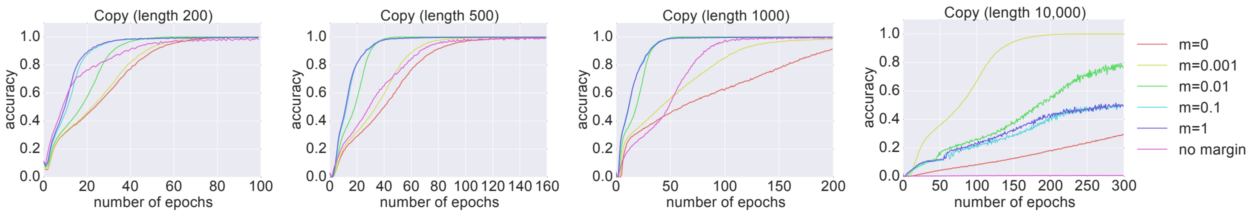


Figure 5.1 (Taken from Vorontsov et al. (2017)). Accuracy curves on the copy task for different sequence lengths given various spectral margins. Convergence speed increases with margin size ; however, large margin sizes are ineffective at longer sequence lengths (N=T=10000, right).

We found that nonlinearities such as tanh was detrimental to solve the copy task. Using tanh in a short sequence length ($T = 100$) copy task requires the same form of orthogonality constraint used in Henaff et al. (2016) and also thousands of training epochs to converge. It is worth mentioning that Henaff et al. (2016) present a solution mechanism for the copy task that does not use nonlinearities in an RNN. To examine more closely the use of nonlinearities in RNNs, we used the parametric leaky ReLU activation function (PReLU) which introduces a learnable slope α for negative valued inputs x, producing $f(x) = \max(x, 0) + \alpha \min(x, 0)$ (He et al., 2015). We first experimented with a fixed value of α where the latter was not learnable. We tested 3 different values of a fixed α, i.e, $\alpha \in \{0.5, 0.7, 1\}$. Setting the slope α to 1 makes the PReLU equivalent to an identity activation function. We set the spectral margin m to 0.3. We found that only the model corresponding to $\alpha = 1$ solved the $T = 1000$ length copy task. We also experimented with a trainable slope α, initialized to 0.7 and found that it converges to 0.96.

These observations lead us to believe that a linear regime of the model (combination of transformations and activations), or a regime close to linearity, is necessary to converge quickly in a task requiring the memorization of information in the medium and long term. Nonlinear activations encourage the model to forget about processed information which is undesirable in long-term tasks. This being said, a small deviation from the linear regime does not harm the performance of the model. This is similar to what has been observed for the orthogonality where small deviation from it allowed a faster convergence to the optimal solution.

For the adding task, we trained the factorized RNN on $T = 1000$ length sequences and used ReLU as an activation function for the hidden representation. The mean squared error (MSE) is shown for different spectral margins in Figure 5.2. A completely different observation is made for the adding task. It can be seen from Figure 5.2 that the loss function for the adding task starts to decrease only for margins bigger or equal to 1 (for $m \in \{10, 100\}$ we applied ReLU on s_i so that it doesn't get negative values). The only models which loss function failed to decrease below the threshold corresponding to outputing the same constant at all time steps are the ones related to strict orthogonality ($m = 0$) and where the singular values are not bound (no margin, i.e, equation (5.11) was not applied on the singular values. Instead, a simple ReLU was applied on p_i)).

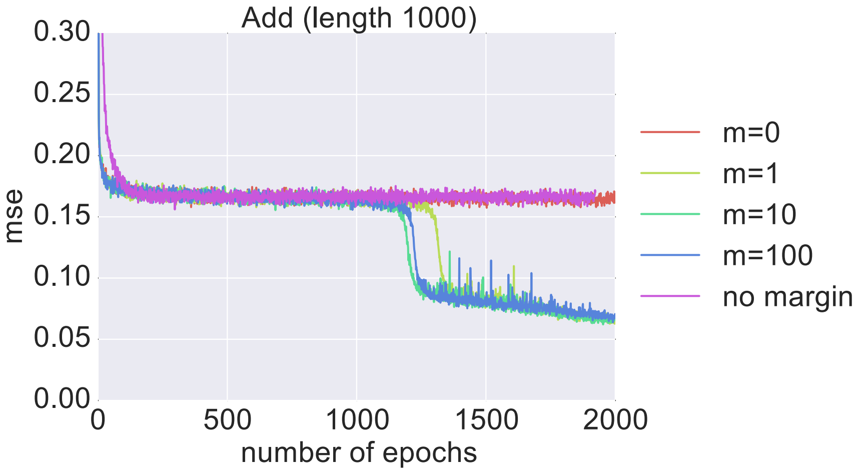


Figure 5.2 (Taken from Vorontsov et al. (2017)). Mean squared error (MSE) curves on the adding task for different spectral margins m. A trivial solution of always outputting the same number has an expected baseline MSE of 0.167.

This is another confirmation of the fact that orthogonality of the transition hidden matrix is not necessarily what we need to converge faster to an optimum. Certainly, orthogonality provides stability in terms of gradient computation of the loss function with respect to the activations and parameters. However, deviation from the orthogonal space of weights is needed in both the copy and adding tasks in order to converge in a faster pace.

5.4.3 Empirical Study on MNIST and PTB

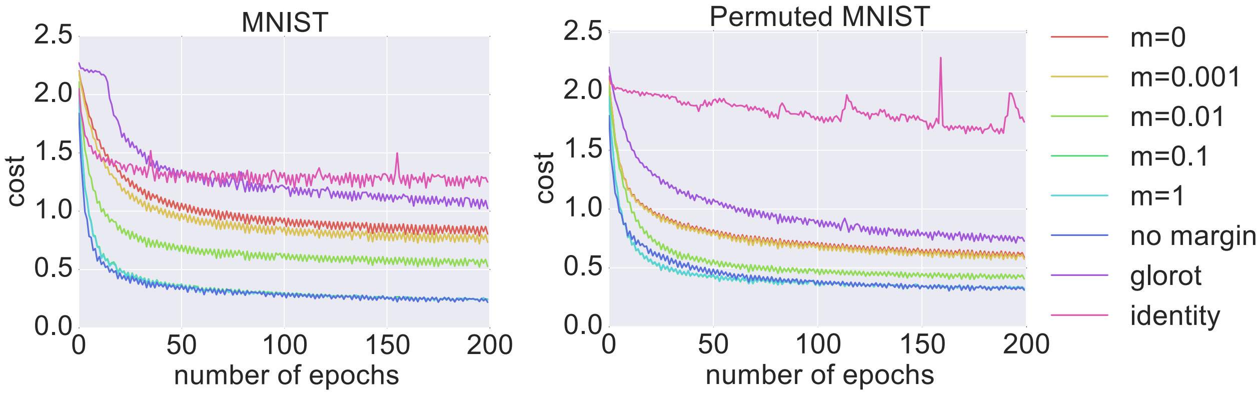


Figure 5.3 (Taken from Vorontsov et al. (2017)). Loss curves for different factorized RNN parameterizations on the sequential MNIST task (left) and the permuted sequential MNIST task (right). The spectral margin is denoted by m ; models with no margin have singular values that are directly optimized with no constraints ; Glorot refers to a factorized RNN with no margin that is initialized with Glorot normal initialization. Identity refers to the same, with identity initialization.

For both sequential and permuted MNIST experiments, loss curves and accuracies are shown in Figure 5.3 and Table 5.1 respectively. We trained a factorized RNN with 128 hidden units for 120 epochs. The tanh function was used as an activation for the hidden representation. For comparison purposes, we also trained an LSTM with 128 hidden units with peephole connections on both tasks. We trained the LSTM for 150 epochs for both sequential and permuted MNIST. The parameters of the LSTM were orthogonally initialized and the forget gate bias was initialized to the unit vector **1** in order to take into account long-term dependencies. Both the factorized RNN and the LSTM were trained with RMSprop using a learning rate of 10^{-4}. We performed gradient clipping for both the factorized RNN and the LSTM. The gradients magnitudes were clipped to 1.

We can observe from the curves drawn in Figure 5.3 and the numbers in Table 5.1 that for relatively higher values of the margin, i.e, $m \in \{0.1, 1\}$, the factorized RNN converged faster and yielded better performances in terms of accuracy. A margin of $m = 0.1$ performed the best on both sequential and permuted MNIST. This confirms again our observation in the copy and adding task where strict

Table 5.1 (Taken from Vorontsov et al. (2017)). Performance on MNIST and PTB for different spectral margins and initializations. Evaluated on classification of sequential MNIST (MNIST) and permuted sequential MNIST (pMNIST) ; character prediction on PTB sentences of up to 75 characters (PTBc-75) and up to 300 characters (PTBc-300).

		MNIST	pMNIST	PTBc-75		PTBc-300	
margin	initialization	accuracy	accuracy	bpc	accuracy	bpc	accuracy
0	orthogonal	77.18	83.56	2.16	55.31	2.20	54.88
0.001	orthogonal	79.26	84.59	-	-	-	-
0.01	orthogonal	85.47	89.63	2.16	55.33	2.20	54.83
0.1	orthogonal	94.10	91.44	2.12	55.37	2.24	54.10
1	orthogonal	93.84	90.83	2.06	57.07	2.36	51.12
100	orthogonal	-	-	2.04	57.51	2.36	51.20
none	orthogonal	93.24	90.51	2.06	57.38	2.34	51.30
none	Glorot normal	66.71	79.33	2.08	57.37	2.34	51.04
none	identity	53.53	42.72	2.25	53.83	2.68	45.35
LSTM		97.30	92.62	1.92	60.84	1.64	65.53

orthogonality does not guarantee a faster convergence to the optimal solution. It is important to stress that for the models that do not use bounded singular values, the Glorot initialization as well as the identity initialization performed poorly compared to an orthogonal initialization.

This suggests that orthogonality is beneficial at the time of initialization and that weights close to the orthogonal space of parameters, especially during the very early phases of training, help to converge quickly towards an optimal solution. This being said, a weight orthogonality maintained throughout the training can affect negatively, not only the speed of convergence, but more importantly, the performance of the model.

Figure 5.4 confirms our observations. For the 200 length copy task, a model with the highest margin ($m = 1$) converged the fastest (see Figure 5.1). We can deduce from Figure 5.4 that the gradient norm is maximal at the beginning of training and is smoothly decreased for the remaining of the training phase as the spectrum of the norm intensities becomes lighter in Figure 5.4. This is not surprising as the more we train, the more we approach a local minimum of the training loss, and so, the norm of the gradient decreases towards 0. The only case where the norm of the gradient does not decrease is for the model which corresponds to a strict orthogonality ($m = 0$). This leads us to think that for a model expressing a strict orthogonality, the learning rate must be decreased in a faster pace, so the norm of the weights update decreases through training iterations. This decreasing of the corrections would allow to converge quickly towards the optimal solution when a strict orthogonality is expressed.

The peephole LSTM outperformed the factorized RNNs in both sequential and permuted MNIST. Nevertheless, the factorized RNN with margin $m = 0.1$ yielded a competitive performance on

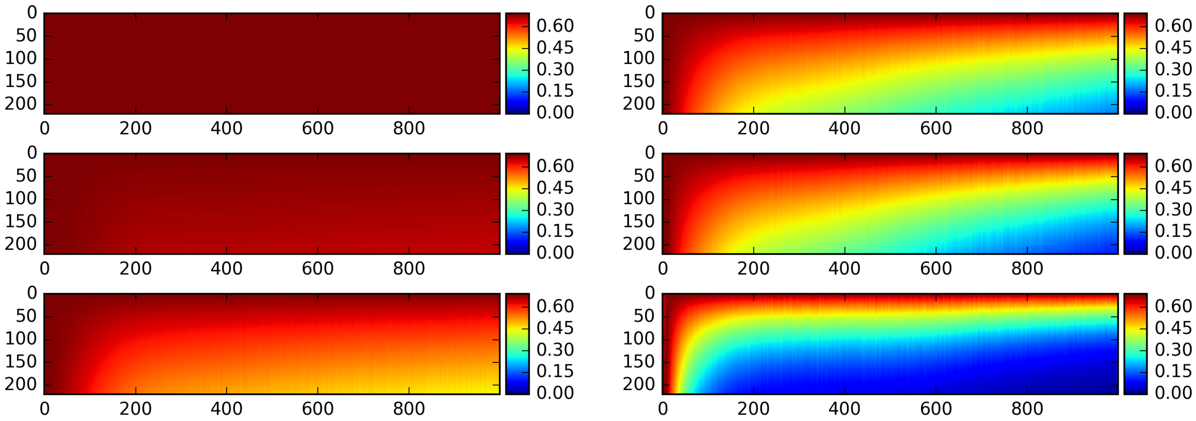


Figure 5.4 (Taken from Vorontsov et al. (2017)). Gradient evolution for the copy task during training. The norm of the gradient of the loss from the last time step with respect to the hidden units at a given time step for a length 220 RNN over 1000 update iterations for different margins. Iterations are along the abscissa and time steps are denoted along the ordinate. The first column margins are : 0, 0.001, 0.01. The second column margins are : 0.1, 1, no margin. Gradient norms are normalized across the time dimension.

permuted MNIST. It is important to mention that the factorized RNNs contain significantly less parameters, and so less capacity, than the LSTM due the gated architecture of the latter.

For PTB character prediction, we use prediction accuracy as well as Bits Per Character (BPC) as evaluation metrics. The Bits per character is the expected $\log_2$ cross-entropy between the predicted and the target distributions of the characters in a given string $\mathbf{x}$ of length N. Given the string $\mathbf{x}$ and the predicted character distribution $\hat{\mathbf{p}}_t$ outputed by an RNN model for the string $\mathbf{x}$, the Bits Per Character formulas is given by :

$$\begin{aligned} bpc(\mathbf{x}, \hat{\mathbf{p}}_t) = \frac{1}{N}\sum_{t=1}^{T} H(\mathbf{p}_t, \hat{\mathbf{p}}_t) &= -\frac{1}{N}\sum_{t=1}^{N}\sum_{c=1}^{|alphabet|} p_t(c)\cdot\log_2\left(\hat{p}_t(c)\right) \\ &= -\frac{1}{N}\sum_{t=1}^{N}\sum_{c=1}^{|alphabet|} p_t(c)\cdot\log_2\left(\hat{p}(c\,|\,\mathbf{h}_t)\right) \\ &= -\frac{1}{N}\sum_{t=1}^{N}\log_2\left(\hat{p}(x_t\,|\,\mathbf{h}_t)\right), \end{aligned} \tag{5.13}$$

where $\mathbf{p}_t$ is the target distribution of the characters at time t. $\mathbf{p}_t$ is a categorical distribution represented by a one-hot vector where $p_t(x_t) = 1$, $p_t(x'_t) = 0\,\forall\, x'_t \neq x_t$ and x_t is the index of the character existing in the alphabet which has the position t in $\mathbf{x}$. $|alphabet|$ is the size of the alphabet.

We trained a factorized RNN with 512 hidden units for 200 epochs with geodesic gradient descent on the bases using a learning rate of 10^{-6} and RMSprop on the other parameters using a learning rate of 10^{-3}. The gradients magnitudes were clipped to 30. We also trained an LSTM with 512 hidden units with peephole connections. Early stopping was performed for the LSTM training in order to avoid overfitting. As for the MNIST experiments, the the LSTM parameters were orthogonally initialized and the forget gate bias was initialized to $\mathbf{1}$.

