Complex-valued neural networks for radar applications

Jose Agustin Barrachina

université
PARIS-SACLAY

Complex-Valued Neural Networks for Radar Applications

Réseau de neurones à valeurs complexes pour les applications Radar

Thèse de doctorat de l'Université Paris-Saclay

École doctorale n° 580, Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat : Traitement du signal et des images
Graduate School: Informatique et sciences du numérique,
Référent : CentraleSupélec

Thèse préparée dans SONDRA (Université Paris-Saclay, CentraleSupélec, ONERA), sous la direction de **Jean-Philippe OVARLEZ**, Directeur de Recherche et le co-encadrement de **Chengfang REN**, Maître de Conférence et le co-encadrement de **Christèle MORISSEAU**, Ingénieur de Recherche

Thèse soutenue à Paris-Saclay, le 06 Decembre 2022, par

Jose Agustin BARRACHINA

Composition du jury

Laurent FERRO-FAMIL Professeur des Universités, ISAE	Président & examinateur
Florence TUPIN Professeur des Universités, Telecom Paris - INP	Rapporteur & Examinatrice
Alexandre BENOIT Professeur des Universités, LISTIC, Université Savoie Mont-Blanc	Rapporteur
Mihai DATCU Professeur, Politehnica University of Bucharest	Examinateur
Jérémy FIX Maître de Conférence, LORIA CentraleSupélec	Examinateur

THESE DE DOCTORAT
NNT : 2022UPASG094

To my grandmother Blanca Calviño
who inspires me every day
with insightful talks and advice

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Réseau de neurones à valeurs complexes pour les applications Radar
Mots clés : Réseau de Neurones à Valeurs Complexes, Radar à synthèse d'ouverture, classification, segmentation sémantique

Résumé :

Le traitement des signaux radars et des images SAR nécessite généralement des représentations et des opérations à valeurs complexes, telles que les transformées de Fourier et d'ondelettes, les filtres de Wiener et les filtres adaptés, etc. Cependant, la grande majorité des architectures d'apprentissage profond sont actuellement basées sur des opérations à valeurs réelles, ce qui limite leur capacité d'apprentissage à partir de données complexes. Malgré l'émergence des réseaux de neurones à valeurs complexes (CVNN), leur application au radar et à l'imagerie SAR manque encore d'études sur leur pertinence et leur efficacité. Et la comparaison avec un réseau de neurones à valeurs réelles (RVNN) équivalent est généralement biaisée.

Dans cette thèse, nous proposons d'étudier les mérites des CVNNs pour classifier des données complexes. Nous montrons que les CVNNs atteignent de meilleures performances que leur equivalent réel pour classifier des vecteurs à données gaussiennes non circulaires. Nous définissons également un critère d'équivalence entre les CVNNs et les RVNNs, entièrement connectés ou convolutifs, en termes du nombre de paramètres entraînables, tout en leur conservant une architecture similaire. Nous comparons ainsi statistiquement les performances de perceptrons multicouches (MLPs), de réseaux convolutifs (CNNs) et entièrement convolutifs (FCNNs) utilisés pour la segmentation d'images SAR polarimétriques. Le partitionnement des images SAR et l'équilibrage des classes sont étudiés afin d'éviter des biais d'apprentissage. En parallèle, nous avons également proposé une librairie open-source pour faciliter l'implémentation des CVNNs et la comparaison avec des réseaux équivalents réels.

Title: Complex-Valued Neural Networks for Radar Applications
Keywords: Complex-Valued Neural Networks, Polarimetric Synthetic Aperture Radar, Classification, Semantic Segmentation

Abstract:

Radar signal and SAR image processing generally require complex-valued representations and operations, e.g., Fourier, wavelet transforms, Wiener, matched filters, etc. However, the vast majority of architectures for deep learning are currently based on real-valued operations, which restrict their ability to learn from complex-valued features. Despite the emergence of Complex-Valued Neural Networks (CVNNs), their application on radar and SAR still lacks study on their relevance and efficiency. And the comparison against an equivalent Real-Valued Neural Network (RVNN) is usually biased.

In this thesis, we propose to investigate the merits of CVNNs for classifying complex-valued data. We show that CVNNs achieve better performance than their real-valued counterpart for classifying non-circular Gaussian data. We also define a criterion of equivalence between feed-forward fully connected and convolutional CVNNs and RVNNs in terms of trainable parameters while keeping a similar architecture. We statistically compare the performance of equivalent Multi-Layer Perceptrons (MLPs), Convolutional Neural Networks (CNNs), and Fully Convolutional Networks (FCNs) for polarimetric SAR image segmentation. SAR image splitting and balancing classes are also studied to avoid learning biases. In parallel, we also proposed an open-source toolbox to facilitate the implementation of CVNNs and the comparison with real-equivalent networks.

Acknowledgements

I first want to extend my gratitude to all members of the jury for accepting to be part of my work and taking the time and dedication to evaluate my work, in particular to Florence Tupin and Alexandre Benoit for their insightful comments and reviews.

I feel grateful to have had the opportunity to work with each of my supervisors, all of whom I admire deeply, and I feel proud to have learned a lot from them. They are:

- Jean-Philippe Ovarlez, Director, Directeur de Recherche, ONERA Palaiseau (DEMR) and CentraleSupélec, Université Paris-Saclay (SONDRA), for accepting me for this Ph.D. topic and for giving me a place to stay when arriving in France. He was an invaluable support both from the academic and the administrative point of view, always open to helping me in every aspect.
- Chengfang REN, Maître de conférences, CentraleSupélec, Université Paris-Saclay (SONDRA), whose assistance was a milestone in the completion of this project.
- Christèle Morisseau, Ingénieur de Recherche, ONERA Palaiseau (DEMR), whose insights enriched my knowledge as a professional.
- Gilles Vieillard, Ingénieur de Recherche, ONERA Palaiseau (DEMR), whose insight was of great value to my work and served as guidance to what and where to look. Unfortunately, due to a change in the doctoral school that took place two months before the publication of my manuscript, he could not appear on the front page of this manuscript. Still, he contributed as much as the rest of my supervisors.

I want to express my deepest appreciation to all my colleagues at DEMR (ONERA) and SONDRA, who were always very welcoming and open. In particular to Thierry Letertre, who lent me his computer to work on during the first weeks at SONDRA and Israel Hinostroza for his insightful comments and recommendation about useful complex-algebra references, to Stephane Saillant, who, by his invaluable encouragement, always helps and motivates the team forward by supporting the team's ideas and proposals, a sign of a good leader and the reason for SONDRA's success, and Virginie Bouvier, who was with me during all the administrative process I had to undertake as well as Isabelle Dulon who integrated quickly as part of SONDRA's team.

I owe my deepest gratitude to my wife, Catalina Carabajal, for all her support and containment when I most needed it, being a safe haven where I could be myself, helping me to fight my battles and fighting alongside me. She is the main reason that I became the person I am today, for which I am very proud.

I want to thank my family, without whom I would have never achieved this, and special gratitude to my parents for constantly pushing me to achieve more and motivating me to pursue this Ph.D. thesis. If I could see further is because I stepped on giant shoulders. And both my grandmothers, Elena Tejada and Blanca Calviño, who I always knew, loved me and supported me unconditionally. Finally, to my grandfathers, Pedro Gervasoni and Ángel Barrachina, and my uncle, Luis Gervasoni, I know would be very proud of me.

Institutions

This work was done in the SONDRA laboratory and ONERA. Several institutions provided funding making this work possible. In particular, I would like to thank ONERA and DGA for funding my scholarship at $50\% - 50\%$ and SONDRA for providing me with the equipment and covering the costs of inscription and travel for all my conferences. Finally, CentraleSupélec provided the GPU servers that allowed me to perform the simulations needed [Fix et al., 2022].

In what follows, a description of the institutions involved.

SONDRA, a laboratory born from the joint collaboration of CentraleSupélec, ONERA, NUS and the DSO National Laboratories. Its acronym represents a contraction of Supélec (now CentraleSupélec), ONERA, NUS, and DSO Research Alliance. SONDRA represents the union between Singapore and France around fundamental research activities in the fields of electromagnetism and signal processing applied to radar. SONDRA started in 2004 with unclassified basic research in advanced Electromagnetism and Radar domains.

ONERA (Office National d'Études et de Recherches Aérospatiales) -The French Aerospace Lab- is the French national aerospace research center, best described as DSO's equivalent in France. It is a public establishment with industrial and commercial operations and carries out application-oriented research to support enhanced innovation and competitiveness in the aerospace and defense sectors.

DGA (Direction Générale de l'Armement) is expertise, testing, and engineering force within the Ministry of the Armed Forces, the DGA is responsible for equipping armies in a sovereign manner, preparing for the future of defense systems, promote European cooperation and support exports. Since its creation in 1961 by General de Gaulle, the DGA has conducted an average of more than a

hundred armament operations per year, covering the entire range of equipment of the highest technological level necessary for the French armies to carry out their missions. It oversees the project management of weapon systems throughout the life of the programs and prepares the future of defense systems with the aim of ensuring the independence of France and its strategic autonomy.

CentraleSupélec (CS) is a prestigious post-graduate French engineering school under the ministerial charter devoted to the sciences and engineering. Ranked among the best universities in the world by QS. CentraleSupélec was officially established on January 1st, 2015, bringing together two leading engineering schools in France; École Centrale Paris and Supélec. The cooperation between these two *Grandes Écoles*, as they are known in the French system, had progressively been gaining momentum since 2009, with the sustained alliance in three core areas: engineering education, executive education, and research.

DSO National Laboratories (Defence Science Organisation) is Singapore's largest defense research and development (R&D) organization, with the critical mission to develop technological solutions to sharpen the cutting edge of Singapore's national security. In 2019, DSO won 3 awards from DTP, the most prestigious award presented annually by the Ministry of Defence (MINDEF) to outstanding teams and individuals who have contributed significantly towards enhancing the defense capabilities of Singapore.

NUS (The National University of Singapore) is a national research university in Singapore. Founded in 1905, NUS is the oldest autonomous university in the country. It offers degree programs in a wide range of disciplines at both the undergraduate and postgraduate levels, including in the sciences, medicine and dentistry, design and environment, law, arts, and social sciences, engineering, business, computing, and music. NUS has consistently been considered as being one of the most prestigious academic institutions in the world as well as in Asia itself. It has consistently featured in the world's top 100 universities as ranked by the Academic Ranking of World Universities, the QS World University Rankings, and the Times Higher Education World University Rankings.

Contents

Acronyms

1HL one hidden layer. 18, 82, 83, 85–87, 89–91

2HL two hidden layers. 18, 83, 85–89

AA Average Accuracy. 19, 120, 121, 123, 125, 128, 135

AI Artificial Intelligence. 17, 29, 30, 35–37

API Application Programming Interface. 50, 52

autodiff automatic differentiation. 40, 47, 59, 163, 165–168

BN Batch Normalization. 45, 64, 117, 118

CN Complex Normal. 84

CNN Convolutional Neural Network. 25, 35, 36, 44, 82, 104, 110, 115, 120, 121, 135, 139, 145

CV-CAE Complex-Valued Convolutional AutoEncoder. 25, 145

CV-CNN Complex-Valued Convolutional Neural Network. 25, 76, 110, 116, 122, 145

CV-FCNN Complex-Valued Fully Convolutional Neural Network. 18, 19, 25, 76, 110, 117, 120, 122–127, 130, 133–135, 139, 145, 146

CV-MLP Complex-Valued MultiLayer Perceptron. 24, 25, 74–76, 109–112, 116, 122, 145

CV-UNET Complex-Valued U-Network. 76

CVNN Complex-Valued Neural Network. 18, 23–26, 47, 48, 53, 58, 59, 61, 63, 75–77, 79, 83–91, 109–113, 116, 118, 120, 125, 128, 134–137, 139–141, 143–147, 154

EM ElectroMagnetic. 18, 93, 97

FCNN Fully Convolutional Neural Network. 21, 25, 36, 104, 115, 120, 121, 124, 125, 127, 133–135, 139, 145

GAN Generative Adversarial Network. 140

ML Machine Learning. 29

MLP MultiLayer Perceptron. 18, 21, 29, 35, 36, 38, 74, 79, 82, 83, 104, 111, 113, 115, 116, 119–121, 135, 139

OA Overall Accuracy. 18, 19, 120–123, 125, 128–130, 134, 136

PolInSAR Polarimetric and Interferometric Synthetic Aperture Radar. 132

PolSAR Polarimetric Synthetic Aperture Radar. 24–27, 36, 95, 96, 102–104, 109, 110, 116, 118, 119, 121, 124, 129, 131, 132, 135, 137, 139, 140, 144–147

PSK Phase-Shift Keying. 68

QAM Quadrature Amplitude Modulation. 68

ReLU Rectified Linear Unit. 17, 41, 42, 44, 61, 62, 73, 116–118, 163, 164, 166

RV-CNN Real-Valued Convolutional Neural Network. 25, 110, 122, 145

RV-FCNN Real-Valued Fully Convolutional Neural Network. 19, 122–127, 130, 133–135, 139

RV-MLP Real-Valued MultiLayer Perceptron. 24, 25, 74, 75, 109–112, 116, 122, 145

RVNN Real-Valued Neural Network. 18, 23–26, 50, 53, 75–77, 79, 83–91, 103, 109–113, 118, 128, 135–137, 139, 140, 143–145, 147

SAR Synthetic Aperture Radar. 18, 24, 26, 93–96, 98, 100–104, 109, 110, 119, 139, 144, 147

SGD Stochastic Gradient Descent. 43, 44, 75, 83, 116

tanh hyperbolic tangent. 17, 40, 41, 61, 65

U-NET U-Network. 36, 52

UX User eXperience. 50

List of Figures

List of Tables

Introduction

In the machine learning community, most neural networks are developed for processing real-valued features (voice signals, RGB images, videos, etc.). The signal processing community, however, has a higher interest in developing theories and techniques in complex fields. Indeed, complex-valued signals are encountered in a large variety of applications such as biomedical sciences, physics, communications, and radar. All these fields use signal processing tools [Schreier and Scharf, 2010], which are usually based on complex filtering operations and Complex-Valued representations or features (Discrete Fourier Transform, Wavelet Transform, Wiener Filtering, Matched Filter, etc.). Thus, Complex-Valued Neural Networks (CVNNs) appear as a natural choice to process and to learn from these complex-valued features since the operation performed at each layer of CVNNs can be interpreted as complex filtering or multiplications. Notably, CVNNs are more adapted than RVNNs to extract phase information [Hirose and Yoshida, 2012]. For this reason, several signal processing fields of study already use CVNNs for their experiments, such as radio frequency signal processing in wireless communications [Gong et al., 2017, Zhang and Wu, 2017, Liu et al., 2017, Ding and Hirose, 2014, Marseet and Sahin, 2017], image processing in computer vision such as classification and segmentation tasks [Akramifard et al., 2012, Hafiz et al., 2015, Popa, 2017, Amilia et al., 2015, Liu et al., 2014, Olanrewaju et al., 2011, Gu and Ding, 2018, Matlacz and Sarwas, 2018, Popa, 2018], audio signal processing [Al-Nuaimi et al., 2012, Kinouchi and Hagiwara, 1996, Kataoka et al., 1998, Hayakawa et al., 2018, Tsuzuki et al., 2013, Tsuzuki et al., 2013], wind prediction [Sepasi et al., 2017, Çevik et al., 2018, Mandic et al., 2009], power transformers [Chistyakov et al., 2011, Minin et al., 2012], Traffic Signal Control [Nishikawa et al., 2005, Nishikawa et al., 2006], cryptography [Dong and Huang, 2019], spam detection [Hu et al., 2008], associative memory [Jankowski et al., 1996], medical applications such as epilepsy diagnosis [Peker et al., 2015] or MRI signal processing [Virtue et al., 2017] and particularly radar applications (our field of study).

In the early 90's, works on CVNN back-propagation were published [Hirose, 1992, Georgiou and Koutsougeras, 1992, Benvenuto and Piazza, 1992, Leung and Haykin, 1991], leaving ground for CVNN to be implemented. Since then, work on CVNN started to increase in the mid-90's or early 2000's [Jankowski et al., 1996, Kim and Adali, 2001b, Kim and Adali, 2001a, Kim and Adali, 2002, Miyauchi and Seki, 1992, Miyauchi et al., 1993, Jianping et al., 2002, Cha and Kassam, 1995] and in 2003, Akira Hirose, most likely the precursor of CVNN published a first book about the topic in 2003 [Hirose, 2003] and two more later on [Hirose, 2013, Hirose, 2012]. Reference [Nitta, 2004] explains the potential of a single complex-valued neuron through the concept of the orthogonal boundary where the intersection of two hyper-surfaces can divide the decision boundary into four regions, reveling

the potential computational power of CVNNs with respect to RVNN. A. Hirose explained that the merits of CVNN lie mainly in the properties of complex multiplication that can be seen as phase rotation, and amplitude modulation advocating for an advantageous reduction of freedom [Hirose, 2013, Hirose, 1992]. References [Hirose, 2009, Hirose, 2011, Hirose, 2010] discussed the merits of CVNN by relating the mathematical expressions of complex numbers with the signal processing field. Two recent surveys were published recently about CVNN *state-of-the-art* advancements [Bassey et al., 2021, Lee et al., 2022]. Both works mention the biological motivation of CVNN as they represent their behavior better.

Recently, we showed that CVNN are more performant in classifying non-circular Gaussian data than its real counterpart, which means CVNNs are more sensible to extract phase information than RVNNs. We do that by comparing vectors of random non-circular data showing that CVNN can profit from this feature and extract its full potential by achieving higher accuracy, less overfitting, and lower variance than the RVNN. Our findings were also cited by Reference [Ko et al., 2022] to justify some properties of their obtained results as they were analogous to ours.

Deep learning techniques are becoming widely popular and have extended into radar and PolSAR image classification [Fix et al., 2021, Marmanis et al., 2018, Marmanis et al., 2016b, Parikh et al., 2020, Konishi et al., 2021, Chen et al., 2016, Hou et al., 2016, Zhou et al., 2016]. Usually, these networks are fed with the amplitude information of the PolSAR image while not making use of the phase data.

Recently, some publications started using CVNNs as an alternative to conventional Real-Valued Neural Network (RVNN) for radar applications [Hirose, 2013, Bassey et al., 2021] since radar data are generally complex-valued. Knowing that Synthetic Aperture Radar (SAR) data is non-circular [El-Darymli et al., 2014, Vasile and Totir, 2012] and therefore phase information plays a crucial part in their representation [Datcu et al., 2007, El-Darymli et al., 2013, El-Darymli et al., 2015], it is no wonder that Complex-Valued Neural Networks are becoming increasingly popular for SAR, PolSAR or InSAR applications [Wilmanski et al., 2016, Oyama and Hirose, 2018, Gleich and Sipos, 2018]

Reference [Hänsch and Hellwich, 2009a], was one of the first to implement a Complex-Valued MultiLayer Perceptron (CV-MLP) for PolSAR applications in 2009. Although a comparison was made with Real-Valued MultiLayer Perceptron (RV-MLP), no confidence interval was given which prevents to assert CV-MLP merits. Furthermore, a different input representation was used for each model, making it a non-pertinent comparison. Later on, the same authors suggested giving the same input representation to get a more precise comparison between the models on [Hänsch, 2010]. References [Hänsch and Hellwich, 2010] and [De et al., 2017] also used CV-MLP on a PolSAR database but did not provide a comparison with RV-MLP. Reference [Cao et al., 2019] did compare CV-MLP against a RV-MLP

but even though CV-MLP performed better than RV-MLP, confidence intervals intersect, leaving room for doubt about CV-MLP out-performance.

Works using Complex-Valued Convolutional Neural Network (CV-CNN) have been published for PolSAR applications. Reference [Zhang et al., 2017] compares a CV-CNN with RV-CNNs but lacks confidence intervals. Other recent works [Sun et al., 2019, Zhao et al., 2019a, Zhao et al., 2019b, Qin et al., 2021, Dong et al., 2020] use a CV-CNN for PolSAR applications but without comparing its result with a RV-CNN.

References [Xie et al., 2020] and [Shang et al., 2019] added complexity to the CNN architecture by using a Recurrent Complex-Valued Convolutional Neural Network and a Complex-Valued Convolutional AutoEncoder (CV-CAE) to obtain higher accuracy results. Lately, References [Cao et al., 2019, Li et al., 2018a] achieved *state-of-the-art* performance using a Complex-Valued Fully Convolutional Neural Network (CV-FCNN) model architecture. All the previously cited works of CVNN applications on PolSAR perform a pixel-wise classification task, which can be seen as a semantic segmentation task. It is, therefore, not a surprise that a FCNN model achieves es higher accuracy as it performs semantic segmentation by design.

In all cases cited here, the dimensions of the RVNN are not justified, furthermore, the RVNN has lower capacity than the CVNN model [Mönning and Manandhar, 2018, Barrachina et al., 2021c, Barrachina et al., 2022d]. For this reason, the merits of CVNN over RVNN for PolSAR classification and semantic segmentation are not fully justified. Either the comparison against a conventional RVNN is lacking, or the models do not have the same capacity. Furthermore, some works omit confidence intervals, or the intervals intersect. In this work, we have provided a method to correctly design and dimension a RVNN so that it is equivalent to the CVNN we want to compare with. Under this framework, we perform classification tasks on two PolSAR datasets for three different complex model architectures and their corresponding real-equivalent networks and prove that for the same task and model architecture, complex networks perform better.

All the work cited above uses the polarimetric coherency matrix as an input representation of the networks. However, we compare the results to those obtained when using the Pauli vector representation and show both Complex-Valued Fully Convolutional Neural Network (CV-FCNN) and its real-equivalent model perform better when using the Pauli vector.

PolSAR studies have the difficulty that rarely more than one image can be used for classification. For that reason, training, validation, and test set must be extracted from the same image resulting in similarities between the subsets that may prevent the appreciation of the generalization performance and yield higher apparent results than those obtained under new data. We propose a new method to split the dataset in order to reduce this effect and evaluate its impact.

The difficulties in implementing CVNN models in practice have slowed down

the field from growing further [Mönning and Manandhar, 2018]. To fill this void, we implemented a *Python* toolbox that allowed the implementation of CVNN, which was made available online and documented accordingly so that the community can make use of it. Indeed, the need of the community for this toolbox was the subject of the code's success which can be seen by their *GitHub* metrics, downloads, and citations. In particular, we perform an experiment that justifies the initialization adaptation for complex layers defined by Reference [Trabelsi et al., 2017], and we manage to show the importance of using this adaptation correctly.

Chapter 1 explains the theory of conventional neural networks. We introduce with history and development of neural networks until today's breakthroughs in *state-of-the-art* algorithms. We later explain the neural network's basic operation and training phase. Finally, we explain in detail each part of the neural network algorithm and its variants. As neural networks are a very wide field, we limit the Chapter to only those advancements and theories that are relevant to our work.

Chapter 2 explains the adaptation of these conventional real networks to the complex domain and lay ground for Complex-Valued Neural Network (CVNN) implementation. We propose two experiments that prove the importance of the correct adaptation of the initialization technique to the complex domain. In parallel, we describe the published toolbox that allows the implementation of CVNN using *Tensorflow* as the back-end. A toolbox like this one was missing, for it filled a gap that was appreciated by the community, a success that can be seen by the code's citations, downloads or other repository metrics discussed in this Chapter as well. We end the Chapter with an experiment that shows that real-valued temporal signals might profit from a CVNN when using the Hilbert transform to obtain higher performance than when using a conventional real-valued model.

Chapter 3 justify the merits of CVNN over RVNN complex-valued Gaussian datasets that are non-circular, characterized by either a correlation between the real and imaginary parts or a non-identical variance for the real and imaginary part. To do so, we randomly generate multiple classes of vectors of non-circular data that have the property of changing their distribution if we rotate the random generator, contrary to circular data, which is invariant to rotation. Classify them using both CVNN and RVNN, running the simulation several times to infer statistical results. This experiment was run in different flavors, such as different sources of non-circularity, different input representations, different model architecture, hyper-parameters and dimensions, etc., showing that CVNN statistically out-perform RVNN in general and not for a particular case only.

Chapter 4 explains the theory of Synthetic Aperture Radar (SAR) to lay the ground for the following Chapter, which deals with these radar images as an input dataset for pixel-wise classification and segmentation tasks.

Chapter 5 details the experiments and results obtained on PolSAR classification and segmentation. We first propose a framework to assert the comparison between CVNN and RVNN is fair and prove that CVNN perform better for three

different model architectures and two PolSAR images. We later argue that the input representation used in most PolSAR classification tasks may not be optimal for all cases and that convolutional layers may perform better using another representation and sustain our argument by performing the corresponding simulations. Secondly, we show that the classification of PolSAR images is a saturated case and that this is mainly due to a correlation between training, validation and test set, and we propose a method to reduce this correlation and lower the saturation of *state-of-the-art* applications.

We close this thesis report with the conclusions in Chapter 6, and we discuss future work perspectives.