Prediction results are shown in Table 5.1 both for a subset of short sequences (up to 75 characters ; 23% of data) and for a subset of long sequences (up to 300 characters ; 99% of data). It is important to recall that in the language modeling tasks, the vanishing and exploding gradient problem is not as severe as it is in sequential and permuted MNIST. This is because an output is predicted at each instant t, and so, a loss is computed at each time step (see section 5.2.3 and equations (5.7) and (5.8)).

As for the sequential and permuted MNIST tasks, and without surprise, the LSTM, with its high capacity gated-architecture, performed better than the simple factorized models on both shorter and longer sequences. For the factorized RNNs, we can see from Table 5.1 that the highest accuracy for shorter sentences corresponds to the model with the highest margin, i.e, $m = 100$. The factorized models corresponding to non-bounded singular values and to the Glorot initialization performed better than the models with a margin less than 100. This is not surprising as for language modeling, the problem of vanishing and exploding gradient is less severe especially when sentences are short.

For longer sentences, the opposite is observed. Models with smaller margins performed better and strict orthogonality was beneficial as it allowed to obtain the best accuracy among the factorized models.

5.4.4 Spectral Evolution During Training

We evaluated the evolution of the singular values in terms of their mean, standard deviation, maximal and minimal values in all of our experiments on permuted MNIST. Singular values evolution for different values of the margin is shown in Figure 5.5. We found that, in general, singular values do not deviate by much from their mean as long as the factorized RNN model is orthogonally intialized. In such case, the mean is maintained around 1.05 along all the training process. The only case where the singular values reached the margin upper and lower bounds is when we used a very big learning rate which does not allow the model to converge. The only case where singular values where deviating by much from their mean is where no bounds on the singular values were set while the parameters where initialized using the Glorot initialization. In such case the mean of the singular values was around 1. However, as it can be seen from Figure 5.5, the singular values had a much bigger standard deviation than models initialized orthogonally. The minimal and maximal values where also deviating by much from the mean. For the model initialized orthogonally where no bounds were set for the singular values, a maximal and a minimal value of approximately 1.18 and 0.91 were respectively obtained at end of the epoch 200. For the same model which used Glorot initialization, the maximal and minimal values reached 2.3 and 0 at the end of epoch 200.

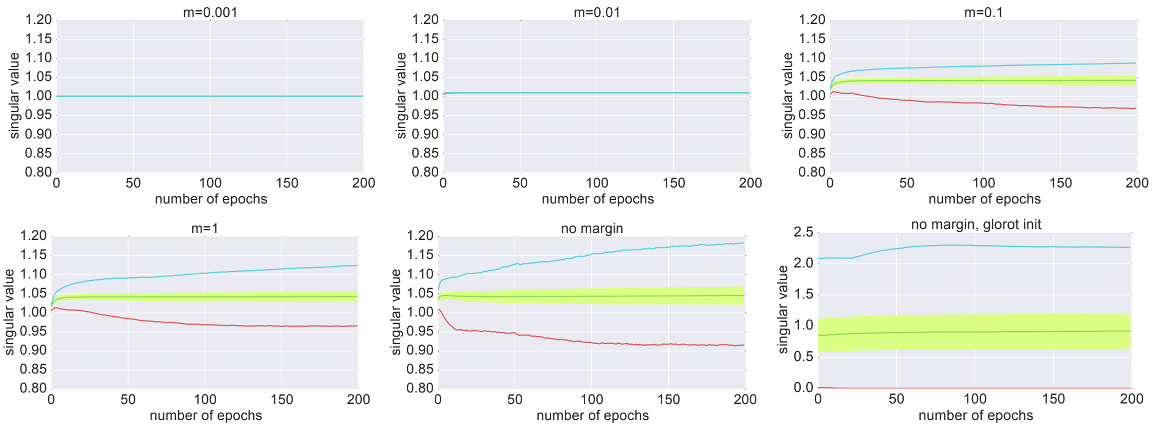


Figure 5.5 (Taken from Vorontsov et al. (2017)). Singular value evolution on the permuted sequential MNIST task for factorized RNNs with different spectral margin sizes (m). The singular value distributions are summarized with the mean (green line, center) and standard deviation (green shading about mean), minimum (red, bottom) and maximum (blue, top) values. All models are initialized with orthogonal hidden to hidden transition matrices except for the model that yielded the plot on the bottom right, where Glorot normal initialization is used.

The spectral evolution results emphasize the importance of the orthogonal initialization as the latter allows to stabilize the training of the RNN model while giving it the freedom to deviate from orthogonality when the margin is different than 0.

5.5 Conclusion

We carried out a study on the effect of orthogonality of the hidden transition matrix of a Recurrent Neural Network on the learning process. More specifically, we assessed the usefulness of orthogonality of the hidden weight matrix for the training procedure by controlling its deviation from the manifold of orthogonal transformations and measuring the generalization performance of the RNN for different bounds of deviation. We conducted experiments on toy and real-world tasks. The obtained results suggest that orthogonality is beneficial at the early stages of the training process as it ensures the propagation of the hidden representation through time and the backpropagation of the gradient. It also suggest that, in most cases, a small deviation from orthogonality after the first iterations is necessary to achieve a faster convergence and, more importantly, a better performance of the model. This being said, a non-controlled deviation may lead to instabilities during training due to the vanishing and exploding gradient problem. It may also lead to a slow convergence and a poor performance of the model. This allows us to conclude that a strict orthogonality or an uncontrolled deviation from the Stiefel Manifold are, in general, detrimental to the training process and that a trade-off in the amount of deviation has to be considered in order to ensure a faster convergence and a good performance of the model.

CHAPTER 6 CONCLUSION AND GENERAL DISCUSSION

In this chapter we first summarize and discuss the main results obtained in the thesis focusing on the contributions we have made in chapters 3, 4 and 5. In a second step, we present the limits and possible improvements of the proposed approaches and open a discussion on potential future research directions. In our thesis we pursued three main objectives. The first is to combine theoretical and empirical insights in order to devise a principled approach towards training both real and complex-valued deep neural networks. Here we addressed challenging issues related to the training modules or "building blocks" for complex-valued deep neural networks. The second objective is to provide a solution to the challenging problem of signal retrieval and signal separation in the frequency domain. As a case study, we considered audio source separation. Finally the third objective is to assess the utility of orthogonal matrices, used in training recurrent neural networks, and measure the performance of the corresponding model. Throughout chapters 3 to 5 we showed the expediency of the methods we proposed in the context of the announced challenges and contended objectives. The methods we have conceived are more generally applicable and have been applied to model various deep learning tasks. We return to these avenues in sections 6.1.1 and 6.1.3 with a more general discussion.

6.1 Summary and General Discussion

In this manuscript we proposed a set of original contributions in the form of three chapters on methods for understanding, stabilizing and enhancing the training process for both real and complex-valued deep neural networks. We tackled both theoretical and practical challenging issues associated with problems causing training instability, among them, the vanishing and exploding gradient problem. More specifically the dedicated approaches provided novel complex-valued training modules or "building blocks" including appropriate initialization and normalization techniques for learning various tasks with real and intrinsically complex-valued inputs. Results from intensive experimentation underpinned our theoretic framework. Indeed these building blocks have been shown, in many cases, to avoid numerical problems during training and thereby enable the use of complex-valued representations. These representations are well-suited for frequency domain signals, as they have the capacity to explicitly encode frequency magnitude and phase components. In particular, models built with these building blocks have excelled at tasks including computer vision, music transcription, speech spectrum prediction, signal retrieval and audio source separation. We summarize below the key content of each chapter by reviewing the objectives, motivation, and contributions.

6.1.1 General Discussion on the First Chapter

Real-valued deep learning neural networks have become the benchmark for a wide range of computer vision, automatic speech recognition and natural language processing tasks. However one of the major drawbacks they face during the training process is the vanishing and exploding gradients. Difficulties intensify even more when it comes to capturing long-term dependency structures. This is the case for training recurrent neural networks (RNNs).

Building blocks including initialization and normalization techniques have provided solutions to overcome this notorious problem. Initialization techniques have been debated in Glorot and Bengio (2010) and He et al. (2015), Normalization techniques such as batch normalization (Ioffe and Szegedy, 2015), weight normalization (Salimans and Kingma, 2016) and layer normalization (Ba et al., 2016) have been shown to reduce training instability, sustain higher learning rate, speed-up convergence and boost generalization performance. The mainstream of building blocks for training deep neural networks are established for real-valued inputs, structures and computational operations. However real-valued parametrization are not suitable for learning tasks where intrinsically complex inputs are required or greatly effective. This is also the case when real-valued input is converted to a complex form by means of a transformation such as the Fourier-related transforms. In fact complex-valued signals occur in many areas and are thus of fundamental interest. Examples of such inputs include, Magnetic Reasoning Images, Synthetic Aperture Radar Data, Radar Imaging, Acoustic Signal, Ultrasonic Imaging, Communications Signal Processing, Very-Long-Baseline Interferometry, etc.

The use of complex arithmetic for neural networks representations has been sidelined for some time. Currently, deep complex neural networks are in their infancy. Few articles have investigated the utility of using complex numbers in deep learning architectures. The tardy adoption of such architectures could be explained for the following reasons. First, they are hard to train using the gradient descent algorithm. This is mainly due to the problem that we unveiled in section 3.6.6. The training instability is caused by the increased correlation between the real and the imaginary parts of each neural unit output through the network layers. Furthermore, there aren't public libraries available to train complex-valued models.

Latest work on recurrent neural networks and past studies on signal processing advocate that there is definitely palpable payoffs in using complex arithmetic to parametrize deep neural networks, as its ability to encode complex inputs. Actually the reviewed recent works show that complex numbers allow richer representational capacity (Wisdom et al., 2016; Arjovsky et al., 2016) better generalization (Hirose and Yoshida, 2012), enable easier optimization (Nitta, 2002), faster learning (Arjovsky et al., 2016; Wisdom et al., 2016; Danihelka et al., 2016), higher memory capacity and noise-robust memory retrieval mechanisms (Danihelka et al., 2016).

Notwithstanding their appealing properties and the imminent likelihood of introducing new neural architectures, complex-valued deep neural networks have been disregarded due to the nonexistence of proper building blocks. The lack of a framework for complex-valued building blocks represents a clear gap in deep learning tooling. In chapter 3 we filled this gap by providing, a novel complex-valued deep neural network and training methods that go further than a simplistic theoretical adaptation of the real-valued counterpart. We showed that the new complex design allows the functionality of neural networks to be sustained in order to solve a variety of applied problems that do fit within the realm of real-valued neural schemes. We argued that it is beneficial not only for wave-originating inputs (complex by design such as radar data), but also for a large-spectrum of information processing with time-frequency treatment using Short Time Fourier transform (Real inputs transformed by STFT).

We have specifically designed here the main atomic components of complex-valued deep neural networks and applied them to convolutional feed-forward neural networks (FFNNs) and convolutional LSTMs. From a methodological perspective we relied on complex convolutions and presented algorithms for complex batch-normalization and complex weight initialization schemes. These complex building blocks are further implemented in intensive experiments with end-to-end training plans. These are related to image classification, automatic music transcription and spectrum prediction. The contributions presented in chapter 3 and discussed in more details beneath are :

1. A specification of a new complex valued activation function and a comparison of its performance to that of various complex-based activation functions. See sections 3.3.4, 3.4.1 and 3.6.7 ;
2. A formulation of complex batch normalization, which is described in Section 3.3.5 ;
3. Complex weight initialization, which is presented in Section 3.3.6 ;
4. A state of the art result on the MusicNet multi-instrument music transcription dataset, presented in Section 3.4.2 ;
5. A state of the art result in the Speech Spectrum Prediction task on the TIMIT dataset, presented in Section 3.4.3.

Contributions 1-3 represent the core of the work we presented in this chapter 3, laying down the mathematical details for implementing complex-valued building blocks of a deep neural network.

In Contribution 1, we defined a new complex-valued rectifier activation function, referred to as $\mathbb{C}$ReLU. We also compared its performance, in deep feed-forward neural networks, with that of different analogues. The results in Table 3.1, show the superiority of our $\mathbb{C}$ReLU.

As we pointed out earlier, deep neural networks generally rely upon batch normalization to ensure training stability and a good generalization performance on held-out data. The formulation of batch normalization relates only to real values. In contribution 2 we suggested a specification of batch normalization that is appropriate to complex-valued neural networks. To transform an array of complex numbers into a standard complex array, it is not sufficient to shift and scale them to get zero mean and identity covariance matrix. This kind of normalization does not lead necessarily to equal variances of the real and imaginary components and null covariance between them. Furthermore the resulting distribution is not guaranteed to be circular even if the original data is Gaussian. As an alternative we have formulated the problem as whitening two dimensional vectors In fact, in our first experimentation, we discovered that in a deep complex-valued neural network, the training instabilities were caused, not only by the vanishing and exploding gradients, but also by the increased correlation between the real and the imaginary parts of each neural unit output through the network layers. The whitening step in the normalization procedure permits to decorrelate the imaginary and real parts of a neural unit. This has the potential of preventing co-adaptation between the two components which lowers the risk of overfitting (Cogswell et al., 2015; Srivastava et al., 2014).

In Contribution 3 we proposed a formulation of complex weight initialization which is generally applied when batch normalization is not implemented. We followed the same orientations as in Glorot and Bengio (2010) and He et al. (2015) to derive the variance of the complex weight parameters. In the reported experimentation, we have used variants of the derived complex weight initialization that leverages the independence property of unitary matrices. Indeed learning uncorrelated features helps to accomplish better generalization and faster learning. Notice that we also proposed in section 3.3.6 an equivalent initialization for real-valued models using the independence property of orthogonal matrices.

Contributions 4 and 5 are the result of an evaluation of the performance of our deep complex models on many computer vision tasks including music transcription using the MusicNet dataset and on Speech Spectrum Prediction using the TIMIT dataset. State-of-the-art performance has been achieved on these audio-related tasks. In addition, our approach has been shown to be effective on standard image classification benchmarks, specifically, CIFAR-10, CIFAR-100. For vision classification tasks, learning deep complex-valued representations yielded a performance that is competitive with the respective real-valued architectures.

Recent Developments Based on Contributions within the First Chapter

Our work has paved the way for new research to integrate and extend our complex-valued framework. For instance, Gaudet and Maida (2018), Parcollet et al. (2018), Parcollet et al. (2019) and Wu et al. (2019) adopted the methodology that we had introduced to design new deep neural networks architectures based on hypercomplex arithmetic. Gaudet and Maida (2018) presented deep quaternion networks where quaternion weight initialization and batch normalization methods were formulated. Later complex and quaternion methods have been extended by Wu et al. (2019) to octonion networks. In another essay, Parcollet et al. (2018) leveraged the intialization method presented in Gaudet and Maida (2018) to perform a speech recognition task. The magnitude spectrogram, its first order and second order temporal differences were embedded in the quaternion space through the layers of a deep convolutional network. This has the advantage of encoding the nonlinear correlations that may be present in the quaternion-based input features. Similar adaptations to RNNs and LSTMs were also discussed in Parcollet et al. (2019).