Publications

Journals

- Barrachina, J. A., Ren, C., Morisseau, C., Vieillard, G., and Ovarlez, J.-P. (2022d). Comparison Between Equivalent Architectures of Complex-Valued and Real-Valued Neural Networks - Application on Polarimetric SAR Image Segmentation. *Journal of Signal Processing Systems*, pages 1–10

Proceedings

- Barrachina, J. A., Ren, C., Vieillard, G., Morisseau, C., and Ovarlez, J.-P. (2022e). Real- and Complex-Valued Neural Networks for SAR image segmentation through different polarimetric representations. In *2022 IEEE International Conference on Image Processing (ICIP)*

- Barrachina, J. A., Ren, C., Morisseau, Vieillard, G., C., and Ovarlez, J.-P. (2022c). Merits of Complex-Valued Neural Networks for PolSAR image segmentation. In *GRETSI XXVIIIème Colloque Francophone de Traitement du Signal et des Images*

- **3MT finalist**. Barrachina, J. A., Ren, C., Morisseau, Vieillard, G., C., and Ovarlez, J.-P. (2022a). Complex-Valued Neural Networks for Polarimetric SAR segmentation using Pauli representation. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*

- **Ranked top 15% reviewer score**. Barrachina, J. A., Ren, C., Vieillard, G., Morisseau, C., and Ovarlez, J.-P. (2021c). About the Equivalence Between Complex-Valued and Real-Valued Fully Connected Neural Networks - Application to PolInSAR Images. In *2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6

- **r/MachineLearning (reddit) WAYR week 116**. Barrachina, J. A., Ren, C., Morisseau, C., Vieillard, G., and Ovarlez, J.-P. (2021a). Complex-Valued vs. Real-Valued Neural Networks for Classification Perspectives: An Example

on Non-Circular Data. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2990–2994

Without proceedings

- **Workshop presentation and publication**. Barrachina, J. A., Ren, C., Morisseau, Vieillard, G., C., and Ovarlez, J.-P. (2022b). Complex-Valued Neural Networks for Polarimetric SAR segmentation using Pauli representation. 5th SONDRA Workshop

- **Poster session**. Barrachina, J. A. (2020). A comparison between complex and real-valued fully connected neural networks on non-circular complex data. XXII Giambiagi Winter School: Artificial intelligence and deep learning in physics

- **Code**. Barrachina, J. A. (2021). NEGU93/cvnn. https://doi.org/10.5281/zenodo.4452131
  Barrachina, J. A. (2019). Complex-Valued Neural Networks (CVNN). https://github.com/NEGU93/cvnn

1 - Theory of Real-Valued Neural Networks

1.1 . Terminology

With the late success of Deep Learning, many sources have cited this term and others, such as Machine Learning (ML), Neural Networks, or Artificial Intelligence (AI). This has caused much confusion through incorrect uses of the terminologies, which are often used interchangeably. The French Ministry of National Education has even created an official report to define all these terms with the objective of reducing confusion around them [Commission d'enrichissement de la langue française, 2019].

The Turing test, originally called the imitation game by Alan Turing in 1950, is a test of a machine's ability to exhibit intelligent behavior equivalent to, or indistinguishable from, that of a human. In other words, it posed the question:

« Can computers think? »

- Alan Turin [Turing, 1950]

Regardless of the answer, all algorithms that can create the impression that a machine is thinking can be classified as **Artificial Intelligence**. Therefore, AI takes no regard for the algorithm itself; if you could hand-code every possible response to a conversation, it would be considered as AI. Indeed, the term is defined by the Royal Spanish Academy, the global reference for the Spanish language, as the scientific discipline that takes care of creating software that executes operations comparable to those made by the human mind, such as logical reasoning or the capacity of learning [Real Academia Española, 2022].

Neural Networks are all those algorithms of AI that are inspired by the human brain. Such is the case of a MultiLayer Perceptron (MLP), which will be explained in the following Section.

Machine Learning, is an algorithm that, instead of coding the machine with the instruction on how to perform a specific task, learns from the data and examples. In these algorithms, an initial state of the algorithm is initialized, which usually performs badly by randomly generating the result if it can output any result at all, and it is through training that it then "learns" to perform the task.

Finally, **Deep Learning** is both a ML and an Artificial Neural Network algorithm that receives its name because of the use of multiple neural network layers. To sum up, all the terms are closely related; most of them are just subsets of a border definition, being AI a general term that includes all others as shown in Figure 1.1.

1.2 . History of Neural Networks

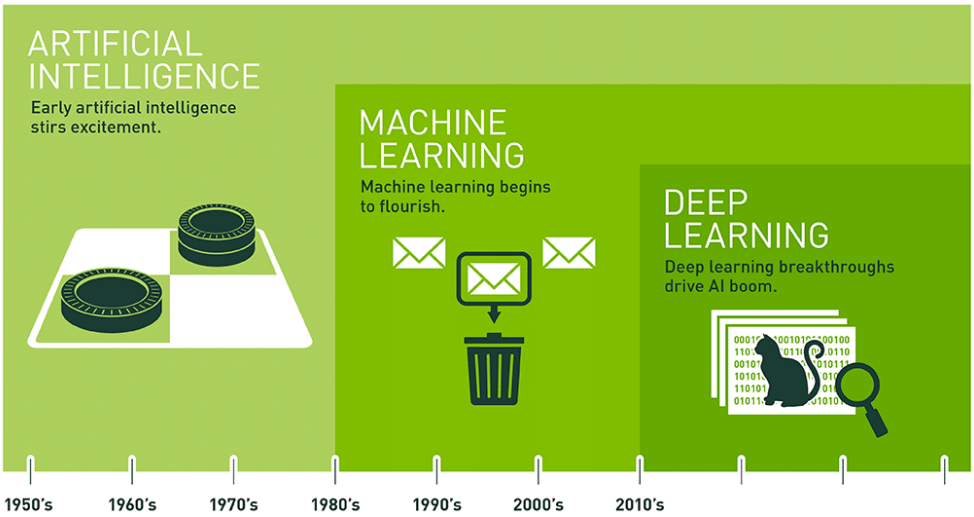


Figure 1.1: Terminology subsets extracted from [Copeland, 2016].

1.2.1 . Associationism

As stated before, AI is all about human brain behavior, and in particular, Neural Networks tries to achieve that by understanding and simulating how it works internally. Perhaps the early traces of this question can be referred to Aristotle's discussion about Associationism around 300 B.C. [Wang and Raj, 2017]. Associationism states that the mind is a set of conceptual elements organized as associations between them based on voluntary recollections grouped into four fundamental categories [Burnham, 1888].

- Similarity
- Contrast
- Frequency
- Contiguity

In Aristotle's vision, the feel, smell, and/or taste of an apple should naturally lead to the concept of an apple. But how can one store these relationships?

1.2.2 . Neural Groupings

The Scottish philosopher, inventor, and engineer Alexander Bain (1810-1877), Figure 1.2, introduced the notion that the information is in the connections [Bain, 1873]. In his model, he processes the associative memory to the distribution of activity of *neural groupings*. In contrast to other storage models, Bain's structure managed to describe a structure that could store multiple associations. As can be seen in Figure 1.3. Stimulation from 'a' and 'c' triggers 'X', from 'a' and 'b' triggers 'Y', and 'b' and 'c' trigger 'Z'.

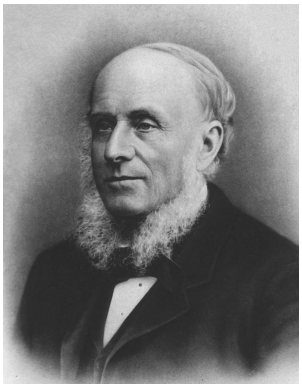


Figure 1.2: Alexander Bain.

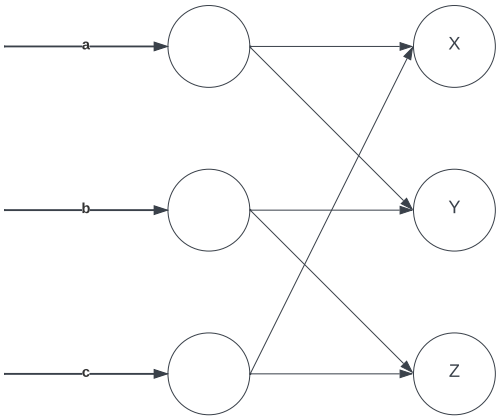


Figure 1.3: Illustration of Bain's Neural Groupings.

The similitude between Bain's neural grouping and Hebb's postulate (to be explained in the next Section) are remarkable, although nowadays, we usually label Hebb's postulate rather than Bain's [Wilkes and Wade, 1997]. This was probably due to the fact that Bain himself recalled all his ideas [Wang and Raj, 2017].

Indeed, ten years after his main publication on neural groupings, Bain published some notes and discussions on his theory where he presented two factors he failed to take properly into account [Bain, 1883]. At the end of his life, Bain withdrew his ideas not because he doubted logic but because he doubted arithmetic [Wilkes and Wade, 1997].

1.2.3 . MCP neuron

As stated in Section 1.1, Neural Networks algorithms are based on the human brain. Therefore they intend to emulate a neuron mathematically. Indeed, the first model of a human neuron was introduced by Warren Sturgis McCulloch (1898-1969) and Walter Harry Pitts (1923-1969), Figure 1.4, known as MCP for McCulloch-Pitts, in their work of 1943 [McCulloch and Pitts, 1943]. Although McCulloch was a neurophysiologist, Pitts was a self-taught logician who ran from

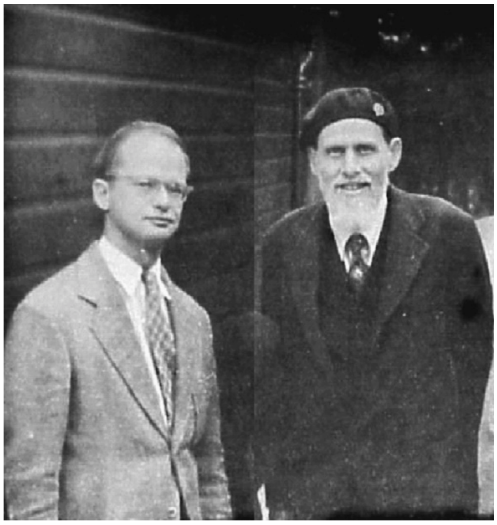


Figure 1.4: W. Pitts (left) and W. McCulloch (right) in 1949 [Moreno-Díaz and Moreno-Díaz, 2007].

his home at the age of 15 and was homeless until McCulloch invited him to live with his family. They both died in 1969.

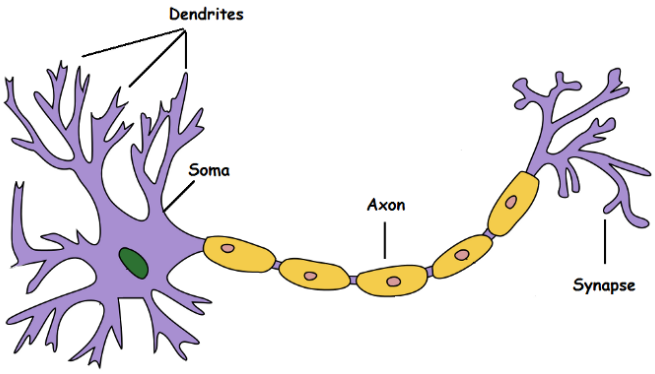


Figure 1.5: Illustration of a biological neuron.

Biological neurons are composed of a soma (body of the neuron), dendrites that receive signals from other neurons, an axon that connects neurons between each other and a synapse, which is the connection point to other neurons, as shown in Figure 1.5. The neuron receives an input signal through the dendrites, then processes the information in the soma and passes (or not) their output signal through the axon to other neurons.

McCulloch and Pitts's model of a neuron is exemplified in Figure 1.6. The main idea is that after receiving the information from the input, the neuron assigns values to each input, and after adding them and computing a threshold function

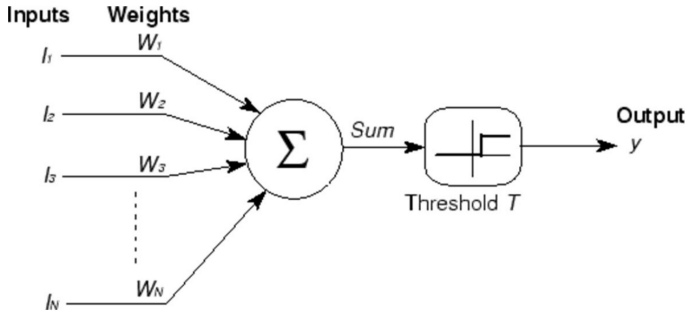


Figure 1.6: Mathematical model of a neuron according to [McCulloch and Pitts, 1943].

T, the neuron's output is obtained. This can be mathematically described as

$$O = T\left(\sum_{i}^{N} I_i\, w_i\right) , \tag{1.1}$$

where I_i are the inputs (generated from a previous neuron), w_i are the weights that correspond to I_i input. Finally, the T function outputs 1 if its input is higher than a certain constant value and 0 otherwise.