The main feature of the works quoted above is their ability to better generalize than real-valued networks while using a reduced number of parameters. For example, Parcollet et al. (2019) achieved better performance in speech recognition using a quaternion LSTM with 3.3 times fewer parameters than its real counterpart. Similarly Wu et al. (2019) used a deep octonion network with eight times fewer parameters than its real counterpart. They obtained significant improvements over the real-valued network for the classification task using the CIFAR-100 dataset. This could be explained by the fact that the hypercomplex operations encode correlations between the real and the imaginary parts of the input. However, hypercomplex operations are more expensive than their real analogues in terms of computation. For example, a complex multiplication or a convolution costs 4 times more than its real counterpart. Quaternion operations are 4 times more expensive than their complex counterparts, and therefore 16 times more than the current analogs. Octonions are 64 times more expensive than their real counterparts. It should also be noted that the analog of our whitening operation is expensive in the case of hypercomplex representations. Indeed computing the inverse of the square root of the covariance matrix requires performing eigenvalue decomposition whose complexity is cubic since the dimension N of the covariance matrix is greater than 2 in this case. Gaudet and Maida (2018) argue that the whitening operation does not require an inverse square root and that the Cholesky decomposition can be used to obtain the inverse of the covariance matrix. However, the complexity of Cholesky decomposition is also cubic.

Motivated by our findings, others proposed new complex-valued models for sequential prediction such as Wolter and Yao (2018a) and Wolter and Yao (2018b). Moran et al. (2018) went beyond our unitary initialization to propose a soft unitary regularization approach to invert transmission effects in multimode optical fibers. Their regularization is similar to the soft orthogonal constraint proposed by Henaff et al. (2016). Dedmari et al. (2018) designed a deep complex U-Net using our building blocks. The deep complex U-Net was applied for magnetic resonance images (MRI) to de-alias the reconstruction artifacts within undersampled MRI images. The proposed method achieved state-of-the-art in recovering fine structures and high frequency textures of MRI images. Using our building blocks, Choi et al. (2019) designed a complex U-Net along with a phase-aware loss in the time domain and a new masking method. The authors achieved state-of-the-art in speech enhancement.

6.1.2 General Discussion on the Second Chapter

In our second chapter, we leveraged these building blocks along with the convolution theorem, to develop methods for extracting and separating complex-valued signals in the Fourier domain by considering audio source separation as a case study. In fact, when modeling audio signal separation within a deep learning setting, it has been common to use a simple algorithm to encode the mixture signal. The encoding is based on the spectral representation by means of the Short Time Fourier Transform. The resulting spectrogram is a complex-valued mapping involving magnitude and phase components. In this setting a main difficulty of isolating phase and magnitude has been acknowledged in previously published work. The phase of the clean speech has been frequently ignored due to the difficulty of its approximation. Most approaches first estimate the magnitude of the spectrum as a time-frequency mask for each source in the mixture, and then reconstructing using the masked magnitude spectrogram and reusing the phase of the corrupted sound mixture. (Huang et al., 2014a; Xu et al., 2014; Grais et al., 2016; Nugraha et al., 2016; Takahashi et al., 2018). Reusing phase of the original corrupted speech has a negative impact on the quality of the inference of the clean speech. The impact is amplified when noisy conditions deteriorate, i.e., when signal-to-noise ratio (SNR) is low.

Recent works have recommended the use of a complex-valued ratio masks (cRMs). Williamson et al. (2016), trained a deep neural network to jointly estimate real and imaginary components of a complex ideal ratio mask (cIRM). The authors assert that the cIRM method provides the solution to the decoupling phase/magnitude dilemma. However, this method is shown to be limited as it propagates only the error between the cIRM and its estimated counterpart at the training stage. This often ends up with performance deterioration (Wang et al., 2014; Yu et al., 2017). Ephrat et al. (2018) propose a cIRM in which the real and imaginary parts are sigmoidal mapping of the matching components of the model output. Despite its advantages, this approach to cIRM have main shortcomings. First the corresponding loss function does not reflect the cIRM distribution appropriately. Furthermore, the masking method yields a cRM values in a bounded rotation interval of 0-90 degrees. This prevents from fine-tuning the noisy phase. Other works, such as Hershey et al. (2015); Chen et al. (2016), estimate only the magnitude of the STFTs and reconstruct the time-domain signal with the Griffin-Lim algorithm (Griffin and Lim, 1984) or other similar procedures (Sturmel and Daudet, 2006).

We leveraged the results acquired in Chapter 3 and provided a novel fully complex-valued pipeline named “Deep Complex Separator” for automatic signal retrieval and source separation in the frequency domain. The Deep Complex Separator is a novel masking-based method founded on a complex-valued version of Feature-wise Linear Modulation (Perez et al., 2018) and takes advantage of the convolution theorem which states that the Fourier transform of two convolved signals is the elementwise product of their Fourier transforms. This new masking method is essential for speech separation tasks and deep architectures. We also formulated a new explicitly phase-aware loss function which takes into accounts the complex-valued components of the spectrogram and has applealing propoerties such as amplitude and shift-invariance.

Our contributions are summarized as follows :

1. We present a novel masking method based on Feature-wise Linear Modulation (FiLM) (Perez et al., 2018) to create multiple separated candidates for each of the signals we aim to retrieve from a mixture of inputs. A signal averaging operation on the candidates is performed in order to increase the robustness of the signal to noise and interference. Before the averaging

procedure, a dropout is implemented on the signal candidates in order to reduce the amount of interference and noise correlation existing between the different candidates. Our masking method could be seen as one performing local ensembling. In the case of audio source separations, we aim to retrieve distinct audio signals associated with each speaker in the input mix. The candidates are averaged in order to obtain the final separated speech for each of the speakers in question. Our experiments demonstrate the efficacy of our proposed masking method, and show its regularizing effect.

2. We propose and explore a new frequency-domain loss taking **explicitly** into account the magnitude and phase of signals. A key characteristic of our loss is that it is amplitude and shift-invariant. Our comparative analysis related to different phase-aware losses defined in time and frequency domains demonstrates the advantage of our proposed loss.

6.1.3 General Discussion on the Third Chapter

It is common practice to use unitary transition matrices and orthogonal constraints as perfect solutions to the technical difficulties of vanishing and exploding gradients in recurrent neural networks for long-term dependency tasks. However real-world tasks, such as natural language processing, might not necessitate such hard constraints. We conjecture that narrowing the weights to fulfill the orthogonality propriety might be excessively restrictive as it confines the search space of the parameters to the Stiefel manifold. Moreover, hard and soft orthogonal constraints are computationally expensive. In this chapter, our goal was to know whether orthogonality is restrictive or offers benefits in terms of optimization convergence and performance of the associated recurrent neural network.

To this end, we assessed the utility of the orthogonality of the hidden transition weight matrix by controlling its deviation from the manifold of orthogonal transformations and measuring the generalization performance of the associated RNN for different deviations. By combining Singular Value Decomposition (SVD) of the hidden transition matrix and the geodesic gradient descent algorithm, we explicitly expressed the magnitude of deviation from orthogonality and optimized it by setting a learnable margin for the singular values. This allowed us to carry out a sensitivity analysis and identify the cases where deviating from orthogonality could be either beneficial or detrimental. We conducted experiments on toy and real-world tasks. The obtained results suggest that orthogonality is beneficial at the early stages of the training process as it ensures the propagation of the hidden representation through time and the backpropagation of the gradient. It also suggest that, in most cases, a small deviation from orthogonality after the first iterations is necessary to achieve a faster convergence and, more importantly, a better performance of the model. That said, an uncontrolled deviation can lead to instabilities during training because of the vanishing and exploding gradient problem. It can also lead to slow convergence and poor performance of the model. We concluded that a strict orthogonality or an uncontrolled deviation from the Stiefel manifold is generally detrimental to the training process and that a compromise of the amount of deviation must be considered in order to ensure a faster convergence and a good performance of the model. To summarize our contributions are :

1. A weight matrix factorization and parameterization strategy is proposed. The latter is employed to bound matrix norms and therein control the degree of expansivity induced during backpropagation.
2. A sensitivity analysis to identify the cases where deviating from orthogonality could be either beneficial or detrimental is proposed.

We have shown that the strict orthogonality of hidden weight matrices is beneficial in the beginning of the training process and a controlled deviation from orthogonality after the early stages of training is generally beneficial.

Recent Developments Based on the Third Chapter

Several works have benefited from our analysis and used similar constraints to optimize along the Stiefel Manifold. Huang et al. (2018) used semi-orthogonal matrices to obtain independent filters for each convolutional layer in a feed-forward network. This is also similar to our idea of orthogonal and unitary initialization for real-valued and complex convolutional layers in chapter 3. Huang et al. (2018) propose to decorrelate the filters of each convolutional layer by performing optimization in the Stiefel manifold of each filter. Jose et al. (2017) perform soft orthogonal constraints similar to ours and the one presented in Henaff et al. (2016) for their complex-valued Kronecker RNN. Zhang et al. (2018c) used our singular value margin formulation expressed in equation (5.11) in order to control the amount of deviation from the space of orthogonal parameters. Bansal et al. (2018) used advanced analytical tools, more precisely, mutual coherence and restricted isometry property to perform soft orthogonality constraints on convolutional layers. The complexity of their regularization method is negligible compared to other methods, including ours, which use singular value decomposition. Kusupati et al. (2018) addressed limitations of methods using orthogonal and unitary RNNs as most of them are costly in terms of computational complexity. They added residual connections between sucessive states in order to take into account long-term dependencies.

6.2 Limitations and Future Work

Although the results we obtained with our complex-based building blocks throughout the thesis are encouraging, and their subsequent adaptations in training deep complex neural networks, through recent research, are promising, much remains to be done to improve their performance and determine the reasons for their successes. In this section we will pin point to some limitations and the future work. We will not propose solutions to them, but will only sketch the outline of possible routes inspired from our observations when conducting empirical studies.

In our work we performed an analysis on the usefulness of orthogonaility for the hidden transition matrix of a vanilla RNN. It would have also been ideal if our analysis had been applied to other models such as GRUs (Cho et al., 2014), LSTMs (Hochreiter and Schmidhuber, 1997) and also attentional weights (Bahdanau et al., 2014) such those of Transformer (Vaswani et al., 2017).

In the case of speech separation task, our experiments are carried out with two speakers. Increasing the number of speakers will shed more light on the robustness of the proposed signal retrieval mechanism to the number of sources to be extracted. Further experimentation is recommended.

A logical follow up project would be to combine audio-visual signals for isolating a single speech signal from a mixture of sounds including speakers and background noise using the complex-valued deep neural network framework.

Recently, a generative model for spectral representations of audio was proposed (Vasquez and Lewis, 2019). It discards phase information by converting the frequency representation to the Mel scale using Mel filter banks followed by an elementwise application of the log function to perform a logarithmic rescaling of the amplitudes. The generated spectrograms are converted back to time-domain signals using the Griffin-Lim algorithm (Griffin and Lim, 1984) or a gradient-based inversion algorithm (Decorsière et al., 2014). Our building could be used to design an end-to-end architecture performing generation of complex-valued spectral representations of audio. This would allow to leverage the phase information and to reduce the amount of artifacts and distortions in the generated spectrogram.

As pointed out in earlier section, adaptation of the complex-valued building blocks methodology in a more abstract setting, such as hypercomplex specification of neural units, has led to increased performance of deep complex neural representations. One reason for such success could be attributed to the diversity (increased number) of features describing the activations of neural units. Other reasons could be attributed to the ability of hypercomplex models to capture dependencies in the input features and learn representation dynamics according to the task at hand. A more fine-grained analyses are needed to identify and understand the causes of such success.

Although ablation studies designed to evaluate the impact of the 2D whitening, performed within the complex batch normalization, established its *raison d'être*, the exact causes for the success of complex batch normalization as a whole transformation need to be understood. A systematic investigation of the origins of the effectiveness may include the control of the internal covariate shift, i.e, the change of the layers' input distributions (Ioffe and Szegedy, 2015). A second element of this investigation is to evaluate the impact of batch normalization on the training process including the effect on the optimization landscape which is tightly linked to the behavior of the gradients.

In a final commentary, we hope that our findings have highlighted the tendency of complex-valued deep neural networks to solve difficult tasks when data lie in the complex domain. We consider these results as a stimulus to pursue more systematic studies of the foundations of the complex-valued representations. We believe that our proposed methods could lead to new directions of research where complex-valued modeling and signal retrieval and extraction are needed.

REFERENCES

M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, “Tensorflow : A system for large-scale machine learning”, dans *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

M. Arjovsky, A. Shah, et Y. Bengio, “Unitary evolution recurrent neural networks”, dans *International Conference on Machine Learning*, 2016, pp. 1120–1128.

J. K.-T. Au, “An ab initio approach to the inverse problem-based design of photonic bandgap devices”, Thèse de doctorat, California Institute of Technology, 2007.

J. L. Ba, J. R. Kiros, et G. E. Hinton, “Layer normalization”, *arXiv preprint arXiv :1607.06450*, 2016.

D. Bahdanau, K. Cho, et Y. Bengio, “Neural machine translation by jointly learning to align and translate”, *arXiv preprint arXiv :1409.0473*, 2014.

S. Bai, J. Z. Kolter, et V. Koltun, “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling”, *arXiv preprint arXiv :1803.01271*, 2018.

D. Balduzzi, M. Frean, L. Leary, J. Lewis, K. W.-D. Ma, et B. McWilliams, “The shattered gradients problem : If resnets are the answer, then what is the question ?” dans *Proceedings of the 34th International Conference on Machine Learning-Volume 70.* JMLR. org, 2017, pp. 342–350.

N. Bansal, X. Chen, et Z. Wang, “Can we gain more from orthogonality regularizations in training deep networks ?” dans *Advances in Neural Information Processing Systems*, 2018, pp. 4261–4271.

K. Benchenane, P. H. Tiesinga, et F. P. Battaglia, “Oscillations in the prefrontal cortex : a gateway to memory and attention”, *Current opinion in neurobiology*, vol. 21, no. 3, pp. 475–485, 2011.

Y. Bengio, P. Simard, et P. Frasconi, “Learning long-term dependencies with gradient descent is difficult”, *IEEE transactions on neural networks*, 1994.

Y. Bengio, P. Lamblin, D. Popovici, et H. Larochelle, “Greedy layer-wise training of deep networks”, dans *Advances in neural information processing systems*, 2007, pp. 153–160.

D. M. Bradley, “Learning in modular systems”, CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, Rapp. tech., 2010.

J. Bruna et S. Mallat, “Invariant scattering convolution networks”, *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1872–1886, 2013.

J. Bruna, W. Zaremba, A. Szlam, et Y. LeCun, “Spectral networks and locally connected networks on graphs”, *arXiv preprint arXiv :1312.6203*, 2013.

J. Bruna, S. Chintala, Y. LeCun, S. Piantino, A. Szlam, et M. Tygert, “A mathematical motivation for complex-valued convolutional networks”, *arXiv preprint arXiv :1503.03438*, 2015.

J. Chalupper et H. Fastl, “Simulation of hearing impairment based on the fourier time transformation”, dans *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100)*, vol. 2. IEEE, 2000, pp. II857–II860.

Z. Chen, Y. Luo, et N. Mesgarani, “Deep attractor network for single-microphone speaker separation”, *CoRR*, vol. abs/1611.08930, 2016.

E. C. Cherry, “Some experiments on the recognition of speech, with one and with two ears”, *The Journal of the acoustical society of America*, vol. 25, no. 5, pp. 975–979, 1953.

K. Cho, B. Van Merriënboer, D. Bahdanau, et Y. Bengio, “On the properties of neural machine translation : Encoder-decoder approaches”, *arXiv preprint arXiv :1409.1259*, 2014.

H.-S. Choi, J. Kim, J. Huh, A. Kim, J.-W. Ha, et K. Lee, “Phase-aware speech enhancement with deep complex u-net”, *ICLR*, 2019.

F. Chollet *et al.*, “Keras : Deep learning library for theano and tensorflow”, *URL : https ://keras.io/k*, 2015.

M. Chui, “Artificial intelligence the next digital frontier ?” *McKinsey and Company Global Institute*, vol. 47, 2017.

M. Cogswell, F. Ahmed, R. Girshick, L. Zitnick, et D. Batra, “Reducing overfitting in deep networks by decorrelating representations”, *arXiv preprint arXiv :1511.06068*, 2015.

F. Crick, “Function of the thalamic reticular complex : the searchlight hypothesis”, *Proceedings of the National Academy of Sciences*, vol. 81, no. 14, pp. 4586–4590, 1984.

I. Danihelka, G. Wayne, B. Uria, N. Kalchbrenner, et A. Graves, “Associative long short-term memory”, *ICML*, 2016.

R. Decorsière, P. L. Søndergaard, E. N. MacDonald, et T. Dau, “Inversion of auditory spectrograms, traditional spectrograms, and other envelope representations”, *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 1, pp. 46–56, 2014.

M. A. Dedmari, S. Conjeti, S. Estrada, P. Ehses, T. Stöcker, et M. Reuter, “Complex fully convolutional neural networks for mr image reconstruction”, dans *International Workshop on Machine Learning for Medical Image Reconstruction.* Springer, 2018, pp. 30–38.

M. Defferrard, X. Bresson, et P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering”, dans *Advances in neural information processing systems*, 2016, pp. 3844–3852.

M. Drozdzal, E. Vorontsov, G. Chartrand, S. Kadoury, et C. Pal, “The importance of skip connections in biomedical image segmentation”, dans *Deep Learning and Data Labeling for Medical Applications.* Springer, 2016, pp. 179–187.

J. Du, Y. Tu, Y. Xu, L. Dai, et C. Lee, “Speech separation of a target speaker based on deep neural networks”, dans *Proc. of ICSP*, 2014, pp. 473–477.

V. Dumoulin, J. Shlens, et M. Kudlur, “A learned representation for artistic style”.

N. Q. K. Duong, E. Vincent, et R. Gribonval, “Under-determined reverberant audio source separation using a full-rank spatial covariance model”, *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2010.

A. Ephrat, I. Mosseri, O. Lang, T. Dekel, K. Wilson, A. Hassidim, W. T. Freeman, et M. Rubinstein, “Looking to listen at the cocktail party : A speaker-independent audio-visual model for speech separation”, *CoRR*, 2018.

H. Erdogan, J. R. Hershey, S. Watanabe, et J. L. Roux, “Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks”, dans *Proc. of ICASSP*, 2015, pp. 708–712.

C. Févotte et J. Idier, “Algorithms for nonnegative matrix factorization with the beta-divergence”, *CoRR*, vol. abs/1010.1763, 2010.

P. Fries, “A mechanism for cognitive dynamics : neuronal communication through neuronal coherence”, *Trends in cognitive sciences*, vol. 9, no. 10, pp. 474–480, 2005.

C. Févotte, N. Bertin, et J.-L. Durrieu, “Nonnegative matrix factorization with the itakura-saito divergence : With application to music analysis”, *Neural Computation*, 2009.

R. Gao, R. S. Feris, et K. Grauman, “Learning to separate object sounds by watching unlabeled video”, *CoRR*, 2018.

C. J. Gaudet et A. S. Maida, “Deep quaternion networks”, dans *2018 International Joint Conference on Neural Networks (IJCNN).* IEEE, 2018, pp. 1–8.

G. M. Georgiou et C. Koutsougeras, “Complex domain backpropagation”, *IEEE transactions on Circuits and systems II : analog and digital signal processing*, 1992.

X. Glorot et Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks”, dans *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

X. Glorot, A. Bordes, et Y. Bengio, “Deep sparse rectifier neural networks”, dans *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 315–323.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, et Y. Bengio, “Generative adversarial nets”, dans *Advances in neural information processing systems*, 2014, pp. 2672–2680.

I. Goodfellow, Y. Bengio, et A. Courville, *Deep Learning.* MIT Press, 2016, `http://www.deeplearningbook.org`.

E. M. Grais, G. Roma, A. J. Simpson, et M. D. Plumbley, “Single-channel audio source separation using deep neural network ensembles”, dans *Audio Engineering Society Convention 140.* Audio Engineering Society, 2016.

K. Greff, R. K. Srivastava, et J. Schmidhuber, “Highway and residual networks learn unrolled iterative estimation”, *arXiv preprint arXiv :1612.07771*, 2016.

D. Griffin et J. Lim, “Signal estimation from modified short-time fourier transform”, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 2, pp. 236–243, 1984.

N. Guberman, “On complex valued convolutional neural networks”, *arXiv preprint arXiv :1602.09046*, 2016.

R. L. Hahnloser, “On the piecewise analysis of networks of linear threshold neurons”, *Neural Networks*, vol. 11, no. 4, pp. 691–697, 1998.

K. He, X. Zhang, S. Ren, et J. Sun, “Delving deep into rectifiers : Surpassing human-level performance on imagenet classification”, dans *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

K. He, X. Zhang, S. Ren, et J. Sun, “Deep residual learning for image recognition”, dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

K. He, X. Zhang, S. Ren, et J. Sun, “Identity mappings in deep residual networks”, dans *European conference on computer vision.* Springer, 2016, pp. 630–645.

M. Henaff, A. Szlam, et Y. LeCun, “Recurrent orthogonal networks and long-memory tasks”, *arXiv preprint arXiv :1602.06662*, 2016.

J. R. Hershey, Z. Chen, J. L. Roux, et S. Watanabe, “Deep clustering : Discriminative embeddings for segmentation and separation”, *CoRR*, 2015.

G. E. Hinton, S. Osindero, et Y.-W. Teh, “A fast learning algorithm for deep belief nets”, *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

A. Hirose, *Complex-valued neural networks : theories and applications.* World Scientific, 2003.

A. Hirose, *Complex-valued neural networks.* Springer Science & Business Media, 2012, vol. 400.

A. Hirose et S. Yoshida, “Generalization characteristics of complex-valued feedforward neural networks in relation to signal coherence”, *IEEE Transactions on Neural Networks and learning systems*, vol. 23, no. 4, pp. 541–551, 2012.

S. Hochreiter, “Untersuchungen zu dynamischen neuronalen netzen”, Thèse de doctorat, 1991.

S. Hochreiter et J. Schmidhuber, “Long short-term memory”, *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

G. Hu et D. Wang, “Monaural speech segregation based on pitch tracking and amplitude modulation”, *Trans. Neur. Netw.*, 2004.

G. Huang, Z. Liu, L. Van Der Maaten, et K. Q. Weinberger, “Densely connected convolutional networks”, dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

L. Huang, X. Liu, B. Lang, A. W. Yu, Y. Wang, et B. Li, “Orthogonal weight normalization : Solution to optimization over multiple dependent stiefel manifolds in deep neural networks”, dans *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

P.-S. Huang, M. Kim, M. Hasegawa-Johnson, et P. Smaragdis, “Deep learning for monaural speech separation”, dans *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2014, pp. 1562–1566.

P.-S. Huang, K. Minje, M. Hasegawa-Johnson, et P. Smaragdis, “Deep learning for monaural speech separation”, *ICASSP*, 2014.

S. L. Hyland et G. Rätsch, “Learning unitary operators with help from u (n)”, dans *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

A. Hyvärinen et E. Oja, “Independent component analysis : algorithms and applications”, *Neural Networks*, 2000.

S. Ioffe et C. Szegedy, “Batch normalization : Accelerating deep network training by reducing internal covariate shift”, *arXiv preprint arXiv :1502.03167*, 2015.

S. Jastrzebski, D. Arpit, N. Ballas, V. Verma, T. Che, et Y. Bengio, “Residual connections encourage iterative inference”, *arXiv preprint arXiv :1710.04773*, 2017.

L. Jing, Y. Shen, T. Dubček, J. Peurifoy, S. Skirlo, M. Tegmark, et M. Soljačić, “Tunable efficient unitary neural networks (eunn) and their application to rnn”, *arXiv preprint arXiv :1612.05231*, 2016.

L. Jing, C. Gulcehre, J. Peurifoy, Y. Shen, M. Tegmark, M. Soljacic, et Y. Bengio, “Gated orthogonal recurrent units : On learning to forget”, *Neural computation*, vol. 31, no. 4, pp. 765–783, 2019.

C. Jose, M. Cisse, et F. Fleuret, “Kronecker recurrent units”, *arXiv preprint arXiv :1705.10142*, 2017.

P. Juszczak, D. Tax, et R. P. Duin, “Feature scaling in support vector data description”. Citeseer, 2002.

T. Karras, T. Aila, S. Laine, et J. Lehtinen, “Progressive growing of gans for improved quality, stability, and variation”, *arXiv preprint arXiv :1710.10196*, 2017.

T. Kim et T. Adalı, “Approximation by fully complex multilayer perceptrons”, *Neural computation*, 2003.

D. Kingma et J. Ba, “Adam : A method for stochastic optimization”, *arXiv preprint arXiv :1412.6980*, 2014.

T. N. Kipf et M. Welling, “Semi-supervised classification with graph convolutional networks”, *arXiv preprint arXiv :1609.02907*, 2016.

G. Klambauer, T. Unterthiner, A. Mayr, et S. Hochreiter, “Self-normalizing neural networks”, dans *Advances in neural information processing systems*, 2017, pp. 971–980.

J. Kohler, H. Daneshmand, A. Lucchi, M. Zhou, K. Neymeyr, et T. Hofmann, “Towards a theoretical understanding of batch normalization”, *arXiv preprint arXiv :1805.10694*, 2018.

A. Krizhevsky, I. Sutskever, et G. E. Hinton, “Imagenet classification with deep convolutional neural networks”, dans *Advances in neural information processing systems*, 2012, pp. 1097–1105.

D. Krueger et R. Memisevic, “Regularizing rnns by stabilizing activations”, *arXiv preprint arXiv :1511.08400*, 2015.

A. Kusupati, M. Singh, K. Bhatia, A. Kumar, P. Jain, et M. Varma, “Fastgrnn : A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network”, dans *Advances in Neural Information Processing Systems*, 2018, pp. 9017–9028.

Q. V. Le, N. Jaitly, et G. E. Hinton, “A simple way to initialize recurrent networks of rectified linear units”, *arXiv preprint arXiv :1504.00941*, 2015.

Y. LeCun, L. Bottou, Y. Bengio, et P. Haffner, “Gradient-based learning applied to document recognition”, *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

Y. A. LeCun, L. Bottou, G. B. Orr, et K.-R. Müller, “Efficient backprop”, dans *Neural networks : Tricks of the trade.* Springer, 2012, pp. 9–48.

Y. Lee, C. Wang, S. Wang, J. Wang, et C. Wu, “Fully complex deep neural network for phase-incorporating monaural source separation”, dans *ICASP*, 2017.

M. Lezcano-Casado et D. Martínez-Rubio, “Cheap orthogonal constraints in neural networks : A simple parametrization of the orthogonal and unitary group”, *arXiv preprint arXiv :1901.08428*, 2019.

A. Liutkus, D. Fitzgerald, Z. Rafii, B. Pardo, et L. Daudet, “Kernel additive models for source separation”, *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2014.

Y. Luo et N. Mesgarani, “Tasnet : time-domain audio separation network for real-time, single-channel speech separation”, *CoRR*, vol. abs/1711.00541, 2017.

Y. Luo et N. Mesgarani, “Tasnet : Surpassing ideal time-frequency masking for speech separation”, *arXiv preprint arXiv :1809.07454*, 2018.

M. P. Marcus, M. A. Marcinkiewicz, et B. Santorini, “Building a large annotated corpus of english : The penn treebank”, *Computational linguistics*, vol. 19, no. 2, pp. 313–330, 1993.

J. H. McDermott, “Audition”, *Stevens’ Handbook of Experimental Psychology and Cognitive Neuroscience*, vol. 2, pp. 1–57, 2018.

Z. Mhammedi, A. Hellicar, A. Rahman, et J. Bailey, “Efficient orthogonal parametrisation of recurrent neural networks using householder reflections”, *arXiv preprint arXiv :1612.00188*, 2016.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, et M. Riedmiller, “Playing atari with deep reinforcement learning”, *arXiv preprint arXiv :1312.5602*, 2013.

V. Mnih, N. Heess, A. Graves *et al.*, “Recurrent models of visual attention”, dans *Advances in neural information processing systems*, 2014, pp. 2204–2212.

O. Moran, P. Caramazza, D. Faccio, et R. Murray-Smith, “Deep, complex, invertible networks for inversion of transmission effects in multimode optical fibres”, dans *Advances in Neural Information Processing Systems*, 2018, pp. 3280–3291.

Y. Nesterov, “A method of solving a convex programming problem with convergence rate o (1/k2)”, 1983.

Y. Nishimori, “A note on riemannian optimization methods on the stiefel and the grassmann manifolds”, *dim*, vol. 1, p. 2, 2005.

T. Nitta, “On the critical points of the complex-valued neural network”, dans *Neural Information Processing, 2002. ICONIP’02. Proceedings of the 9th International Conference on*, vol. 3. IEEE, 2002, pp. 1099–1103.

T. Nitta, “Orthogonality of decision boundaries in complex-valued neural networks”, *Neural Computation*, 2004.

A. A. Nugraha, A. Liutkus, et E. Vincent, “Multichannel audio source separation with deep neural networks”, *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 9, pp. 1652–1664, 2016.

A. V. Oppenheim et J. S. Lim, “The importance of phase in signals”, *Proceedings of the IEEE*, vol. 69, no. 5, pp. 529–541, 1981.

E. Oyallon et S. Mallat, “Deep roto-translation scattering for object classification”, dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2865–2873.

K. Paliwal, K. Wójcicki, et B. Shannon, “The importance of phase in speech enhancement”, *speech communication*, vol. 53, no. 4, pp. 465–494, 2011.

K. K. Paliwal et L. D. Alsteris, “On the usefulness of stft phase spectrum in human listening tests”, *Speech communication*, vol. 45, no. 2, pp. 153–170, 2005.

T. Parcollet, Y. Zhang, M. Morchid, C. Trabelsi, G. Linarès, R. De Mori, et Y. Bengio, “Quaternion convolutional neural networks for end-to-end automatic speech recognition”, *arXiv preprint arXiv :1806.07789*, 2018.

T. Parcollet, M. Ravanelli, M. Morchid, G. Linarès, C. Trabelsi, R. De Mori, et Y. Bengio, “Quaternion recurrent neural networks”, *ICLR*, 2019.

R. Pascanu, T. Mikolov, et Y. Bengio, “On the difficulty of training recurrent neural networks.” *ICML (3)*, vol. 28, pp. 1310–1318, 2013.

A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, et A. Lerer, “Automatic differentiation in pytorch”, 2017.

E. Perez, F. Strub, H. De Vries, V. Dumoulin, et A. Courville, “Film : Visual reasoning with a general conditioning layer”, 2018.

T. Plate, “Holographic reduced representations : Convolution algebra for compositional distributed representations.” dans *IJCAI*, 1991, pp. 30–35.

T. A. Plate, “Holographic reduced representations”, *IEEE Transactions on Neural networks*, vol. 6, no. 3, pp. 623–641, 1995.

T. A. Plate, “Holographic reduced representation : Distributed representation for cognitive structures”, 2003.

C. Poultney, S. Chopra, Y. L. Cun *et al.*, “Efficient learning of sparse representations with an energy-based model”, dans *Advances in neural information processing systems*, 2007, pp. 1137–1144.

D. P. Reichert et T. Serre, “Neuronal synchrony in complex-valued deep networks”, *arXiv preprint arXiv :1312.6115*, 2013.