However, this model did not provide a learning mechanism for the weights, which was soon solved by Hebbian Learning Rule.

1.2.4 . Hebbian Learning Rule

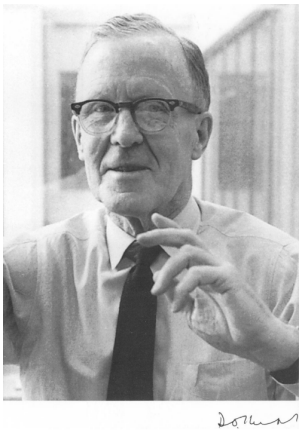


Figure 1.7: Donald Olding Hebb [Milner and Milner, 1996].

Hebbian Learning Rule is named after Donald Olding Hebb (1904-1985), Figure 1.7. In 1949 he introduced the famous rule

« Cells that fire together wire together »

- Donald Olding Hebb [Hebb, 1949].

Hebbian Learning Rule basically states that the connection between two units should be strengthened as the co-occurrences of these two units increase, presenting a rule to train the network connection. This can be mathematically written as:

$$\Delta w_i = \eta \, x_i \, y \, , \tag{1.2}$$

where Δw_i stands for the change of the synaptic weight (w_i) connecting neuron x_i (input) and y, and η is the learning rate.

The main drawback of the Hebbian Learning Rule is that as the frequency between units' co-occurrences rises, the weight increases exponentially and quickly becomes dominant. This problem is known as the instability of the Hebbian Learning Rule [Wang and Raj, 2017].

1.2.5 . Simple Perceptron

(a) Frank Rosenblatt (b) Perceptron machine

Figure 1.8: The perception of Frank Rosenblatt associating the photoelectric cells and cables to recognize alphabet letters.

In 1958, Frank Rosenblatt (1928-1971) merged Hebbian Learning Rule with the MCP neuron creating what he called the Perceptron [Rosenblatt, 1958], a machine that showed the ability to learn. However, his breakthrough was not only limited to the mathematical model but even the physical, electronic machine, as can be seen in Figure 1.8. Apart from the learning perspective, another difference between the Perceptron and the MCP neuron is the introduction of a non-linear function like the sigmoid function instead of the threshold function T detailed previously [Wang and Raj, 2017].

1.2.6 . The AI Winter

In 1969, Marvin Minsky (1927-2016) and Seymour Papert unintentionally unraveled the winter of Neural Networks after publishing their book *Perceptrons* [Minsky and Papert, 1969]. In this book, they highlighted the Perceptron limitation to solve XOR operations. These conclusions resulted in a significant reduction in work in the area until around 1980.

The main problem discussed here was the impossibility of a single Perceptron to perform the XOR operation. It is worth noticing, in the context of this work, that a complex-valued Perceptron does not present this limitation [Nitta, 2004].

In 1982, an initiative by Japan's Ministry of International Trade and Industry (MITI), titled the Fifth Generation Computer Project (FGCP), reignited the AI torch, but it did not last long as the 1984 annual meeting of the Association for the Advancement of Artificial Intelligence (AAAI), Roger Schank and Marvin Minsky talked about the coming AI Winter [Kaynak, 2021].

1.2.7 . The AI Wars

Neural Networks and other algorithms inspired by the actual biological neuron are widely known as connectionism. However, it exists another branch of AI known as Symbolic AI that is based on logic. Indeed, Symbolic AI was the dominant paradigm for a long period, mainly around 1960 and 1970, during the connectionism AI winter. The research community was very divided between these two fields of AI. In the mid-1970s, Roger Schank coined the distinction between *neat* AI and *scruffy* AI. Algorithms like Symbolic AI which used logic and formal mathematical optimization paradigms are known as the *neats*, and many well-known scientists such as John McCarty (1927-2011) from Stanford University work on that domain. On the other hand, *Scruffies* employ a variety of hand-coding or knowledge engineering. The work of Marvin Minsk at MIT, who published the book discussed in the previous Section (Ref. [Minsky and Papert, 1969]), was strongly associated with this term. Indeed, even after the 1990s, the discussion was still a topic of debate [Minsky, 1991].

1.2.8 . Golden era

It turned out that the solution to the main limitation discussed in Reference [Minsky and Papert, 1969] was simply to stack many layers of multiple perceptrons side by side to form what is known as MultiLayer Perceptron (MLP) [Kawaguchi, 2000]. This model has the remarkable property known as the universal approximation property, which roughly describes that a MLP can represent any function provided it is big enough.

Convolutional Neural Network (CNN) represent a huge breakthrough in image processing applications, since Yann LeCun presented LeNet [Le Cun et al., 1995] (5 convolutional layers) designed to recognize handwritten digits in images, the major breakthrough in the history of CNN architectures showing cutting-edge results for image classification, then AlexNet [Krizhevsky et al., 2017] winner of ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 [Kang et al., 2020], VGG-16 [Simonyan and Zisserman, 2014] (16 convolutional layers) and GoogleNet or Inception-V1 [Szegedy et al., 2015] (22 layers) winner of ILSVRC 2014 and ResNet-50 [He et al., 2016] (50 convolutional layers) winner of ILSVRC 2015 which could classify objects in images better than humans. The error rate of ILSVRC winner is displayed in Figure 1.9. In blue, winner methods were not based on CNN,

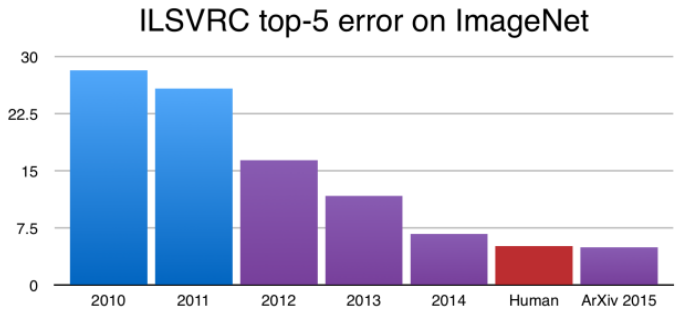


Figure 1.9: The top-5 error rate in the ImageNet Large Scale Visual Recognition Challenge has been rapidly reducing since the introduction of deep neural networks in 2012 [Zhang, 2015].

showing an important reduction in 2012; from then onwards, algorithms based on a CNN architecture won every year until in 2015, human error rate (in red) was beaten by AI. As it can be seen looking at state-of-the-art CNN architectures, the tendency is to make deeper and deeper neural networks, which explains the term *Deep Networks*.

In 2015, Reference [Long et al., 2015] developed a network without using any dense layer but only convolutional layers designed for semantic segmentation tasks. This architecture is called Fully Convolutional Neural Network (FCNN) and current *state-of-the-art* complex-valued architecture for PolSAR segmentation is based on this model and we will use it later in this work. This model is also similar to a possible more popular architecture known as U-Network (U-NET), also published the same year [Ronneberger et al., 2015].

By searching the term *Machine Learning* on Google Trends, a tool from Google that analyzes the popularity of top search queries in Google Search across various regions and languages, we can see a very steep increase in popularity mainly since 2015/2016 (Figure 1.10).

The results of a search in the Web of Knowledge (all databases) with the search string *Artificial Intelligence* also puts in evidence the great popularity of AI. This result is depicted in Figure 1.11 [Kaynak, 2021]. Note that this image was generated in 2021, so the total number of publications in 2021 is incomplete and expected to be higher. Such a surge in the number of publications translates into 5–10 times more submissions to the existing journals, which inevitably results in some bottlenecks on the journey from submission to publication, forcing researchers to seek new avenues for dissemination of their work.

1.3 . MultiLayer Perceptron

We will start our explanation by detailing a fully-connected neural network, also known as MultiLayer Perceptron (MLP). As the name indicates, the network

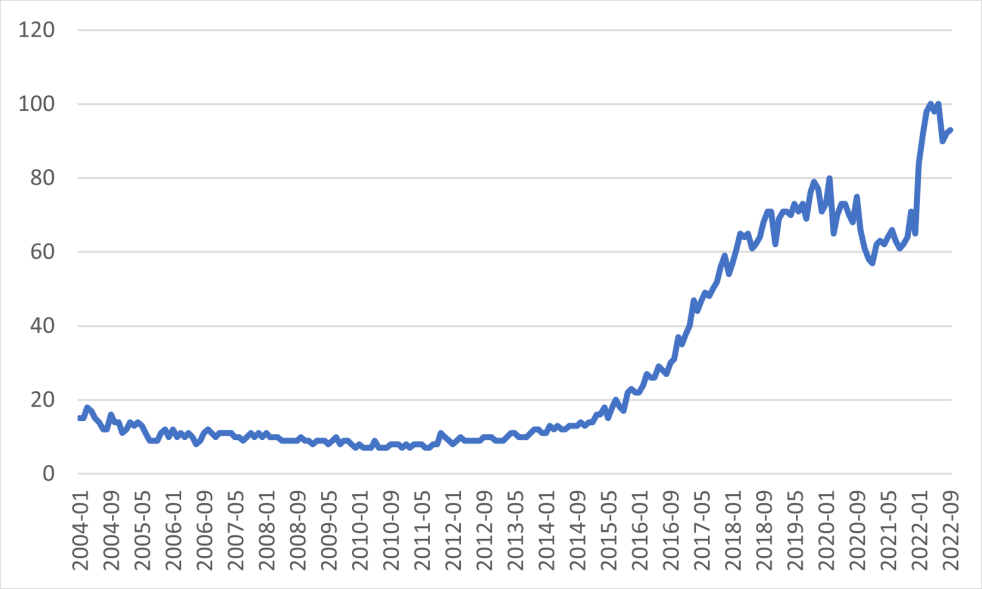


Figure 1.10: Machine Learning (Worldwide) search result on Google Trends.

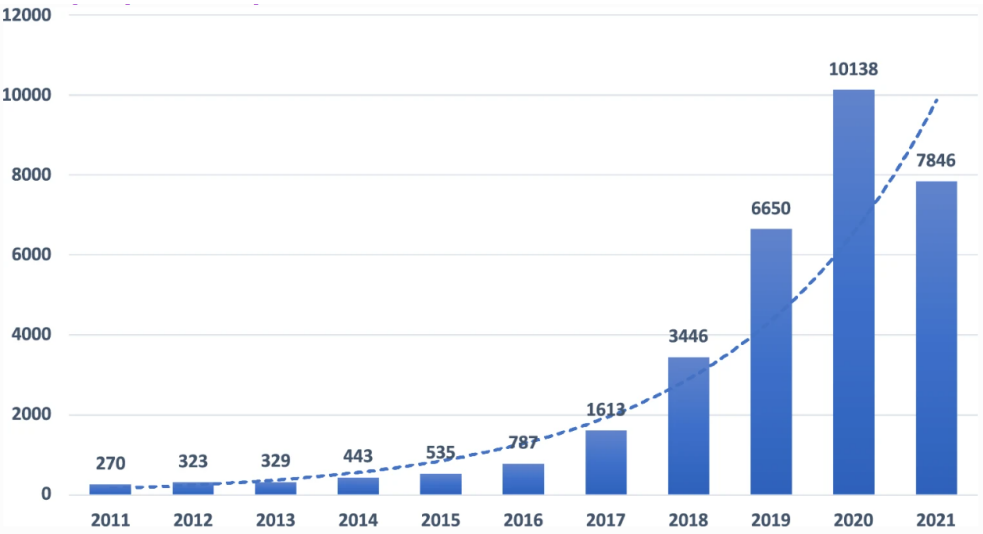


Figure 1.11: Number of AI publications per year according to Web Knowledge extracted from [Kaynak, 2021].

consists of layers composed of several perceptrons connected to all the perceptrons of the following layer.

1.3.1 . Notation

A MLP can be represented generically by Figure 1.12. For that given multi-layered neural network, we define the following variables, which will be used in the following subsections and throughout our work:

- $0 \leq l \leq L$ corresponds to the layer index where L is the output layer index and 0 is the input layer index.

- $1 \leq n \leq N_l$ the neuron index, where N_l denotes the number of neurons of layer l.

- $\omega_{nm}^{(l)}$ weight of the n^{th} neuron of layer $l-1$ with the m^{th} neuron of layer l.

- σ activation function.

- $X_n^{(l)} = \sigma\left(V_n^{(l)}\right)$ considered the output of layer l and input of layer $l+1$, in particular, $X_n^{(L)} = y_n$. With $V_n^{(l)}$ being

- $V_n^{(l)} = \sum_{m=1}^{N_{l-1}} \omega_{nm}^{(l)} X_m^{(l-1)}$

- $e_n(d_n, y_n)$ error function. d_n is the desired output for neuron n of the output layer.

- $\mathcal{L} = \sum_{n}^{N_L} e_n$ cost or loss function.