O. Rippel, J. Snoek, et R. P. Adams, “Spectral representations for convolutional neural networks”, dans *Advances in Neural Information Processing Systems*, 2015, pp. 2449–2457.

O. Ronneberger, P. Fischer, et T. Brox, “U-net : Convolutional networks for biomedical image segmentation”, pp. 234–241, 2015.

F. Rosenblatt, “The perceptron : a probabilistic model for information storage and organization in the brain.” *Psychological review*, vol. 65, no. 6, p. 386, 1958.

F. Rosenblatt, “Principles of neurodynamics. perceptrons and the theory of brain mechanisms”, CORNELL AERONAUTICAL LAB INC BUFFALO NY, Rapp. tech., 1961.

D. E. Rumelhart, G. E. Hinton, et R. J. Williams, “Learning representations by back-propagating errors.” *Nature*, 1986.

T. Salimans et D. P. Kingma, “Weight normalization : A simple reparameterization to accelerate training of deep neural networks”, *arXiv preprint arXiv :1602.07868*, 2016.

S. Santurkar, D. Tsipras, A. Ilyas, et A. Madry, “How does batch normalization help optimization ?” dans *Advances in Neural Information Processing Systems*, 2018, pp. 2483–2493.

A. M. Sarroff, V. Shepardson, et M. A. Casey, “Learning representations using complex-valued nets”, *arXiv preprint arXiv :1511.06351*, 2015.

A. M. Saxe, J. L. McClelland, et S. Ganguli, “Exact solutions to the nonlinear dynamics of learning in deep linear neural networks”, *arXiv preprint arXiv :1312.6120*, 2013.

G. Shi, M. M. Shanechi, et P. Aarabi, “On the importance of phase in human speech recognition”, *IEEE transactions on audio, speech, and language processing*, vol. 14, no. 5, pp. 1867–1874, 2006.

D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, “Mastering the game of go without human knowledge”, *Nature*, vol. 550, no. 7676, p. 354, 2017.

K. Simonyan et A. Zisserman, “Very deep convolutional networks for large-scale image recognition”, dans *Proc. ICLR*, 2015.

W. Singer et C. M. Gray, “Visual feature integration and the temporal correlation hypothesis”, *Annual review of neuroscience*, vol. 18, no. 1, pp. 555–586, 1995.

P. Smaragdis, B. Raj, et M. Shashanka, “A probabilistic latent variable model for acoustic modeling”, dans *In Workshop on Advances in Models for Acoustic Processing at NIPS*, 2006.

P. Smaragdis, B. Raj, et M. Shashanka, “Supervised and semi-supervised separation of sounds from single-channel mixtures”, dans *International Conference on Independent Component Analysis and Signal Separation.* Springer, 2007, pp. 414–421.

J. O. Smith, “Digital audio resampling”, *Online http ://www-ccrma. stanford. edu/˜ jos/resample*, 2002.

L. Sorber, M. V. Barel, et L. D. Lathauwer, “Unconstrained optimization of real functions in complex variables”, *SIAM Journal on Optimization*, vol. 22, no. 3, pp. 879–898, 2012.

M. Spiertz, “Source-filter based clustering for monaural blind source separation”, *International Conference on Digital Audio Effects*, 2009.

N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, et R. Salakhutdinov, “Dropout : a simple way to prevent neural networks from overfitting.” *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

R. K. Srivastava, K. Greff, et J. Schmidhuber, “Training very deep networks”, dans *Advances in neural information processing systems*, 2015, pp. 2377–2385.

G. B. Stanley, “Reading and writing the neural code”, *Nature neuroscience*, vol. 16, no. 3, p. 259, 2013.

N. Sturmel et L. Daudet, “Signal reconstruction from stft magnitude : A state of the art”, dans *In Proc. of the International conference on digital audio effects*, 2006.

H. D. Tagare, “Notes on optimization on stiefel manifolds”, Tech. Rep., Yale University, Rapp. tech., 2011.

N. Takahashi, N. Goswami, et Y. Mitsufuji, “Mmdenselstm : An efficient combination of convolutional and recurrent neural networks for audio source separation”, dans *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*. IEEE, 2018, pp. 106–110.

T. T. D. Team, R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov *et al.*, “Theano : A python framework for fast computation of mathematical expressions”, *arXiv preprint arXiv :1605.02688*, 2016.

E. Terhardt, “Fourier transformation of time signals : Conceptual revision”, *Acta Acustica United with Acustica*, vol. 57, no. 4-5, pp. 242–255, 1985.

Theano Development Team, “Theano : A Python framework for fast computation of mathematical expressions”, *arXiv e-prints*, vol. abs/1605.02688, Mai 2016. En ligne : `http://arxiv.org/abs/1605.02688`

J. Thickstun, Z. Harchaoui, et S. Kakade, “Learning features of music from scratch”, dans *Proc. ICLR*, 2016.

T. Tieleman et G. Hinton, “Lecture 6.5—RmsProp : Divide the gradient by a running average of its recent magnitude”, COURSERA : Neural Networks for Machine Learning, 2012.

J. Tompson, R. Goroshin, A. Jain, Y. LeCun, et C. Bregler, “Efficient object localization using convolutional networks”, dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 648–656.

C. Trabelsi, O. Bilaniuk, Y. Zhang, D. Serdyuk, S. Subramanian, J. F. Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, et C. J. Pal, “Deep complex networks”, *ICLR*, 2017.

T. Trouillon et M. Nickel, “Complex and holographic embeddings of knowledge graphs : a comparison”, *arXiv preprint arXiv :1707.01475*, 2017.

T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, et G. Bouchard, “Complex embeddings for simple link prediction”, dans *International Conference on Machine Learning*, 2016, pp. 2071–2080.

M. Tygert, A. Szlam, S. Chintala, M. Ranzato, Y. Tian, et W. Zaremba, “Scale-invariant learning and convolutional networks”, *arXiv preprint arXiv :1506.08230*, 2015.

P. Uhlhaas, G. Pipa, B. Lima, L. Melloni, S. Neuenschwander, D. Nikolić, et W. Singer, “Neural synchrony in cortical networks : history, concept and current status”, *Frontiers in integrative neuroscience*, vol. 3, p. 17, 2009.

D. Ulyanov, A. Vedaldi, et V. Lempitsky, “Instance normalization : The missing ingredient for fast stylization”, *arXiv preprint arXiv :1607.08022*, 2016.

A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, et K. Kavukcuoglu, “Wavenet : A generative model for raw audio”, *CoRR abs/1609.03499*, 2016.

A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves *et al.*, “Conditional image generation with pixelcnn decoders”, dans *Advances In Neural Information Processing Systems*, 2016, pp. 4790–4798.

S. Vasquez et M. Lewis, “Melnet : A generative model for audio in the frequency domain”, *arXiv preprint arXiv :1906.01083*, 2019.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, et I. Polosukhin, “Attention is all you need”, dans *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

S. Venkataramani et P. Smaragdis, “End-to-end networks for supervised single-channel speech separation”, *CoRR*, vol. abs/1810.02568, 2018.

E. Vincent, R. Gribonval, et C. Févotte, “Performance measurement in blind audio source separation.” *IEEE Trans. Audio, Speech & Language Processing*, vol. 14, no. 4, pp. 1462–1469, 2006.

T. Virtanen, “Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria”, *Trans. Audio, Speech and Lang. Proc.*, 2007.

C. Von Der Malsburg, “The correlation theory of brain function”, dans *Models of neural networks.* Springer, 1994, pp. 95–119.

E. Vorontsov, C. Trabelsi, S. Kadoury, et C. Pal, “On orthogonality and learning recurrent networks with long term dependencies”, dans *Proceedings of the 34th International Conference on Machine Learning-Volume 70.* JMLR. org, 2017, pp. 3570–3578.

B. Wang et M. Plumbley, “Investigating single-channel audio source separation methods based on non-negative matrix factorization”, *ICA Research Network International Workshop*, 2006.

D. Wang et J. Chen, “Supervised speech separation based on deep learning : An overview”, *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2018.

Y. Wang, A. Narayanan, et D. Wang, “On training targets for supervised speech separation”, *IEEE/ACM transactions on audio, speech, and language processing*, vol. 22, no. 12, pp. 1849–1858, 2014.

Z. Wang, J. L. Roux, D. Wang, et J. R. Hershey, “End-to-end speech separation with unfolded iterative phase reconstruction”, *CoRR*, vol. abs/1804.10204, 2018.

Z. Wang, X. Wang, X. Li, Q. Fu, et Y. Yan, “Oracle performance investigation of the ideal masks”, dans *2016 IEEE International Workshop on Acoustic Signal Enhancement (IWAENC)*. IEEE, 2016, pp. 1–5.

C. Weng, D. Yu, M. L. Seltzer, et J. Droppo, “Deep neural networks for single-channel multi-talker speech recognition”, *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 23, no. 10, pp. 1670–1679, 2015.

F. Weninger, H. Erdogan, S. Watanabe, E. Vincent, J. Le Roux, J. R. Hershey, et B. Schuller, “Speech enhancement with lstm recurrent neural networks and its application to noise-robust asr”, dans *International Conference on Latent Variable Analysis and Signal Separation*. Springer, 2015, pp. 91–99.

R. J. Williams et J. Peng, “An efficient gradient-based algorithm for on-line training of recurrent network trajectories”, *Neural computation*, vol. 2, no. 4, pp. 490–501, 1990.

D. S. Williamson, Y. Wang, et D. Wang, “Complex ratio masking for monaural speech separation”, *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 24, no. 3, pp. 483–492, 2016.

S. Wisdom, T. Powers, J. Hershey, J. Le Roux, et L. Atlas, “Full-capacity unitary recurrent neural networks”, dans *Advances in Neural Information Processing Systems*, 2016, pp. 4880–4888.

M. Wolter et A. Yao, “Fourier rnns for sequence analysis and prediction”, *arXiv preprint arXiv :1812.05645*, 2018.

M. Wolter et A. Yao, “Gated complex recurrent neural networks”, *arXiv preprint arXiv :1806.08267*, 2018.

D. E. Worrall, S. J. Garbin, D. Turmukhambetov, et G. J. Brostow, “Harmonic networks : Deep translation and rotation equivariance”, *arXiv preprint arXiv :1612.04642*, 2016.

J. Wu, L. Xu, Y. Kong, L. Senhadji, et H. Shu, “Deep octonion networks”, *arXiv preprint arXiv :1903.08478*, 2019.

Y. Wu et K. He, “Group normalization”, dans *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 3–19.

S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, et W.-c. Woo, “Convolutional lstm network : A machine learning approach for precipitation nowcasting”, dans *Advances in neural information processing systems*, 2015, pp. 802–810.

Y. Xu, J. Du, L.-R. Dai, et C.-H. Lee, “A regression approach to speech enhancement based on deep neural networks”, *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 1, pp. 7–19, 2014.

G.-P. Yang, C.-I. Tuan, H.-Y. Lee, et L.-s. Lee, “Improved speech separation with time-and-frequency cross-domain joint embedding and clustering”, *arXiv preprint arXiv :1904.07845*, 2019.

G. Yang, J. Pennington, V. Rao, J. Sohl-Dickstein, et S. S. Schoenholz, “A mean field theory of batch normalization”, *arXiv preprint arXiv :1902.08129*, 2019.

D. Yu, M. Kolbæk, Z.-H. Tan, et J. Jensen, “Permutation invariant training of deep models for speaker-independent multi-talker speech separation”, dans *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 241–245.

S. Zagoruyko et N. Komodakis, “Wide residual networks”, *arXiv preprint arXiv :1605.07146*, 2016.

R. S. Zemel, C. K. Williams, et M. C. Mozer, “Lending direction to neural networks”, *NIPS*, 1995.

J. Zhang, Q. Lei, et I. S. Dhillon, “Stabilizing gradients for deep neural networks via efficient svd parameterization”, *arXiv preprint arXiv :1803.09327*, 2018.

J. Zhang, Y. Lin, Z. Song, et I. S. Dhillon, “Learning long term dependencies via fourier recurrent units”, *arXiv preprint arXiv :1803.06585*, 2018.

Z. Zhang, J. Geiger, J. Pohjalainen, A. E.-D. Mousa, W. Jin, et B. Schuller, “Deep learning for environmentally robust speech recognition : An overview of recent developments”, *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 9, no. 5, p. 49, 2018.

M. Zibulevsky et B. A. Pearlmutter, “Blind source separation by sparse decomposition in a signal dictionary”, *Neural Computation*, 2001.

J. G. Zilly, R. K. Srivastava, J. Koutník, et J. Schmidhuber, “Recurrent highway networks”, *arXiv preprint arXiv :1607.03474*, 2016.

E. Zwicker et H. Fastl, *Psychoacoustics : Facts and models.* Springer Science & Business Media, 2013, vol. 22.

APPENDIX A BASIC CONCEPTS

Complex Numbers

Complex numbers are the set of numbers that can be expressed as $z = a + ib$ where $a,\, b \in \mathbb{R}$ and z is in the complex space $\mathbb{C}$. i denotes the solution for the equation $x^2 = -1$. As a consequence, we can write $i^2 = -1$ or $\sqrt{-1} = \pm i$. i is called the *imaginary number*. a is called *real part* of z and is denoted by $\Re(z)$. b is called *imaginary part* of z and is denoted by $\Im(z)$. The coordinates $(a,\, b) = (\Re(z),\, \Im(z))$ of the complex number z are called the Cartesian coordinates of z in the complex plane. Another alternative to represent a complex number is to use the polar form $(|z|,\, \theta_z)$ where $|z|$ is called modulus, magnitude and also amplitude of z. θ_z is the phase of z. $|z| = \sqrt{\Re(z)^2 + \Im(z)^2} = \sqrt{a^2 + b^2}$. $\frac{\Im(z)}{\Re(z)} = \tan(\theta_z) = \frac{\sin(\theta_z)}{\cos(\theta_z)}\ \forall\ \theta_z \not\equiv 0 \pmod{2\pi}$. θ_z could then expressed as :

$$\theta_z = \begin{cases} \tan^{-1}(\frac{\Im(z)}{\Re(z)}) & \forall\ \Re(z) \in \mathbb{R} \setminus \{0\} \\ \quad \pi/2 & \forall\ \Im(z) \in\,]0,\ +\infty[\ \text{ and } \Re(z) = 0 \\ -\pi/2 & \forall\ \Im(z) \in\,]-\infty,\, 0[\ \text{ and } \Re(z) = 0 \\ 0 & \text{if } \ \Re(z) = 0 \ \text{ and } \ \Im(z) = 0 \end{cases} \tag{A.1}$$

We can write a complex number in any of the following forms :

$$z = \begin{cases} a + i\, b & \text{(Cartesian form)} \\ \Re(z)\ +\ i\,\Im(z) & \text{(Cartesian form)} \\ |z|\,[\cos(\theta_z) + i\sin(\theta_z)] & \text{(Polar form)} \\ |z|\, e^{i\theta_z} & \text{(Polar form)} \end{cases} \tag{A.2}$$

The conjugate of a complex number z is denote by z^* or $\bar{z}$ where $z^* = \bar{z} = a - ib = |z|e^{-i\theta}$. As a consequence, $zz^* = |z|^2 = |z^*|^2$, $z + z^* = 2\Re(z)$ and $z - z^* = 2i\,\Im(z)$.