- $E = \frac{1}{P} \sum_{p}^{P} \mathcal{L}_p$ minimum error function, with P the total number of training cases or the size of the desired batch size.

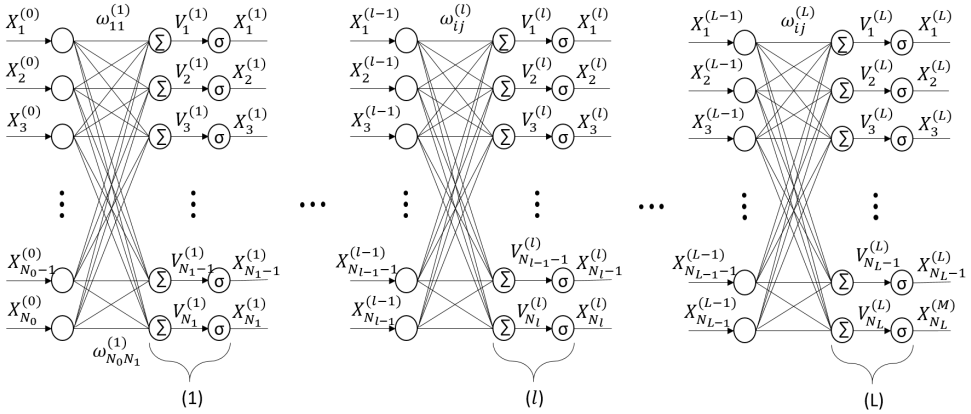


Figure 1.12: Feed-forward Neural Network Diagram.

1.4 . Feed-forward phase

Regardless if we are performing classification or regression tasks, we will have an input vector $X^{(0)}$ and the desired output d. Because the labels d are used for training, we say this is a *supervised learning* algorithm. Although there exist some variances to doing semi-supervised learning with this type of network, this is outside the scope of this thesis, and we will not explore it.

The output of neuron n of a layer l is computed as

$$X_n^{(l)} = \sigma\left(\sum_m^{N_{l-1}} \omega_{nm}^{(l)} X_m^{(l-1)} + b_n\right) . \tag{1.3}$$

The new term b_n, not discussed in Section 1.3.1, is known as the bias term and is also a trainable parameter that helps shift the activation function to the left or the right, which might be critical for successful training.

Knowing that $X_n^{(0)}$ is the input and $X_n^{(L)}$ is the output, we can simply randomly initialize $\left\{\omega_{nm}^{(l)}\right\}_{l,m,n}$, and we will be able to compute the output of the network given a certain input. This output, however, will be random and far from the expected behavior. We need, therefore, to train our network.

1.5 . Backpropagation

After computing the output of the network, we can compute the cost or $\mathcal{L}$ function as defined in Section 1.3.1 using the labels or ground truth d_n. By computing the derivative of the loss with respect to each trainable parameter to obtain the gradient, we would be able to minimize the function by knowing in which direction we should update the weights. We will, therefore, upgrade the

weights with the following optimization definition:

$$\Delta\omega_{nm}^{(l)} = -\eta \frac{\partial \mathcal{L}}{\partial \omega_{nm}^{(l)}} \,. \tag{1.4}$$

In order to successfully train a neural network, we will therefore need to be able to compute the gradient automatically using some algorithm. This is done using the algorithm known as automatic differentiation (autodiff), which is explained in detail in Appendix C. This algorithm is also used by *Tensorflow* to compute the gradients, and it already supports complex numbers for which there was no need to implement this algorithm in our toolbox. Furthermore, the mathematical development for the backpropagation algorithm is shown in Appendix B.6. Both Appendixes (B and C) explain the real case first and then develop the complex-valued algorithm using the real-valued case as a base.

1.6 . Modules

The model explained so far is a basic model to which many optimizations and tweaks have been made over the years to improve performance. We will explain the most relevant for our research here.

1.6.1 . Activation functions

The activation function is of vital importance for the generalization of the neural network. Indeed, if not used, adding hidden layers would have no effect, and the network will just perform as a linear classifier.

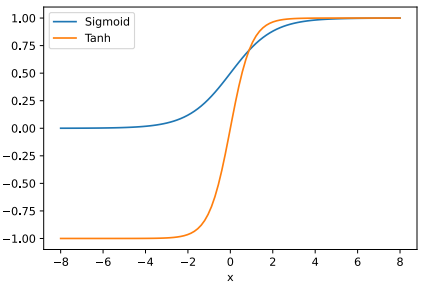
As discussed previously, in Section 1.2.3, the MCP neuron model proposes a threshold function T that, when its threshold value is 0, is equivalent to the binary step (sign) function. However, researchers migrated from its beginning to the logistic sigmoid function, a special case of the logistic equation defined by the Belgian mathematician Pierre-François Verhulst (1804-1849) in 1838 [Bacaër, 2011]. The logistic sigmoid function is defined as

$$\text{Sigmoid}(x) = \frac{1}{1+e^{-x}} \,. \tag{1.5}$$

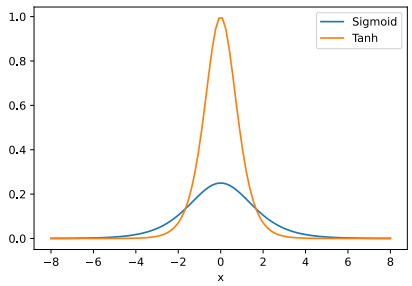
Another popular option for feed-forward networks is the hyperbolic tangent (tanh), whose equation is

$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \,. \tag{1.6}$$

Both of these functions are s-shapes for their form, as can be seen in Figure 1.13a. tanh function has the virtue of being zero centered, $\tanh \in [-1, 1]$ and thus having zero mean. Both these functions, however, have a problem known as the *vanishing gradient*. This issue means that for high or low values of x, the gradient will be null, as we can see from Figure 1.13b, and the network will stop learning.



(a) S-shape activation functions



(b) S-shape activation derivates

Figure 1.13: Logistic Sigmoid and tanh activation functions and its derivatives.

In 2010, Vinod Nair and Geoffrey Everest Hinton (1947-) proposed to use the Rectified Linear Unit (ReLU) activation function [Nair and Hinton, 2010], which is currently the most widely used activation function since that date [Nwankpa et al., 2020, Ramachandran et al., 2018]. ReLU is a piece-linear activation function as it can be computed as

$$\text{ReLU}(x) = \max(0, x) = \begin{cases} x & \text{if } 0 < x \\ 0 & \text{if otherwise}\,. \end{cases} \tag{1.7}$$

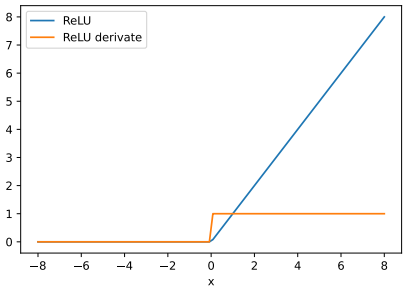


Figure 1.14: ReLU activation function.

From Equation 1.7, we can deduce that the calculation of this function is fast to compute as it does not involve any exponential or division as sigmoid or tanh need. The derivative is also very fast as it can be seen as the sign function with an indeterminate value at zero (see Figure 1.14). Indeed, although being not bounded and not derivable at 0, it helps mitigate the *vanishing gradient* problem. In general, ReLU seems like the best option for deep networks as they are easier to optimize, converge faster, generalize better, and faster to compute [Zeiler et al., 2013, Krizhevsky et al., 2012].

Due to ReLU popularity, many variations have been developed, such as ELU [Clevert et al., 2015], LeakyReLU [Maas et al., 2013], SeLU [Klambauer et al., 2017], Parametric ReLU [He et al., 2015], etc.

A very useful activation function when dealing with classification problems is the *softmax* function [Bridle, 1989], which is defined as

$$softmax(\boldsymbol{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{N_L} e^{x_j}}\,, \tag{1.8}$$

where the input is a vector of size, N_L, and i is the i-th element of the *softmax* output vector. This function ensures all its outputs are positive and that their sum adds to 1, constituting a valid probability distribution. If we are to use a neural network for classification problems, we could design a network whose output size N_L equals the total amount of classes and create our labels to match these outputs; for example, in a case of four classes, an input vector pertaining to class number two (counting from zero) would be represented by the vector $(0, 0, 1, 0)^T$. This kind of representation is known as *one-hot encoding*. Under this context, we could use *softmax* as the output activation function, and it will suffice to choose the higher value as the prediction.

For all our use cases, we will be using *softmax* as the output function and any of the other functions as the hidden layers activation function, with a preference towards the ReLU function.

1.6.2 . Loss

For multi-class classification tasks, Categorical cross-entropy is probably the most well-known loss function of all and the only one we will be discussing here. Categorical cross-entropy is defined as

$$e_n(d_n, y_n) = \sum_i d_{n,i} \log(y_{n,i})\,, \tag{1.9}$$

with $d_{n,i}$ being the i^{th} component of the n^{th} sample (i.e., either 0 or 1 depending on the label).

Notice that this definition is for multi-class tasks using *one-hot encoding* representation. We could simplify the definition for binary classification tasks by using:

$$e_n(d_n, y_n) = -d_n \log(y_n) - (1 - d_n)\, \log(1 - y_n)\,. \tag{1.10}$$

Among the two terms, only one will be used depending on the value of d_n. In both cases, a correct prediction will yield a 1 inside the log function, which will equal a loss value of zero, as seen in Figure 1.15. As the interior of the $\log$ goes farther from 1, the error increases faster. This behavior translates into a higher penalization where predictions had higher confidence than errors where the model was not certain of the decision.

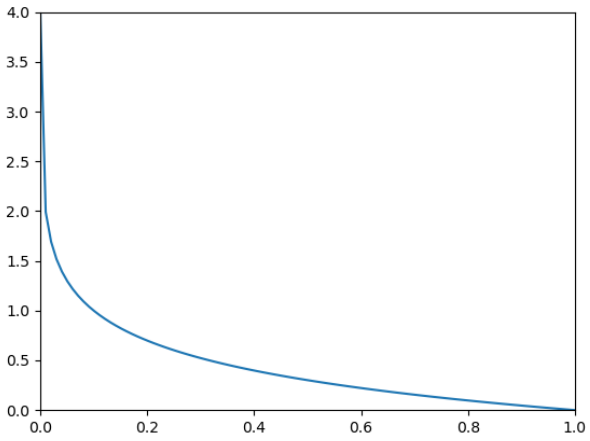


Figure 1.15: $-\log_{10}$ function

For our semantic segmentation experiments, we sometimes used patch images where not all pixels had a label. This was represented as a vector with zeros like $\boldsymbol{d} = (0, 0, 0, 0)^T$. Although not specifically mentioned in the following Chapters, these cases were contemplated in the code, and the loss function ignored those pixels.

1.6.3 . Optimizer

The optimizer listed in Equation 1.4 is known as *gradient descent*. In this case, the loss function is computed using the average of all the training examples to approach the optimal minima. However, we normally have a huge number of training examples which can make the training extremely slow to compute, as all costs must be computed before each parameter upgrade. Instead, what is done is to split the input dataset into smaller batches of a given size and trains iteratively per batch. If you have, for example, one hundred batches, the trainable parameters will be updated one hundred times with only seeing each training example once. Once all the batches have been used for training, we say we did one epoch. Training a neural network is composed of hundreds or even thousands of epochs. This methodology is called Stochastic Gradient Descent (SGD).

Extensive work has been done in the field of optimizers. Adding momentum to the SGD algorithm can simulate some sort of inertia that increase stability and avoid certain local minima. AdaGrad [Duchi et al., 2011] uses a different learning rate for each neuron at each layer. There is RMS-Prop (Root Mean Square Propagation) [Tieleman and Hinton, 2012] which can be seen as the combination of both momentum and AdaGrad. Other optimizations might be mentioned, such as Adadelta [Zeiler, 2012] or Ftrl [McMahan et al., 2013] but probably the most significant of all (and the one we will choose to work within our experiments) is

Adam (Adaptive Moment Estimation) [Kingma and Ba, 2014]. Adam is easy to implement, computationally efficient, and requires little memory [Kingma and Ba, 2014]. Both Adam and RMS-Prop clearly outperform SGD [Choi et al., 2019,Desai, 2020]. Other modifications were done to Adam optimizer such as Nadam [Dozat, 2016] or AmsGrad [Reddi et al., 2018].

1.6.4 . Initialization

As explained in Section 1.4, one must randomly initialize the trainable parameters. However, which law to use for this initialization was not explained. One of the most popular initialization algorithms is Glorot, also known as Xavier initialization technique [Glorot and Bengio, 2010]. This method proposes to maintain a constant variance when doing the feed-forward and backpropagation phases. As it is not possible to maintain both cases at the same time, a compromise is chosen.