The Vector and Matrix Representations of a Complex Number

let z_1 and $z_2 \in \mathbb{C}$ where $z_1 = a + ib$ and $z_2 = c + id$. the product of z_1 and z_2 would be equal to :

$$\begin{aligned} z_1 \cdot z_2 = z_2 \cdot z_1 &= ac + i\,ad + i\,bc - bd \\ &= \ ac - bd \\ &+ i(ad + bc). \end{aligned} \tag{A.3}$$

If we use the vector form in order to represent any complex number z such that $z = [\Re(z), \Im(z)]^T$ we can write the product $z_1 \cdot z_2$ in the following way :

$$z_1 \cdot z_2 = \begin{pmatrix} a & -b \\ b & a \end{pmatrix} \begin{pmatrix} c \\ d \end{pmatrix} = z_2 \cdot z_1 = \begin{pmatrix} c & -d \\ d & c \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} ac - bd \\ ad + bc \end{pmatrix}. \tag{A.4}$$

If we use the matrix form in order to represent any complex number z such that $z = \begin{pmatrix} \Re(z) & -\Im(z) \\ \Im(z) & \Re(z) \end{pmatrix}$ then :

$$z_1 \cdot z_2 = \begin{pmatrix} a & -b \\ b & a \end{pmatrix} \begin{pmatrix} c & -d \\ d & c \end{pmatrix} = z_2 \cdot z_1 = \begin{pmatrix} c & -d \\ d & c \end{pmatrix} \begin{pmatrix} a & -b \\ b & a \end{pmatrix} = \begin{pmatrix} ac - bd & -(ad + bc) \\ ad + bc & ac - bd \end{pmatrix}. \tag{A.5}$$

We can see from equations (A.4) and (A.5) that the first element in the multiplication is always written in the matrix form even if the result and the second element of the multiplication are written in the vector form.

Convolution

Convolution is a linear operation on two functions or signals which expresses how much two signals effect linearly one another. Similar to cross-correlation, it is considered as similarity or linear dependency measure between two signals. A convolution operation over two discrete 1D signals is defined as :

$$\mathbf{y}[n] = (\mathbf{f} * \mathbf{g})[n] = \sum_{k=-\infty}^{\infty} \mathbf{f}[k] \cdot \mathbf{g}[n-k] = (\mathbf{g} * \mathbf{f})[n] = \sum_{k=-\infty}^{\infty} \mathbf{g}[k] \cdot \mathbf{f}[n-k], \tag{A.6}$$

where the * denotes the convolution operation between the signals $\mathbf{f}$ and $\mathbf{g}$ while $\mathbf{y}$ is the output of the convolution operation. In the Digital Signal Processing community $\mathbf{g}$ is called the impulse response filter and the $\mathbf{g}[k]$ is the impulse response at the k^{th} instant. In the machine learning community, g is a referred as a convolution kernel and it is generally denoted by $\mathbf{w}$. In a convolution neural network terminology, $\mathbf{f}$ is referred as the input and it is commonly denoted by $\mathbf{x}$, $\mathbf{g}$ as the kernel and is commonly denoted by $\mathbf{w}$, while the output $\mathbf{y}$ is called the feature map. We can clearly observe in equation (A.6) the commutative property of convolution. In fact, besides of being a linear and a commutative operator, convolution is distibutive, i.e, $\mathbf{f} * (\mathbf{g} + \mathbf{h}) = (\mathbf{f} * \mathbf{g}) + (\mathbf{f} * \mathbf{h})$, and associative, i.e, $\mathbf{f} * (\mathbf{g} * \mathbf{h}) = (\mathbf{f} * \mathbf{g}) * \mathbf{h}$.

Another important propoerty of the convolution operator is its translation equivariance. Let's call T^k a shifting operator (also called lag operator or backshift operator in the time series community) which shifts a signal by k time steps or spatial positions such that $\mathrm{T}^k(\mathrm{f}[n]) = \mathrm{f}_k[n] = \mathrm{f}[n-k]$. The convolution is a translation equivariant operator means that $\mathrm{T}^k(\mathbf{f} * \mathbf{g}) = \mathrm{T}^k(\mathbf{f}) * \mathbf{g} = \mathbf{f} * (\mathrm{T}^k(\mathbf{g}))$.

It is important to notice that translation equivariance of the convolution operator is different from the translation invariance property of the convolution neural network. In fact, along with the pooling operation, the equivariance property of the convolution operator contributes to the approximate translation invariance of the convolutional layers in the neural network. When a max pooling is used in a convolution layer, the maximal values in a receptive field will be kept. Max pooling is a downsampling operation along the spatial dimension(s) of the signal allowing to retain only the maximum value of each overlapping window in the receptive field. The size of the overlapping window is called the pooling size. The amount of overlap is called the stride. Given that convolution is translation equivariant, these maximal values will be captured by the convolutional filters even if they have been probably translated from their positions after performing the pooling operation. This is why the combination of the max pooling operation and the convolution gives to the network its approximate translation invariance property.

For 2D signals, such as images and spectrograms, the convolution operation over two discrete signals is defined as :

$$\begin{aligned}\mathbf{Y}[n, m] = (\mathbf{F} * \mathbf{G})[n,\, m] &= \sum_j \sum_k \mathbf{F}[j,\, k] \cdot \mathbf{G}[n-j,\, m-k] \\ &= (\mathbf{G} * \mathbf{F})[n,\, m] = \sum_j \sum_k \mathbf{G}[j,\, k] \cdot \mathbf{F}[n-j,\, m-k].\end{aligned} \tag{A.7}$$

Discrete Fourier Transform

The Discrete Fourier Transform (DFT) is a linear transformation $\mathbf{F}$ applied on a finite complex-valued sequence $\mathbf{x}$ of N samples $\{\mathrm{x_0}, \mathrm{x_1}, \cdot, \mathrm{x_{N-1}}\}$ regularly spaced in time or space. It produces another finite complex-valued sequence $\mathbf{s}$ of N samples $\{\mathrm{s_0}, \mathrm{s_1}, \cdot, \mathrm{s_{N-1}}\}$ regularly spaced in the frequency space and given by :

$$\mathbf{Fx} = \mathbf{s} = \begin{cases} \mathrm{s}[0] = \mathrm{s}_0 = \sum_{n=0}^{N-1} \mathrm{x}_n \cdot e^{-i\frac{2\pi}{N}0n} \\ \mathrm{s}[1] = \mathrm{s}_1 = \sum_{n=0}^{N-1} \mathrm{x}_n \cdot e^{-i\frac{2\pi}{N}1n} \\ \mathrm{s}[2] = \mathrm{s}_2 = \sum_{n=0}^{N-1} \mathrm{x}_n \cdot e^{-i\frac{2\pi}{N}2n} \\ \vdots \\ \mathrm{s}[k] = \mathrm{s}_k = \sum_{n=0}^{N-1} \mathrm{x}_n \cdot e^{-i\frac{2\pi}{N}kn} \\ \vdots \\ \mathrm{s}[N-1] = \mathrm{s}_{N-1} = \sum_{n=0}^{N-1} \mathrm{x}_n \cdot e^{-i\frac{2\pi}{N}(N-1)n}. \end{cases} \tag{A.8}$$

$\mathbf{s}$ is called the spectrum of $\mathbf{x}$. We can see in equation (A.8) that the elements in $\mathbf{s}$ are regularly spaced in the frequency domain by $-i\frac{2\pi}{N}$. It can be observed, from equation (A.8) that the elements of $\mathbf{s}$ are obtained using a matrix product between The DFT matrix $\mathbf{F}$ and the signal $\mathbf{x}$. We denote

$e^{-i\frac{2\pi}{N}k.n}$ by $\omega^{k.n}$ (i.e, $\omega = e^{-i\frac{2\pi}{N}}$). Then, the matrix $\mathbf{F}$ could be written as :

$$\mathbf{F} = \begin{bmatrix} e^{-i\frac{2\pi}{N}0.0} & e^{-i\frac{2\pi}{N}0.1} & e^{-i\frac{2\pi}{N}0.2} & \dots & e^{-i\frac{2\pi}{N}0.(N-1)} \\ e^{-i\frac{2\pi}{N}1.0} & e^{-i\frac{2\pi}{N}1.1} & e^{-i\frac{2\pi}{N}1.2} & \dots & e^{-i\frac{2\pi}{N}1.(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ e^{-i\frac{2\pi}{N}(N-1).0} & e^{-i\frac{2\pi}{N}(N-1).1} & e^{-i\frac{2\pi}{N}(N-1).2} & \dots & e^{-i\frac{2\pi}{N}(N-1).(N-1)} \end{bmatrix} \tag{A.9}$$

$$\Rightarrow \mathbf{F} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{(N-1)} & \omega^{2(N-1)} & \dots & \omega^{(N-1)(N-1)} \end{bmatrix}. \tag{A.10}$$

The inverse of the DFT matrix $\mathbf{F}$ is $\mathbf{F^{-1}} = \frac{1}{N}\mathbf{F}^*$, where $\mathbf{F}^*$ is the conjugate transpose of $\mathbf{F}$. So when normalized by $\frac{1}{\sqrt{N}}$, $\mathbf{F}$ becomes unitary, i.e, $\frac{1}{\sqrt{N}}\mathbf{F} \cdot \frac{1}{\sqrt{N}}\mathbf{F}^* = \frac{1}{\sqrt{N}}\mathbf{F}^* \cdot \frac{1}{\sqrt{N}}\mathbf{F} = \mathbf{I}$. It could then be easily deduced that the transformed signal $\mathbf{x}$ could be retrieved using the conjugate transpose of the DFT matrix applied on the spectrum $\mathbf{s}$. This means that $\mathbf{x} = \frac{1}{N}\,\mathbf{F}^*\mathbf{s}$ where $\mathrm{x}_k = \frac{1}{N}\sum_{n=0}^{N-1} \mathrm{s}_n \cdot e^{i\frac{2\pi}{N}kn}$.

The Parseval's Theorem and Conservation of Energy

If $\mathbf{x}$ and $\mathbf{y}$ are 2 signals and $\mathbf{s}_x$ and $\mathbf{s}_y$ are their respective spectrums, the Parseval's theorem states the following :

$$\begin{aligned} \mathbf{x}^*\mathbf{y} &= \frac{1}{N}\mathbf{s}_x^*\mathbf{s}_y \\ \Leftrightarrow \sum_{n=0}^{N-1} \mathrm{x}_n^* \cdot \mathrm{y}_n &= \frac{1}{N}\sum_{n=0}^{N-1} (\mathrm{s}_x[n])^* \cdot \mathrm{s}_y[n]. \end{aligned} \tag{A.11}$$

Now for the special case when $\mathbf{x} = \mathbf{y}$, the Parseval's theorem becomes the Plancherel theorem which states the following :

$$\begin{aligned} \mathbf{x}^*\mathbf{x} &= \frac{1}{N}\mathbf{s}_x^*\mathbf{s}_x \\ \Leftrightarrow \sum_{n=0}^{N-1} \mathrm{x}_n^* \cdot \mathrm{x}_n &= \frac{1}{N}\sum_{n=0}^{N-1} (\mathrm{s}_x[n])^* \cdot \mathrm{s}_x[n] \\ \Leftrightarrow \sum_{n=0}^{N-1} |\mathrm{x}_n|^2 &= \frac{1}{N}\sum_{n=0}^{N-1} |\mathrm{s}_x[n]|^2 \\ \Leftrightarrow ||\mathbf{x}||^2 &= \frac{1}{N}\,||\mathbf{s}_x||^2. \end{aligned} \tag{A.12}$$

This means that the energy in $\mathbf{s}_x$ is N times the energy in $\mathbf{x}$ where N is the number of regularly spaced samples in $\mathbf{x}$.

The Convolution Theorem

The convolution theorem states that the convolution of 2 infinite discrete signals is obtained by the Inverse Discrete Fourier Transform of the product of their spectrums. It also states that the element-wise multiplication between 2 infinite sequences is obtained by the inverse Discrete Fourier Transform of the convolution of their spectrums. This could be written in the following way :

$$\begin{cases} \mathbf{x} * \mathbf{y} &= \mathbf{F}^{-1}\left[\mathbf{F}(\mathbf{x}) \odot \mathbf{F}(\mathbf{y})\right] \\ \mathbf{x} \odot \mathbf{y} &= \mathbf{F}^{-1}\left[\mathbf{F}(\mathbf{x}) \, * \mathbf{F}(\mathbf{y})\right], \end{cases} \tag{A.13}$$

where $\odot$ is the Hadamard product and it denotes the element-wise complex multiplication. In the case of finite sequences, one of two assumptions can be made. The first is that the finite sequence $\mathbf{x}$ of N samples repeats itself an infinite number of times, i.e, $\mathbf{x}$ is periodic with a period N. In such case, the circular convolution of the 2 finite discrete signals is obtained by the inverse Discrete Fourier Transform of the product of their spectrums and the convolution in equation (A.13) is replaced by a circular convolution.

$$\begin{cases} \mathbf{x} \circledast \mathbf{y} &= \mathbf{F}^{-1}\left[\mathbf{F}(\mathbf{x}) \odot \mathbf{F}(\mathbf{y})\right] \\ \mathbf{x} \odot \mathbf{y} &= \mathbf{F}^{-1}\left[\mathbf{F}(\mathbf{x}) \, \circledast \mathbf{F}(\mathbf{y})\right], \end{cases} \tag{A.14}$$

where $(\mathbf{x} \circledast \mathbf{y})[n] = \sum\limits_{k=0}^{N-1} \mathbf{x}[k] \cdot \mathbf{y}[(n-k) \bmod N]$.

The other assumption that could be made about finite sequences is that the signal contains zeros outside the finite number of samples it contains.

The Discrete Short-Time Fourier Transform

The Discrete Fourier Transform only applies to stationary processes. It only encodes the input in the frequency domain ignoring the time dimension. For non-stationary signals such as natural signals like speech, we may assume local stationarity (Paliwal and Alsteris, 2005) and, therefore, can be processed by a transformation which output is represented into time chunks. This is the case of the Discrete short-time Fourier Transform. The Discrete Short-Time Fourier Transform of a signal $\mathbf{s}$ is a representation of the signal in the time-frequency domain. The Discrete STFT is defined by :

$$\begin{aligned}
\mathrm{s}(m,\omega) = \mathrm{s}(m,\frac{2\pi}{N}k) &= \sum_{n=0}^{N-1} \mathrm{x}_n \cdot \mathrm{w}_{n-m} \cdot e^{-i\frac{2\pi}{N}kn} \\
&= \sum_{n=0}^{N-1} \mathrm{x}[n] \cdot \mathrm{w}[n-m] \cdot e^{-i\frac{2\pi}{N}kn} \\
&= \sum_{n=0}^{N-1} \left(\mathrm{x}[n] \cdot e^{-i\frac{2\pi}{N}kn}\right) \cdot \mathrm{w}[n-m] \\
&= \sum_{n=0}^{N-1} \left(\mathrm{x}[n] \cdot e^{-i\omega n}\right) \cdot \mathrm{w}[n-m].
\end{aligned} \tag{A.15}$$

We can see from equation (A.15) that the window sequence **w** is convolved with the signal formed by the product of **x** and the sequence containing the points on the unit circle that are separate by a phase of $\frac{2\pi k}{N}$. N samples of the signal **x** are converted to the frequency domain for each each time chunk.

APPENDIX B GLOROT AND HE INITIALIZATIONS

The Glorot initialization

Despite the increasing number of practical uses for deep learning, training deep neural networks remains a challenging problem. Training advanced deep neural networks is computationally expensive and even tuning a simple neural network to get improved results is particularly tedious. But, one way to reduce training time is initialization of the model's parameters (weights and biases). If the initialization is done appropriately then optimization and hence parameter estimation could be achieved in least time, otherwise, convergence to a minima using stochastic gradient descent will not be guaranteed. In this thesis, we make use of two initialization techniques, widely used in deep learning research, the Glorot and He initializations. Given the lack of technical details, found in subsequent research work related to these methods, we have found it useful, for the reader that we include, in the following, an in-depth mathematical derivation of the missing details. Notice that in this thesis we build on these initializations to introduce new ones for the complex-valued neural network architectures that we propose in the next chapters.