Xavier initialization technique was designed for sigmoid activation functions, which were popular at that time. To develop their methodology, Glorot et al. assumed to be working on the linear part of the activation function. This is a bold assumption when using ReLU as an activation function and, therefore He (or Kaiming) initialization technique was born [He et al., 2015], which avoids the use of this assumption.

1.6.5 . Convolutional Neural Networks

So far, we have used a fully-connected layer for our models. However, several different layers can be used. In this section, we will remit ourselves to only one different type of layer, which is very popular whenever we are dealing with images, the convolutional layer [Le Cun et al., 1999]. These layers give the name to the famous Convolutional Neural Network (CNN)s, which are used in most state-of-the-art benchmarks [Patel and Patel, 2020].

Convolutional layers consist of a set of filters or kernels. Each kernel is slid through the input image, multiplying its values with those of the input element-wise and then adding them later to form the output or feature map. This procedure is exemplified in Figure 1.16.

Normally, zeros are added to the input image (known as zero-padding) to prevent the feature map from being of a smaller size. Other parameters are important such as stride, defined as the step size of the kernel when traversing the image or dilatation rate [Yu and Koltun, 2015], which defines a spacing between the values in a kernel. In particular, a stride of 2 is used on the Pooling layers to reduce the size of the image. Reference [Dumoulin and Visin, 2016] talks about all these variables and how they relate to the convolutional layer output shape.

A popular layer used when implementing CNN is the pooling layer which reduces the size of the giving input. The two main pooling layers are the average pooling (reduce the size by performing an average of adjacent pixels) and the max pooling (keeps the maximum value of a given boxcar section exemplified in Figure 1.17).

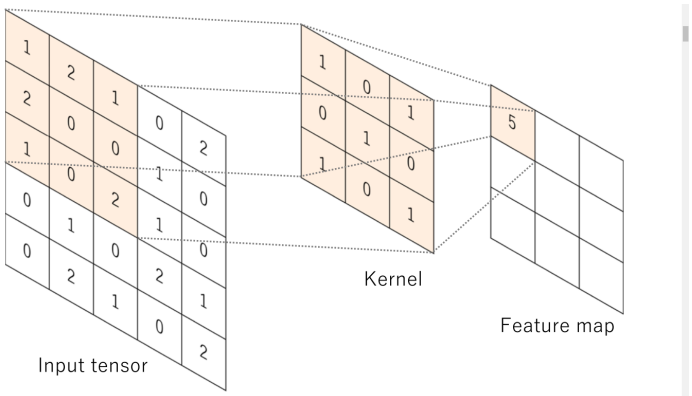


Figure 1.16: Convolutional layer example extracted from [Yamashita et al., 2018].

1.6.6 . Overfitting and Regularization

If we are to fit the dots on Figure 1.18 with a polynomial equation using only the blue dots as training, we could have different curves depending on the order of the polynomial equation we are to implement. Indeed, the lower right plot will obtain 100% accuracy on the training data, but we can easily see how the upper figure generalizes better for the desired task. This phenomenon is known as overfitting. If we train long enough on certain data, it is possible that although the accuracy rises, the network generalizes poorly for unseen data. A simple technique to avoid this is called *early stopping* [Morgan and Bourlard, 1990] which consists of using a validation set that computes the loss after each epoch, and the moment the loss starts to rise compared to previous epochs, we stop the training.

Another simple and powerful technique to reduce overfitting is Dropout [Srivastava et al., 2014], which consists of randomly eliminating some neurons at certain epochs.

When training Deep Neural Networks, the distribution of each layer inputs changes during training because the parameters of the previous layer change, slowing down, training [Ioffe and Szegedy, 2015]. It is possible to speed up this process by using Batch Normalization (BN) which normalizes layer inputs. Batch Normalization also reduces overfitting as well as provides similar regularization benefits as Dropout [Ioffe and Szegedy, 2015].

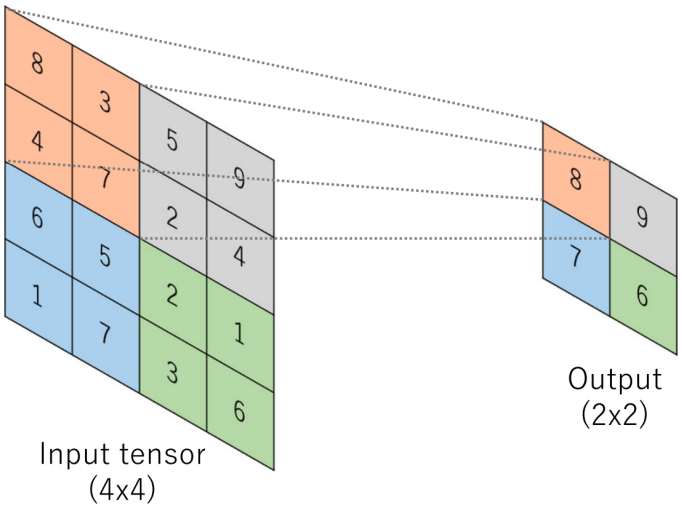


Figure 1.17: Max pooling example extracted from [Yamashita et al., 2018].

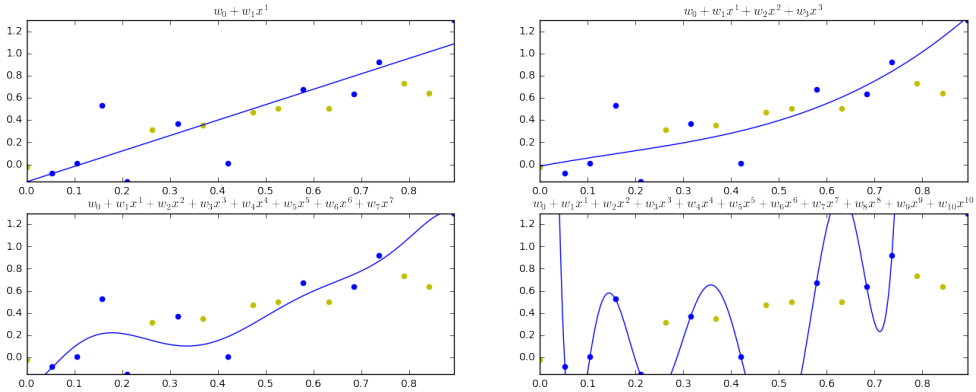
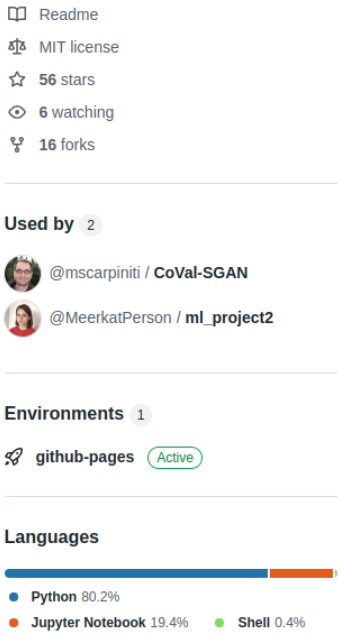


Figure 1.18: Overfitting example extracted from [Selmo et al., 2018].

2 - Implementation of the Complex-Valued Neural Network Modules

Although CVNN has been investigated for particular structures of complex-valued data [Hirose and Yoshida, 2012, Hänsch and Hellwich, 2009a, Hirose, 2012, Hirose, 2009], the difficulties in implementing CVNN models in practice have slowed down the field from growing further [Mönning and Manandhar, 2018]. Indeed, the two most popular Python libraries for developing deep neural networks, which are Pytorch, from Meta (formerly named Facebook) and Tensorflow from Google do not fully support the creation of complex-valued models.



Even though *Tensorflow* does not fully support the implementation of a CVNN, it has one significant benefit: It enables the use of complex-valued data types for the automatic differentiation (autodiff) algorithm [Hoffmann, 2016] to calculate the complex gradients as defined in Appendix C. Since July 2020, *PyTorch* also added this functionality as BETA with the version 1.6 release. later on, on June 2022, after the release of version 1.12, *PyTorch* extended its complex functionality by adding complex convolutions (also as BETA). Although this indicates a clear intention to develop towards CVNN support, there is still a lot of development to be done.

Libraries to develop CVNNs do exist, the most important one of them being probably the code published in [Trabelsi et al., 2017]. However, we have decided not to use this library since the latter uses *Theano* as a back-end, which is no longer maintained. Another code was published on GitHub that uses *Tensorflow* and *Keras* to implement CVNN [Dramsch, 2019]. However, as *Keras* does not support

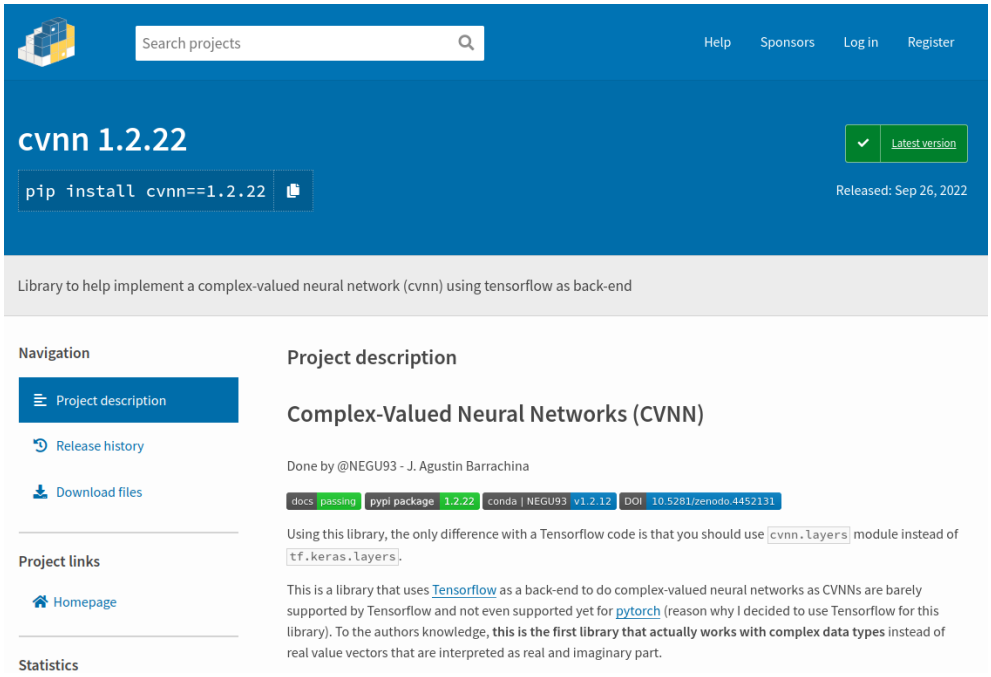


Figure 2.1: PIP cvnn presentation page.

complex-valued numbers, the published code simulates complex operations using real-valued datatypes. Therefore, the user has to transform its complex-valued data into a real-valued equivalent before using this library. The same happened with ComplexPyTorch [Matthès et al., 2021] until it was updated in January 2021 to support complex tensors.

During this thesis, we created a *Python* tool to deal with the implementation of CVNN models using *Tensorflow* as back-end. Note that the development of this library started in 2019, whereas *PyTorch* support for complex numbers started in mid-2020, which is the reason why the decision to use *Tensorflow* instead of *Pytorch* was made. To the author's knowledge, this was the first library that natively supported complex-number data types. The library is called CVNN and was published [Barrachina, 2021] using CERN's Zenodo platform which received already 18 downloads. It can be installed using both Python Index Package (PIP) (Figure 2.1) and Anaconda. The latter already has 193 downloads as of the 8^{th} October 2022, as shown in Figure 2.2, from which none of those downloads were ourselves.

The library was open-sourced for the community on GitHub [Barrachina, 2019] and has received a very positive reception from the community. As can be seen from Figure 2.3, the GitHub repository received an average of 2 clones and almost 50 visits per day in the last two weeks. With 62 stars at the beginning of October 2022. It also has a total of 16 forks and one pull request for a new feature that has already been reviewed and accepted. Six users are actively watching every update

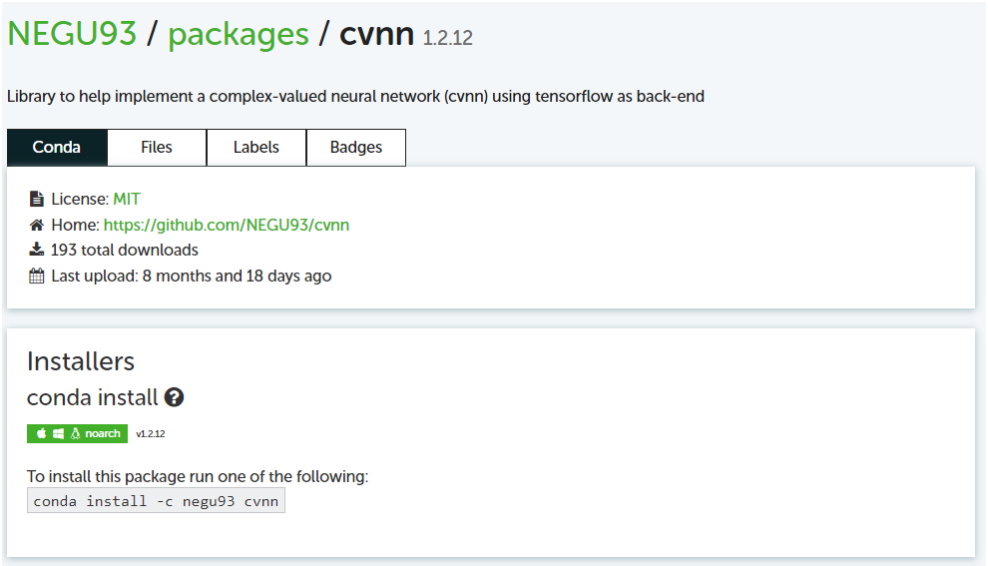


Figure 2.2: Anaconda cvnn presentation page.