Weight initialization is shown to have a great impact on the network performance. As will be shown, controlling the variance of the distribution, from which initial weights are drawn, allows to regulate the way how layers gradually intensify or reduce the variance of the activations and error signal. Appropriate initializing schemes of deep neural networks has led into a sequence of innovations dating back to the unsupervised pretraining in Hinton et al. (2006); Bengio et al. (2007). The intuition of Glorot and Bengio (2010) is that controlling the variance of the distributions from which parameters are drawn permits to regulate the way layers gradually increase or decrease the variance of activations and error signals. Lately, He et al. (2015) refined the method by selecting linear rectifiers as nonlinear transformations to activate layer units. Indeed rectifiers halve the sample variance since, at initialization and on average, they are active for half their inputs.

The goal of the Glorot and He initializations (Glorot and Bengio, 2010; He et al., 2015) is to prevent training instability that appears in the course of neural networks training process. These instabilities are caused by either the growing or the vanishing variance of the activations and their respective derivatives. The exploding and the vanishing of the variance could happen either when propagating the information from the bottom to the top of the neural network, or, when backpropagating the error information related to the derivative of the loss function with respect to the activations.

Glorot and Bengio (2010) were influenced by the thesis of Bradley (2010) where the latter explains that the variance of the gradient, in a linear neural network (neural network with linear activation functions), diminishes when backpropagating the error term. Bradley (2010) noticed that the decaying behavior, of the gradient variance, starts immediately after the initialization. Glorot and

Bengio (2010) have provided a proper initialization for neural networks in a linear regime.

A feed-forward neural network is a nonlinear transformation of an input vector $\mathbf{x}$ to an output vector $\mathbf{y}$. Consider the l^{th} hidden layer in a deep feed-forward neural network, and let $\mathbf{x}^l$, $\mathbf{y}^l$ and $\mathbf{b}^l$ be the activation, the preactivation and bias vectors of the l^{th} layer respectively. Then, for a network of L layers we can write :

$$\begin{cases} \mathbf{y}^l = \mathbf{W}^l.\mathbf{x}^{l-1} + \mathbf{b}^l \\ \mathbf{x}^l = f(\mathbf{y}^l), \end{cases} \quad \forall\ l = 1, 2, ..., L \tag{B.1}$$

where f is an element-wise nonlinear activation function and $\mathbf{W}^l$ is a weight matrix propagating information from the $(l-1)^{th}$ layer units to the l^{th} layer units. In equation (B.1) the summed inputs $\mathbf{y}^l$ to the l^{th} layer are computed through the linear projection weight matrix $\mathbf{W}^l$ and the bottom-up inputs $\mathbf{x}^{l-1}$ coming from previous layer. $\mathbf{x}^l$, $\mathbf{y}^l$ and $\mathbf{b}^l$ are vectors in $\mathbb{R}^{N^l_{out}}$. N^l_{in} and N^l_{out} are the number of inputs fed to the l^{th} layer, and the number of outputs coming out from the l^{th} layer, respectively. $\mathbf{W}^l \in \mathbb{R}^{N^l_{in} \times N^l_{out}}$ (equivalently $\mathbf{W}^l \in \mathbb{R}^{N^{l-1}_{out} \times N^l_{out}}$ since $N^l_{in} = N^{l-1}_{out}$). From equation (B.1), the j^{th} component x^l_j of the activation vector $\mathbf{x}^l$ is :

$$x^l_j = f\left(y^l_j\right) = f\left(\sum_{i=1}^{N^l_{in}} W^l_{ji}\, x^{l-1}_i + b^l_j\right), \quad j = 1, ..., N^l_{out}, \tag{B.2}$$

where $x^l_j \in \mathbb{R}$ the activation of the j^{th} unit in the l^{th} layer, $y^l_j \in \mathbb{R}$ the pre-activation of the j^{th} unit in the l^{th} layer, $W^l_{ji} \in \mathbb{R}$ is the scalar weight connecting the i^{th} unit of the $(l-1)^{th}$ layer to the j^{th} unit of the l^{th} layer. Equation (B.2) implies that :

$$\begin{aligned} \delta x^{l-1}_i = \frac{\partial Cost}{\partial x^{l-1}_i} &= \sum_{j=1}^{N^l_{out}} \frac{\partial Cost}{\partial y^l_j} \frac{\partial y^l_j}{\partial x^{l-1}_i} = \sum_{j=1}^{N^l_{out}} W^l_{ji}\, \frac{\partial Cost}{\partial y^l_j} = \sum_{j=1}^{N^l_{out}} W^l_{ji}\, \delta y^l_j \\ &= \sum_{j=1}^{N^l_{out}} W^l_{ji}\, \frac{\partial Cost}{\partial x^l_j} \frac{\partial x^l_j}{\partial y^l_j} = \sum_{j=1}^{N^l_{out}} W^l_{ji}\, f'(y^l_j)\, \frac{\partial Cost}{\partial x^l_j} = \sum_{j=1}^{N^l_{out}} W^l_{ji}\, f'(y^l_j)\, \delta x^l_j, \end{aligned} \tag{B.3}$$

where $Cost$ is the cost function. The sum in equation B.3 $\sum_{j=1}^{N_{out}}$ takes into account all the neural units in the l^{th} layer connected, and so affected, by the value of x^{l-1}_i.

Glorot and Bengio (2010) propose to control the decrease and the growth of the variance observed by Bradley (2010) for $\delta x^l_i\, \forall\, i, l$. We can see a that δx^l_i in equation (B.3) is expressed in a recursive way as $\delta x^{l-1}_i = \sum_{j=1}^{N_{out}} W^l_{ji}\, f'(y^l_j)\, \delta x^l_j$. So in order to derive the expression of the variance for δx^{l-1}_i, Glorot and Bengio (2010) make some assumptions. even though they consider neural networks with tanh and sigmoidal activation functions, they assume that, at the time of initialization, all the units are operating in the linear regime, i.e, $f(x) \approx x$, and so, $f'(y^l_j) \approx 1$. All the units in a given l^{th} layer are assumed to be independently and identically distributed (and as a consequence they have the same variance). The, the pre-activations are accordingly identically and independently distributed as the units are assumed to be operating in a linear regime. The biases vectors are initialized to the same constant ($b^l_j = 0 \,\forall\, j,\, l$) so that $\mathrm{Var}(b^l_j) = 0,\, \forall\, j, l$. The authors also assume that δx^l_j and W^l_{ji} are independent of each other as well as W^l_{ji} and x^{l-1}_i are also independent. The activations are supposed to be centered around 0. The same assumptions apply to δx^l_i and W^l_{ji}, i.e, $\mathbb{E}(\delta x^l_i) = 0,\, \forall\, i, l$ and $\mathbb{E}(W^l_{ji}) = 0,\, \forall\, i, j, l$. With this set of assumptions, the variance for δx^{l-1}_i can be expressed as :

$$\begin{aligned}
\mathrm{Var}(\delta x^{l-1}_i) &= \mathrm{Var}\left(\sum_{j=1}^{N^l_{out}} W^l_{ji}\, f'(y^l_j)\, \delta x^l_j\right) \approx \mathrm{Var}\left(\sum_{j=1}^{N^l_{out}} W^l_{ji}\, \delta x^l_j\right) = \sum_{j=1}^{N^l_{out}} \mathrm{Var}\left(W^l_{ji}\, \delta x^l_j\right) \\
&= \sum_{j=1}^{N^l_{out}} \mathrm{Var}(W^l_{ji})\, \mathrm{Var}(\delta x^l_j) + \mathbb{E}^2(W^l_{ji})\, \mathrm{Var}(\delta x^l_j) + \mathbb{E}^2(\delta x^l_j)\, \mathrm{Var}(W^l_{ji}) \\
&= \sum_{j=1}^{N^l_{out}} \mathrm{Var}(W^l_{ji})\, \mathrm{Var}(\delta x^l_j) + 0 \times \mathrm{Var}(\delta x^l_j) + 0 \times Var(W^l_{ji}) \\
&= \sum_{j=1}^{N^l_{out}} \mathrm{Var}(W^l_{ji})\, \mathrm{Var}(\delta x^l_j) \\
&= \sum_{j=1}^{N^l_{out}} \mathrm{Var}(W^l_{ji}) \sum_{k=1}^{N^{l+1}_{out}} \mathrm{Var}(W^{l+1}_{kj})\, \mathrm{Var}(\delta x^{l+1}_k) \\
&= \sum_{j=1}^{N^l_{out}} \mathrm{Var}(W^l_{ji}) \sum_{k=1}^{N^{l+1}_{out}} \mathrm{Var}(W^{l+1}_{kj}) \,\cdots\, \sum_{q=1}^{N^L_{out}} \mathrm{Var}(W^L_{qp})\, \mathrm{Var}(\delta x^L_q) \\
&= N^l_{out}\, \mathrm{Var}(W^l_{ji})\; N^{l+1}_{out}\, \mathrm{Var}(W^{l+1}_{kj}) \;\cdots\; N^L_{out}\, \mathrm{Var}(W^L_{qp})\, \mathrm{Var}(\delta x^L_q) \\
&= \left[\mathrm{Var}(\delta x^L) \prod_{layer=l}^{L} N^{layer}_{out} \mathrm{Var}(W^{layer})\right] \quad \text{(ommited indices for simplicity)}.
\end{aligned} \tag{B.4}$$

We can then write the variance of the activation's gradient in the first layer of the network in terms of the gradient's variance in the last layer as :

$$\mathrm{Var}(\delta x^1) = \mathrm{Var}(\delta x^L) \prod_{l=2}^{L} N^l_{out} \mathrm{Var}(W^l). \tag{B.5}$$

To avoid the vanishing of the gradient's variance during backpropagation, the key is to keep the product term in equation (B.5) equal to a constant, for instance $\prod_{l=1}^{L} N^l_{out} \mathrm{Var}(W^l) = 1$. This constraint is a way to keep the variance stable during the backpropagation. In order to have $\prod_{l=1}^{L} N^l_{out} \mathrm{Var}(W^l) = 1$, a basic solution would be to constraint $N^l_{out} \mathrm{Var}(W^l) = 1 \,\forall\, l$. In that case, the variance of the weights in each layer would then be :

$$N^l_{out} \mathrm{Var}(W^l) = 1 \,\forall\, l \Rightarrow \mathrm{Var}(W^l) = \frac{1}{N^l_{out}} \,\forall\, l. \tag{B.6}$$

Equation (B.5) defines the constraints on the variance of the weights so that the variance of the gradient, or backward signal, does not explode or vanish. Let's now investigate the Glorot criterion so that the variance of the activation, or forward signal, does neither vanish nor explode.

For the variance of the forward signal, we make the same set of assumptions that we used in the case of the gradient signal. Equation B.2 gives the expression of the activation units in the l^{th} layer. The variance of the j^{th} unit in the l^{th} layer is then given by :

$$\begin{aligned}
\mathrm{Var}(x_j^l) &= \mathrm{Var}\left[f(y_j^l)\right] \approx \mathrm{Var}\left(y_j^l\right) = \mathrm{Var}\left(\sum_{i=1}^{N_{in}^l} W_{ji}^l\, x_i^{l-1} + b_j^l\right) \\
&= \mathrm{Var}\left(\sum_{i=1}^{N_{in}^l} W_{ji}^l\, x_i^{l-1}\right) + \mathrm{Var}(b_j^l) = \mathrm{Var}\left(\sum_{i=1}^{N_{in}^l} W_{ji}^l\, x_i^{l-1}\right) \\
&= \sum_{i=1}^{N_{in}^l} \mathrm{Var}(W_{ji}^l\, x_i^{l-1}) \\
&= \sum_{j=1}^{N_{in}^l} \mathrm{Var}(W_{ji}^l)\,\mathrm{Var}(x_i^{l-1}) + \mathbb{E}^2(W_{ji}^l)\,\mathrm{Var}(x_i^{l-1}) + \mathbb{E}^2(x_i^{l-1})\,\mathrm{Var}(W_{ji}^l) \\
&= \sum_{j=1}^{N_{in}^l} \mathrm{Var}(W_{ji}^l)\,\mathrm{Var}(x_i^{l-1}) + 0 \times \mathrm{Var}(x_i^{l-1}) + 0 \times Var(W_{ji}^l) \qquad \text{(B.7)} \\
&= \sum_{j=1}^{N_{in}^l} \mathrm{Var}(W_{ji}^l)\,\mathrm{Var}(x_i^{l-1}) \\
&= \sum_{j=1}^{N_{in}^l} \mathrm{Var}(W_{ji}^l) \sum_{k=1}^{N_{in}^{l-1}} \mathrm{Var}(W_{kj}^{l-1}) \,\cdots\, \sum_{q=1}^{N_{in}^2} \mathrm{Var}(W_{qp}^2)\,\mathrm{Var}(x_q^1) \\
&= N_{in}^l\,\mathrm{Var}(W_{ji}^l)\; N_{in}^{l-1}\,\mathrm{Var}(W_{kj}^{l-1}) \;\cdots\; N_{in}^2\,\mathrm{Var}(W_{qp}^2)\,\mathrm{Var}(x_q^1) \\
&= \left[\mathrm{Var}(x^1) \prod_{layer=1}^{l} N_{in}^{layer}\mathrm{Var}(W^{layer})\right] \quad \text{(ommited indices for simplicity)}.
\end{aligned}$$

Finally we obtain the variance of the activations in the last layer of the network in terms of the variance of the activations in the first layer by setting $l = L$. We can then express as :

$$\mathrm{Var}(x^L) = \mathrm{Var}(x^1) \prod_{l=l}^{L} N_{in}^l \mathrm{Var}(W^l). \qquad \text{(B.8)}$$

Similar to what was done in equation (B.5), to avoid the vanishing and exploding of the variance in (B.8), Glorot and Bengio (2010) choose to constrain the value of the product in (B.8) to 1, i.e, $\prod_{l=l}^{L} N_{in}^l \mathrm{Var}(W^l) = 1$. This provides the second constraint for the variance of the weights at each layer :

$$N_{in}^l \mathrm{Var}(W^l) = 1\ \forall\, l \Rightarrow \mathrm{Var}(W^l) = \frac{1}{N_{in}^l}\ \forall\, l. \qquad \text{(B.9)}$$

Equation (B.9) defines the constraints for the variance of the weights so that the variance of the forward signal does not explode and does not vanish. Combining constraints B.6 and B.9 for the variance of the weights, Glorot and Bengio (2010) propose the following trade-off between the two :

$$\mathrm{Var}(W^l) = \frac{2}{N_{in}^l + N_{out}^l} \; \forall \, l, \tag{B.10}$$

where the distribution of W^l has to be centred around 0, i.e, $\mathbb{E}(W^l) = 0$.