Figure 2.3: GitHub cvnn toolbox traffic.

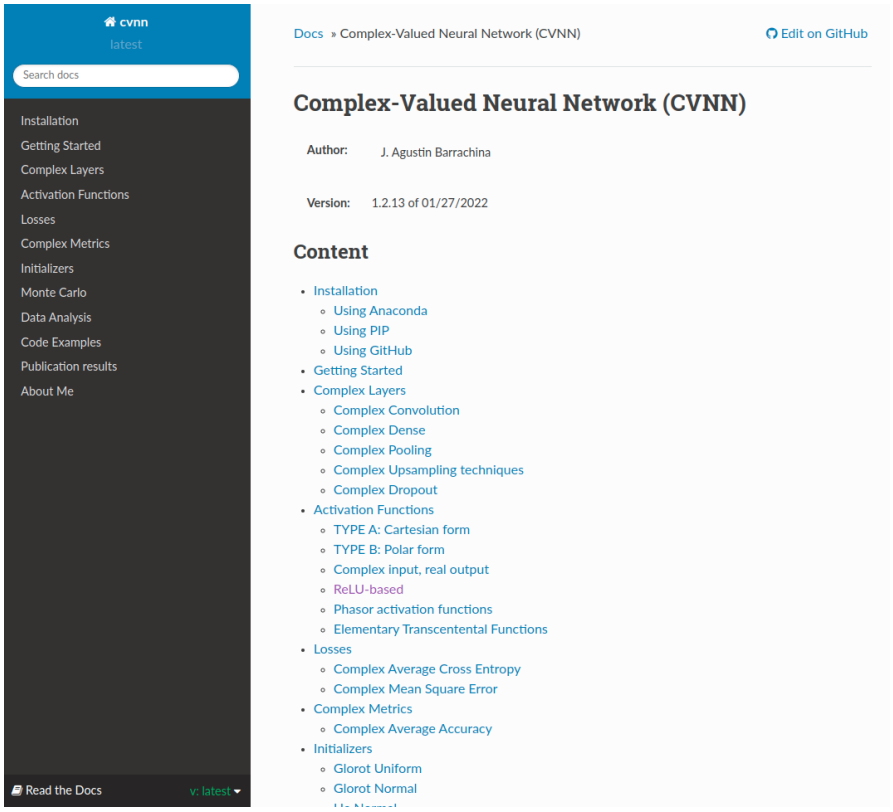
on the code as they have activated the notifications. Finally, two users have codes in GitHub importing the library. Thirty issues have been reported, and it was also subject to 31 private emails. All these metrics are evidence of the impact and interest of the community in the published code.

The library was documented using *reStructuredText* and uploaded for worldwide availability. The link for the full documentation (displayed in Figure 2.4a) can be found in the following link: complex-valued-neural-networks.rtfd.io. The documentation received a daily view which varied from a minimum of 18 views on one day to a maximum of 173 views as shown in Figure 2.4b

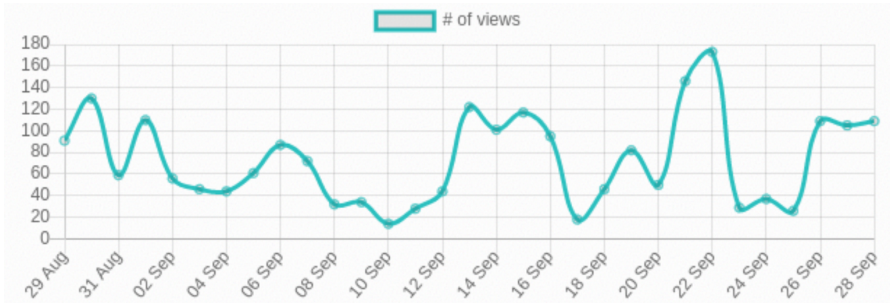
The *Python* testing framework Pytest was used to maintain a maximum degree of quality and keep it as bug-free as possible. Before each new feature implementation, a test module was created to assert the expected behavior of the feature and reduce bugs to a minimum. This methodology also guaranteed feature compatibility as all designed test modules must pass in order to deploy the code. The library allows the implementation of a Real-Valued Neural Network (RVNN) as well with the intention of changing as little as possible the code when using complex data types. This made it possible to perform a straight comparison against *Tensorflow*'s models, which helped in the debugging. Indeed, some *Pytest* modules achieved the same result that *Tensorflow* when initializing the models with the same seed.

Special effort was made on the User eXperience (UX) by keeping the Application Programming Interface (API) as similar as possible to that of *Tensorflow*. The following code extract would work both for a *Tensorflow* or cvnn application code:

```
import numpy as np
import tensorflow as tf

# Gets the dataset, when using cvnn you normally want this to be complex
# for example numpy arrays of dtype np.complex64
# to be done by each user
(train_images, train_labels), (test_images, test_labels) = get_dataset()

# This function returns a tf.Model object
model = get_model()

# Compile as any TensorFlow model
model.compile(optimizer='adam', metrics=['accuracy'], loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True))
model.summary()

# Train and evaluate
history = model.fit(train_images, train_labels, epochs=epochs, validation_data=(test_images, test_labels))
test_loss, test_acc = model.evaluate(test_images,
```



(a) *Read the Docs* documentation presentation page.



(b) cvnn documentation daily total views in *Read The Docs*.

Figure 2.4: Documentation hosted by *Read the Docs*.

```
    test_labels, verbose=2)
```

For creating the model, two APIs are available, the first one known as the sequential API whose usage is something like the following code extract:

```
import cvnn.layers as layers

def get_model():
    model = tf.keras.models.Sequential()
    model.add(layers.ComplexInput(input_shape=(32, 32, 3))
    )
    model.add(layers.ComplexConv2D(32, (3, 3), activation=
    'cart_relu'))
    model.add(layers.ComplexAvgPooling2D((2, 2)))
    model.add(layers.ComplexConv2D(64, (3, 3), activation=
    'cart_relu'))
    model.add(layers.ComplexMaxPooling2D((2, 2)))
    model.add(layers.ComplexConv2D(64, (3, 3), activation=
    'cart_relu'))
    model.add(layers.ComplexFlatten())
    model.add(layers.ComplexDense(64, activation='
    cart_relu'))
    model.add(layers.ComplexDense(10, activation='
    convert_to_real_with_abs'))
    # An activation that casts to real must be used at the
    last layer.
    # The loss function cannot minimize a complex number
    return model
```

However, some models are simply impossible to create with the sequential API like a U-NET architecture. For that, the functional API must be used like this:

```
import cvnn.layers as layers

def get_model():
    inputs = layers.complex_input(shape=(128, 128, 3))
    c0 = layers.ComplexConv2D(32, activation='cart_relu',
    kernel_size=3)(inputs)
    c1 = layers.ComplexConv2D(32, activation='cart_relu',
    kernel_size=3)(c0)
    c2 = layers.ComplexMaxPooling2D(pool_size=(2, 2),
    strides=(2, 2), padding='valid')(c1)
    t01 = layers.ComplexConv2DTranspose(5, kernel_size=2,
    strides=(2, 2), activation='cart_relu')(c2)
    concat01 = tf.keras.layers.concatenate([t01, c1], axis
    =-1)

    c3 = layers.ComplexConv2D(4, activation='cart_relu',
    kernel_size=3)(concat01)
    out = layers.ComplexConv2D(4, activation='cart_relu',
    kernel_size=3)(c3)
    return tf.keras.Model(inputs, out)
```

When using Keras loss functions, the output of the model must be real-valued as Tensorflow will not work with complex values. There are activation functions defined that receive complex values as input but output real value results to deal with this problem. Another solution is to use the cvnn library-defined loss functions instead.

Tensorflow blocks the use of a complex-valued loss as the optimizer input but allows the update over complex-valued trainable parameters by means of the *Wirtinger calculus* (Appendix B.5). As the loss is real-valued, the optimizer can be the same as the one used for real-valued networks, so no implementation was needed in this regard. However, the other modules must be implemented. Their detail will be described in the following Sections.

2.1 . Complex-Valued layers

CVNN, as opposed to conventional RVNN, possesses complex-valued input, which allows working with imaginary data without any pre-processing needed to cast its values to the real-valued domain. Each layer of the complex network operates analogously to a real-valued layer with the difference that its operations are on the complex domain (addition, multiplication, convolution, etc.) with trainable parameters being complex-valued (weights, bias, kernels, etc.). Activation functions are also defined on the complex domain so that $f : \mathbb{C} \to \mathbb{C}$ and will be described on Section 2.3.

A wide variety of complex layers is supported by the library, and the full list can be found in `complex-valued-neural-networks.rtfd.io/en/latest/layers.html`.

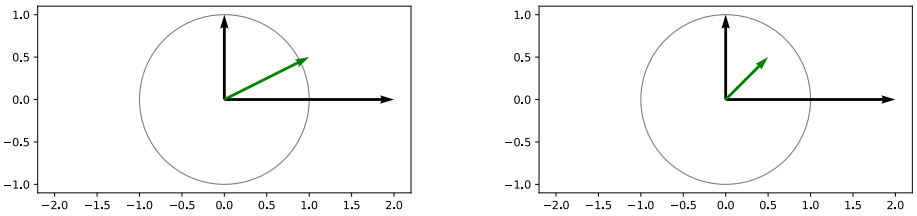
Some layers, such as dense layers, can be extended naturally as addition and multiplication are defined in the complex domain. Therefore, just by making the neurons complex-valued, their behavior is evident. The same analogy can be made for convolutional layers as the transformation from the complex to the real plane does not change the resolution of the image to justify increasing the kernel size or changing the stride.

Special care must be taken when implementing `ComplexDropout` as applying it to both real and imaginary parts separately will result in unexpected behavior as the ignoring weights will not be coincident, e.g., one might mask the real part while using the imaginary part for the computation. This, however, was taken into account for the layer implementation. The usage of this layer is analogous to *Tensorflow* Dropout layer, also known as *inverted dropout*, which also uses the boolean `training` parameter indicating whether the layer should behave in training mode (adding dropout) or in inference mode (doing nothing).

Other layers, such as `ComplexFlatten` or `ComplexInput`, needed to be implemented as *Tensorflow* equivalent `Flatten` and `Input` cast the output to float.

2.1.1 . Complex Pooling Layers

Pooling layers are not so straightforward. In the complex domain, their values are not ordered as the real values, meaning there is no sense of a maximum value, rendering it impossible to implement a Max Pooling layer directly on the input. Reference [Zhang et al., 2017] proposes to use the norm of the complex figure to make this comparison, and this method is used for the toolbox implementation of the `ComplexMaxPooling` layer. Average Pooling opens the possibility to other interpretations as well. Even if, for computing the average, we could add the complex numbers and divide by the total number of terms as one would do with real numbers, another option arises known as circular mean. The circular or angular mean is designed for angles and similar cyclic quantities.



(a) Arithmetic mean (b) Circular mean

Figure 2.5: Mean example for two complex values.

When computing the average of $2 + 0\,i$ and $0 + 1\,i$, the conventional complex mean will yield $1 + 0.5\,i$ for what the angle will be $\pi/6$ although the vectors had angles of $\pi/2$ and 0 (Figure 2.5a). The circular mean consists of normalizing the values before computing the mean, which yields $0.5 + 0.5\,i$, having an angle of $\pi/4$ (Figure 2.5b). The circular mean will have a norm inside the unit circle. It will be at the unit circle if all angles are equal, and it will be null if the angles are equally distributed. Another option for computing the mean is to use the circular mean definition for the angle and then compute the arithmetic mean of the norm separately, as represented in Figure 2.6.

All these options were used for computing the average pooling so the user could choose the case that fits their data best.