The He initialization

Initialization methods became even more popular with the introduction of the He initialization (He et al., 2015). Here, the weights are initialized, memorizing, in parallel, the size of the previous layer. This helps to achieve an overall minimum of the cost function more quickly and efficiently. The weights are always random but their range varies according to the size of the previous layer. This provides controlled initialization resulting in a faster and more efficient gradient descent. The initialization in He et al. (2015) is based on a set of assumptions similar to those in Glorot and Bengio (2010). He et al. (2015) use the Rectifier Linear Unit (Hahnloser, 1998), commonly called ReLU, as activation function. The ReLU activation function is defined as :

$$f(x) = \mathrm{ReLU}(x) = x^+ = \max(0, x) = \begin{cases} x & \forall \, x \in [0, +\infty[\,, \\ 0 & \text{otherwise.} \end{cases} \tag{B.11}$$

As a consequence of the use of ReLU, the activation derivative will be defined as :

$$f'(y_j^l) = \begin{cases} 1 & \forall \; y_j^l \in [0, +\infty[\,, \\ 0 & \text{otherwise.} \end{cases} \tag{B.12}$$

So instead of assuming that all the units of the model are operating in the linear regime at initialization time, it is rather assumed that the distributions of both y_i^l and δy_i^l are symmetric around 0. This means the mean and the median both coincide in 0 for both y_i^l and δy_i^l. We will show that, as a consequence of this assumption, have $\mathrm{Var}\left(x_i^l\right) = \mathbb{E}\left[(x_i^l)^2\right] = \frac{\mathrm{Var}(y_i^l)}{2}$ and $\mathrm{Var}\left(\delta y_i^l\right) = \mathbb{E}\left[(\delta y_i^l)^2\right] = \frac{\mathrm{Var}(\delta x_i^l)}{2}$. Besides the set of assumptions in the context of the Glorot initialization, The following assumptions are made for any integers i, j, l : δy_i^l are identically and independently distributed and their distribution is symmetric around 0 ; δy_j^l and W_{ji}^l are mutually independent ; $f'(y_j^l)$ and δx_j^l are mutually independent. With this set of assumptions, the variance for δx_i^{l-1} can be expressed as :

$$
\begin{aligned}
\mathrm{Var}(\delta x_i^{l-1}) &= \mathrm{Var}\left[\sum_{j=1}^{N_{out}^l} W_{ji}^l\, \delta y_j^l\right] = \sum_{j=1}^{N_{out}^l} \mathrm{Var}(W_{ji}^l\, \delta y_j^l) = \sum_{j=1}^{N_{out}^l} \mathrm{Var}(W_{ji}^l\, \delta y_j^l) \\
&= \sum_{j=1}^{N_{out}^l} \mathrm{Var}(W_{ji}^l)\, \mathrm{Var}(\delta y_j^l) + \mathbb{E}^2(\delta y_j^l)\, \mathrm{Var}(W_{ji}^l) + \mathbb{E}^2(W_{ji}^l)\, \mathrm{Var}(\delta y_j^l) \\
&= \sum_{j=1}^{N_{out}^l} \mathrm{Var}(W_{ji}^l)\, \mathrm{Var}(\delta y_j^l) + \mathbb{E}^2(\delta y_j^l)\, \mathrm{Var}(W_{ji}^l) + 0 \times \mathrm{Var}(\delta y_j^l) \\
&= \sum_{j=1}^{N_{out}^l} \mathrm{Var}(W_{ji}^l)\, \mathrm{Var}(\delta y_j^l) + \mathbb{E}^2(\delta y_j^l)\, \mathrm{Var}(W_{ji}^l) \\
&= \sum_{j=1}^{N_{out}^l} \mathrm{Var}(W_{ji}^l)\, \left[\mathrm{Var}(\delta y_j^l) + \mathbb{E}^2(\delta y_j^l)\right] \\
&= \sum_{j=1}^{N_{out}^l} \mathrm{Var}(W_{ji}^l)\, \left[\mathrm{Var}\left(f'(y_j^l)\, \delta x_j^l\right) + \mathbb{E}^2\left(f'(y_j^l)\, \delta x_j^l\right)\right].
\end{aligned}
\tag{B.13}
$$

In equation B.13 we expressed $\mathrm{Var}(\delta x_i^{l-1})$ in terms of $\mathrm{Var}\left[f'(y_j^l)\, \delta x_j^l\right]$ and $\mathbb{E}^2\left[f'(y_j^l)\, \delta x_j^l\right]$. It is important to notice that this is exactly where the He initialization is mostly different from the Glorot one. He et al. (2015) take into account the activation's derivative in the computation of the variance of the gradient. Glorot and Bengio (2010) assumed that the activation's derivative is approximately equal to one during initialization time, which is not necessarily true. In fact, In the case of tanh and sigmoidal activations, the activation's derivative is exactly equal to one only when the pre-activation is at the origin. He et al. (2015) on the other hand, leveraged the use of ReLU as an activation as it is trivial to compute its statistics (mean and variance). $\mathrm{Var}(\delta y_j^l)$ can be now written as :

$$
\begin{aligned}
\mathrm{Var}(\delta y_j^l) &= \mathrm{Var}\left[f'(y_j^l)\, \delta x_j^l\right] \\
&= \mathrm{Var}(f'(y_j^l))\, \mathrm{Var}(\delta x_j^l) + \mathbb{E}^2(\delta x_j^l)\mathrm{Var}(f'(y_j^l)) + \mathbb{E}^2(f'(y_j^l))\mathrm{Var}(\delta x_j^l) \\
&= \mathrm{Var}(f'(y_j^l))\, \mathrm{Var}(\delta x_j^l) + 0 \times \mathrm{Var}(f'(y_j^l)) + \mathbb{E}^2(f'(y_j^l))\mathrm{Var}(\delta x_j^l) \\
&= \mathrm{Var}(\delta x_j^l)\left[\mathrm{Var}(f'(y_j^l)) + \mathbb{E}^2(f'(y_j^l))\right].
\end{aligned}
\tag{B.14}
$$

In order to get the expression of $\left[\mathrm{Var}(f'(y_j^l)) + \mathbb{E}^2(f'(y_j^l))\right]$ we have to compute $\mathbb{E}(f'(y_j^l))$ and then $\mathrm{Var}(f'(y_j^l))$.

$$
\mathbb{E}(f'(y_j^l)) = \int_{-\infty}^{+\infty} p(y_j^l) f'(y_j^l)\, dy_j^l = \int_{0}^{+\infty} p(y_j^l) \times 1\, dy_j^l = \frac{1}{2}. \tag{B.15}
$$

As a consequence $\mathbb{E}^2(f'(y_j^l)) = \frac{1}{4}$. The variance $\mathrm{Var}(f'(y_j^l))$ can then be computed :

$$\begin{aligned}
\mathrm{Var}(f'(y_j^l)) &= \int_{-\infty}^{+\infty} p(y_j^l) \left[f'(y_j^l) - \mathbb{E}(f'(y_j^l))\right]^2 dy_j^l \\
&= \int_{-\infty}^{0} p(y_j^l) \times \frac{1}{4}\, dy_j^l + \int_{0}^{+\infty} p(y_j^l) \times \frac{1}{4}\, dy_j^l \\
&= \frac{1}{4} \int_{-\infty}^{+\infty} p(y_j^l)\, dy_j^l = \frac{1}{4}.
\end{aligned} \tag{B.16}$$

Leveraging the results obtained in equations (B.15) and (B.16) we can express equation (B.14) as :

$$\mathrm{Var}(\delta y_j^l) = \frac{1}{2}\,\mathrm{Var}(\delta x_j^l). \tag{B.17}$$

Having obtained the expression of $\mathrm{Var}(\delta y_j^l)$ we can use it later in equation (B.13). We will now deduce the expression for $\mathbb{E}^2(\delta y_j^l)$.

$$\mathbb{E}(\delta y_j^l) = \mathbb{E}\left[f'(y_j^l)\,\delta x_j^l\right] = \mathbb{E}\left[f'(y_j^l)\right]\mathbb{E}\left[\delta x_j^l\right] = \frac{1}{2} \times 0 = 0. \tag{B.18}$$

Given the results found in equations (B.17) and (B.18), we can replace $\mathrm{Var}(\delta y_j^l)$ and $\mathbb{E}^2(\delta y_j^l)$ in equation (B.13) and express $\mathrm{Var}(\delta x_i^{l-1})$ as :

$$\begin{aligned}
\mathrm{Var}(\delta x_i^{l-1}) &= \sum_{j=1}^{N_{out}^l} \mathrm{Var}(W_{ji}^l)\left[\mathrm{Var}(\delta y_j^l) + \mathbb{E}^2(\delta y_j^l)\right] \\
&= \sum_{j=1}^{N_{out}^l} \mathrm{Var}(W_{ji}^l)\,\frac{1}{2}\,\mathrm{Var}(\delta x_j^l) \\
&= \sum_{j=1}^{N_{out}^l} \mathrm{Var}(W_{ji}^l)\,\frac{1}{2} \sum_{k=1}^{N_{out}^{l+1}} \mathrm{Var}(W_{kj}^{l+1}) \;\cdots\; \frac{1}{2} \sum_{q=1}^{N_{out}^{L}} \mathrm{Var}(W_{qp}^L)\,\mathrm{Var}(\delta x_q^L) \\
&= N_{out}^l\,\mathrm{Var}(W_{ji}^l)\,\frac{1}{2}\,N_{out}^{l+1}\,\mathrm{Var}(W_{kj}^{l+1}) \;\cdots\; \frac{1}{2}\,N_{out}^L\,\mathrm{Var}(W_{qp}^L)\,\frac{1}{2}\,\mathrm{Var}(\delta x_q^L) \\
&= \left[\mathrm{Var}(\delta x^L) \prod_{layer=l}^{L} N_{out}^{layer}\,\frac{1}{2}\,\mathrm{Var}(W^{layer})\right] \quad (\text{ommited indices for simplicity}).
\end{aligned} \tag{B.19}$$

We can then write the $\mathrm{Var}(\delta x_i^{l-1})$ in terms of $\mathrm{Var}(\delta x_q^L)$. We can express it as :

$$\mathrm{Var}(\delta x^1) = \mathrm{Var}(\delta x^L) \prod_{l=2}^{L} N_{out}^l\,\frac{1}{2}\,\mathrm{Var}(W^l). \tag{B.20}$$

Equation (B.20) is very similar to the gradient's variance constraint found for the Glorot initialization and expressed in equation (B.5). The only difference remains in the multiplicative factor $\frac{1}{2}$

expressed inside the product of variances. As for the Glorot initialization a simple solution to avoid vanishing and exploding variance during backpropagation is to set the expression inside the product equal to 1. We would then obtain :

$$N_{out}^l \, \frac{1}{2} \, \mathrm{Var}(W^l) = 1 \; \forall \, l \Rightarrow \mathrm{Var}(W^l) = \frac{2}{N_{out}^l} \; \forall \, l. \tag{B.21}$$

Let's now investigate the He criterion for the forward signal. The set of assumption made in the backward case is maintained.

$$\begin{aligned}
\mathrm{Var}\left(y_j^l\right) &= \mathrm{Var}\left(\sum_{i=1}^{N_{in}^l} W_{ji}^l \, x_i^{l-1} + b_j^l\right) \\
&= \mathrm{Var}\left(\sum_{i=1}^{N_{in}^l} W_{ji}^l \, x_i^{l-1}\right) + \mathrm{Var}(b_j^l) = \mathrm{Var}\left(\sum_{i=1}^{N_{in}^l} W_{ji}^l \, x_i^{l-1}\right) \\
&= \sum_{i=1}^{N_{in}^l} \mathrm{Var}(W_{ji}^l \, x_i^{l-1}) \\
&= \sum_{j=1}^{N_{in}^l} \mathrm{Var}(W_{ji}^l) \, \mathrm{Var}(x_i^{l-1}) + \mathbb{E}^2(W_{ji}^l) \, \mathrm{Var}(x_i^{l-1}) + \mathbb{E}^2(x_i^{l-1}) \, \mathrm{Var}(W_{ji}^l) \\
&= \sum_{j=1}^{N_{in}^l} \mathrm{Var}(W_{ji}^l) \, \mathrm{Var}(x_i^{l-1}) + 0 \times \mathrm{Var}(x_i^{l-1}) + \mathbb{E}^2(x_i^{l-1}) \times Var(W_{ji}^l) \\
&= \sum_{j=1}^{N_{in}^l} \mathrm{Var}(W_{ji}^l) \, \left[\mathrm{Var}(x_i^{l-1}) + \mathbb{E}^2(x_i^{l-1})\right] = \sum_{j=1}^{N_{in}^l} \mathrm{Var}(W_{ji}^l) \, \mathbb{E}\left[(x_i^{l-1})^2\right].
\end{aligned} \tag{B.22}$$

In order to express $\mathrm{Var}\left(y_j^l\right)$ in terms of $\mathrm{Var}\left(y_j^{l-1}\right)$, we have to express $\mathbb{E}\left[(x_i^{l-1})^2\right]$ in terms of y_j^{l-1}.

$$\begin{aligned}
\mathbb{E}\left[(x_i^{l-1})^2\right] &= \int_{-\infty}^{+\infty} p(x_i^{l-1})[x_i^{l-1}]^2 \, dx_i^{l-1} = \int_{-\infty}^{0} p(y_j^l) \times 0 \, dx_i^{l-1} + \int_{0}^{+\infty} p(x_i^{l-1})[x_i^{l-1}]^2 \, dx_i^{l-1} \\
&= \int_{0}^{+\infty} p(y_i^{l-1})[y_i^{l-1}]^2 \, dy_i^{l-1} = \int_{0}^{+\infty} p(y_i^{l-1})[y_i^{l-1} - 0]^2 \, dy_i^{l-1} \\
&= \frac{1}{2} \, \mathrm{Var}(y_i^{l-1}).
\end{aligned} \tag{B.23}$$

We can replace $\mathbb{E}\left[(x_i^{l-1})^2\right]$ by $\frac{1}{2} \, \mathrm{Var}(y_j^{l-1})$ in equation (B.22) to obtain the He criterion for the forward propagated signal :

$$\begin{aligned}
\mathrm{Var}\left(y_j^l\right) &= \sum_{j=1}^{N_{in}^l} \mathrm{Var}(W_{ji}^l)\,\frac{1}{2}\,\mathrm{Var}(y_i^{l-1}) \\
&= \sum_{j=1}^{N_{in}^l} \mathrm{Var}(W_{ji}^l)\,\frac{1}{2}\,\sum_{k=1}^{N_{in}^{l-1}} \mathrm{Var}(W_{kj}^{l-1})\,\cdots\,\frac{1}{2}\,\sum_{q=1}^{N_{in}^2} \mathrm{Var}(W_{qp}^2)\,\frac{1}{2}\,\mathrm{Var}(y_q^1) \\
&= N_{in}^l\,\mathrm{Var}(W_{ji}^l)\,\frac{1}{2}\,N_{in}^{l-1}\,\mathrm{Var}(W_{kj}^{l-1})\,\cdots\,\frac{1}{2}\,N_{in}^2\,\mathrm{Var}(W_{qp}^2)\,\frac{1}{2}\,\mathrm{Var}(y_q^1) \\
&= \left[\mathrm{Var}(y^L)\,\prod_{layer=l}^{l} N_{in}^{layer}\,\frac{1}{2}\,\mathrm{Var}(W^{layer})\right] \quad \text{(ommited indices for simplicity)}.
\end{aligned} \tag{B.24}$$

Equation (B.24) provides the initialization constraint to ensure the propagation of the signal. Thiss is expressed as :

$$N_{in}^l\,\frac{1}{2}\,\mathrm{Var}(W^l) = 1\;\forall\,l \Rightarrow \mathrm{Var}(W^l) = \frac{2}{N_{in}^l}\;\forall\,l. \tag{B.25}$$

We have expressed the He criteria for both the forward and the backward signals respectively in equations (B.21) and (B.25). In practice, one of either one of these 2 criteria is used during the intialization of the parameters.