2.1.2 . Complex Upsampling layers

Upsampling techniques, which enable the enlargement of 2D images, were implemented. In particular, 3 techniques were applied, all of them documented in complex-valued-neural-networks.readthedocs.io/en/latest/layers/complex_upsampling.html.

- **Complex Upsampling** Upsampling layer for 2D inputs. The upsampling can be done using the nearest neighbor or bilinear interpolation. There are

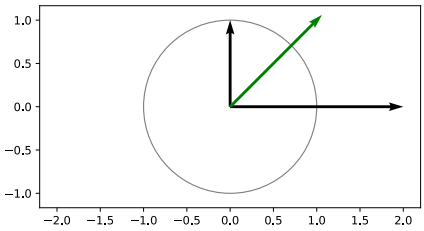


Figure 2.6: Circular mean with norm average.

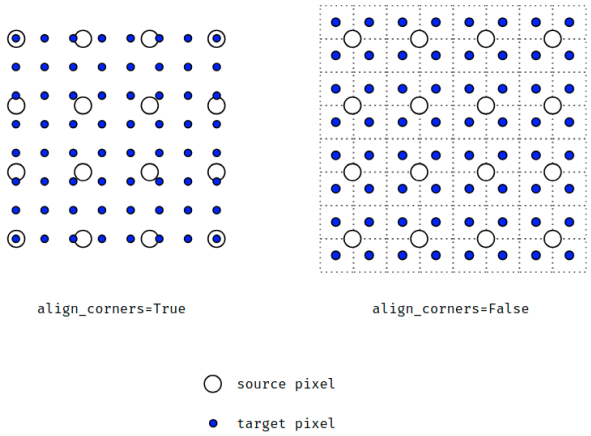


Figure 2.7: Upsampling alignments options extracted from [bkkm16, 2019].

at least two possible ways to implement the upsampling method depending if the corners are aligned or not (see Figure 2.7). Our implementation does not align corners.

- **Complex Transposed Convolution** Sometimes called **Deconvolution** although it does not compute the inverse of a convolution [Zeiler et al., 2010].
- **Complex Un-Pooling** Inspired on the functioning of Max Un-pooling explained in Reference [Zafar et al., 2018]. Max un-pooling technique receives the maxed locations of a previous Max Pooling layer and then expands an image by placing the input values on those locations and filling the rest with zeros as shown in Figure 2.8.

Complex un-pooling locations are not forced to be the output of a max pooling layer. However, in order to use it, we implemented as well a layer class named `ComplexMaxPooling2DWithArgmax` which returns a tuple of tensors, the max pooling output and the maxed locations to be used as input of the un-pooling layer.

There are two main ways to use the unpooling layer, either by using the expected output shape or using the `upsampling_factor` parameter.

```
from cvnn.layers import ComplexUnPooling2D, complex_input,
    ComplexMaxPooling2DWithArgmax
import tensorflow as tf
import numpy as np

x = get_img()       # Gets an image just for the example
# Removes the batch size shape
inputs = complex_input(shape=x.shape[1:])
# Apply max-pooling and also get argmax
max_pool_o, max_arg = ComplexMaxPooling2DWithArgmax(
    strides=1, data_format="channels_last", name="argmax")(
    inputs)
# Applies the Unpooling
outputs = ComplexUnPooling2D(x.shape[1:])([max_pool_o,
    max_arg])

model = tf.keras.Model(inputs=inputs, outputs=outputs,
    name="pooling_model")
model.summary()
model(x)
```

It is possible to work with variable size images using a partially defined tensor, for example, `shape=(None, None, 3)`. In this case, the second option (using `upsampling_factor`) is the only way to deal with them in the following manner.

```
# Input is an unknown size RGB image
inputs = complex_input(shape=(None, None, 3))
max_pool_o, pool_argmax = ComplexMaxPooling2DWithArgmax(
    strides=1, data_format="channels_last", name="argmax")(
    inputs)
```

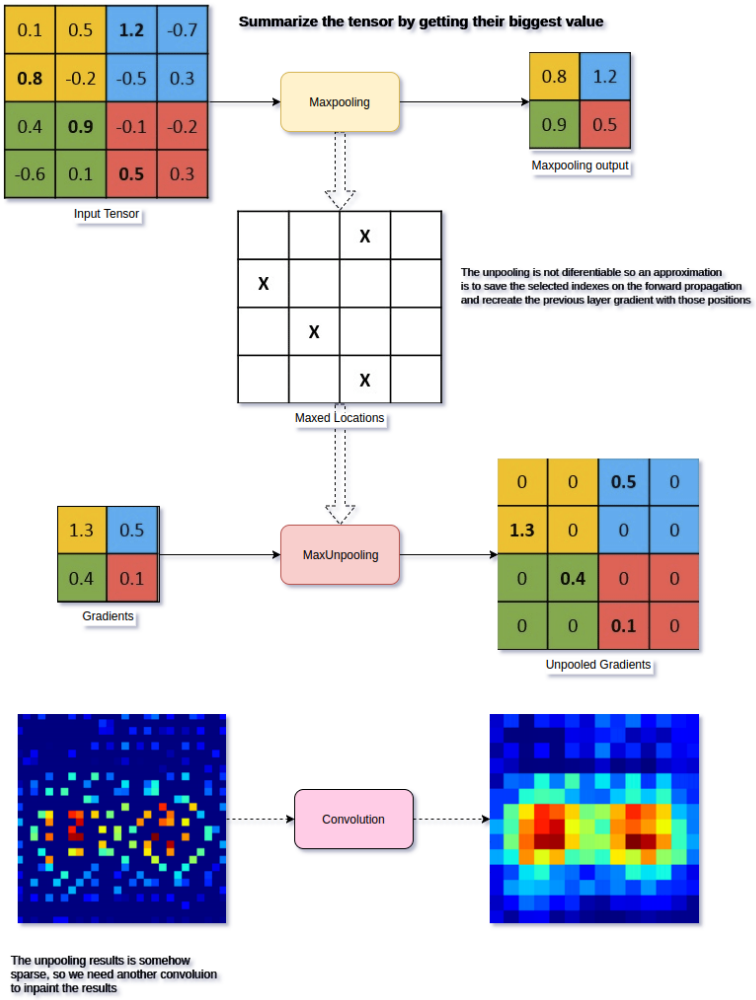


Figure 2.8: Max Unpooling graphical explanation extracted from [Zafar et al., 2018].

```
unpool = ComplexUnPooling2D(upsampling_factor=2)([
    max_pool_o, pool_argmax])

model = tf.keras.Model(inputs=inputs, outputs=outputs,
    name="pooling_model")
model.summary()
model(x)
```

All the discussed layers in this Section have a `dtype` parameter which defaults to `tf.complex64`, however, if `tf.float32` or `tf.float64` is used, the layer behaviour should be arithmetically equivalent to the corresponding *Tensorflow* layer, allowing for fast test and comparison. In some cases, for example, `ComplexFlatten`, this parameter has no effect as the layer can already deal with both complex- and real-valued input. Also, a method `get_real_equivalent` is implemented which returns a new layer object with a real-valued `dtype` and allows a `output_multiplier` parameter in order to re-dimension the real network if needed. This is used to obtain an equivalent real-valued network as described in Section 5.1.

2.2 . Complex-Valued Backpropagation

As mentioned earlier in this Chapter, the loss function remains real-valued to minimize an empirical risk during the learning process. Despite the architectural change for handling complex-valued inputs, the main challenge of CVNN is the way to train such neural networks.

A problem arises when implementing the learning algorithm (commonly known as backpropagation). The parameters of the network must be optimized using the gradient or any partial-derivative-based algorithm. However, standard complex derivatives only exist for the so-called *holomorphic* or *analytic* functions.

Because of Liouville's theorem (discussed in Appendix B.4), CVNNs are bound to use *non-holomorphic* functions and therefore can not be derived using standard complex derivative definition. CVNNs bring in *non-holomorphic* functions in at least two ways [Hirose et al., 2013]:

- with the loss function being minimize over complex parameters
- with the *non-holomorphic* complex-valued activation functions

Liouville's theorem implications were considered to be a big problem around 1990 as some researchers believed it led to the impossibility of obtaining and/or analyzing the dynamics of the CVNNs [Hirose, 2013].

However, Wirtinger calculus (discussed in Appendix B.5) generalizes the notion of complex derivative, making the *holomorphic* function a special case only, allowing researchers to successfully implement CVNNs. Under Wirtinget calculus, the gradient is defined as [Amin et al., 2011, Li and Adalı, 2008]:

$$\nabla_z f = 2\frac{\partial f}{\partial \overline{z}}\,. \tag{2.1}$$

When applying reverse-mode autodiff on the complex domain, some good technical reports can be found, such as [Boeddeker et al., 2017] or [Hunger, 2007]. However, it is left to be verified if *Tensorflow* correctly applies the equations mentioned in these reports. Indeed, no official documentation could be found that asserts the implementation of these equations and *Wirtinger Calculus* when using *Tensorflow* gradient on complex variables. This is not the case with *PyTorch*, where they explicitly say that Wirtinger calculus is used when computing the derivative in the following link: `pytorch.org/docs/stable/notes/autograd.html`. In said link, they indirectly say also that "This convention matches TensorFlow's convention for complex differentiation [...]" referencing the implementation of Equation 2.1.

However, we do know reverse-mode autodiff is the method used by *Tensorflow* [Géron, 2019]. When reverse engineering the gradient definition of *Tensorflow*, the conclusion discussed on the official *Tensorflow*'s GitHub repository issue report 3348 is that the gradient for $f : \mathbb{C} \to \mathbb{C}$ is computed as:

$$\nabla_z f = \overline{\left(\frac{\partial f}{\partial z} + \frac{\partial \overline{f}}{\partial z}\right)} = 2\frac{\partial \mathrm{Re}(f)}{\partial \overline{z}} \,. \tag{2.2}$$

For application purposes, as the loss function is real-valued, we are only interested in cases where $f : \mathbb{C}^n \longrightarrow \mathbb{R}$ for what the above equation can be simplified as:

$$\nabla_z f = 2\frac{\partial f}{\partial \overline{z}} = \left(\frac{\partial f}{\partial x} + i\frac{\partial f}{\partial y}\right) \,, \tag{2.3}$$

which indeed coincides with Wirtinger calculus definition. For this reason, it was not necessary to implement autodiff from scratch, and *Tensorflow*'s algorithm was used instead.

The mathematical equations on how to compute this figure are explained on Appendix B.6 and it's implementation for automatic calculation on a CPU or GPU is described in Appendix C.1 and C.2.

2.3 . Complex Activation functions

One of the essential characteristics of CVNN is its activation functions, which should be non-linear and complex-valued. An activation function is usually chosen to be piece-wise smooth to facilitate the computation of the gradient. The complex domain widens the possibilities to design an activation function, but the probable more natural way would be to extend a real-valued activation function to the complex domain.

Our toolbox currently supports a wide range of complex-activation functions listed on `act_dispatcher` dictionary.

```
act_dispatcher = {
    'linear': linear,
```

```
    # Complex input, real output
    'cast_to_real': cast_to_real,
    'convert_to_real_with_abs': convert_to_real_with_abs,
    'sigmoid_real': sigmoid_real,
    'softmax_real_with_abs': softmax_real_with_abs,
    'softmax_real_with_avg': softmax_real_with_avg,
    'softmax_real_with_mult': softmax_real_with_mult,
    'softmax_of_softmax_real_with_mult':
   softmax_of_softmax_real_with_mult,
    'softmax_of_softmax_real_with_avg':
   softmax_of_softmax_real_with_avg,
    'softmax_real_with_polar': softmax_real_with_polar,
    # Phasor networks
    'georgiou_cdbp': georgiou_cdbp,
    'mvn_activation': mvn_activation,
    'complex_signum': complex_signum,
    # Type A (cartesian)
    'cart_sigmoid': cart_sigmoid,
    'cart_elu': cart_elu,
    'cart_exponential': cart_exponential,
    'cart_hard_sigmoid': cart_hard_sigmoid,
    'cart_relu': cart_relu,
    'cart_leaky_relu': cart_leaky_relu,
    'cart_selu': cart_selu,
    'cart_softplus': cart_softplus,
    'cart_softsign': cart_softsign,
    'cart_tanh': cart_tanh,
    'cart_softmax': cart_softmax,
    # Type B (polar)
    'pol_tanh': pol_tanh,
    'pol_sigmoid': pol_sigmoid,
    'pol_selu': pol_selu,
    # Elementary Transcendental Functions (ETF)
    'etf_circular_tan': etf_circular_tan,
    'etf_circular_sin': etf_circular_sin,
    'etf_inv_circular_atan': etf_inv_circular_atan,
    'etf_inv_circular_asin': etf_inv_circular_asin,
    'etf_inv_circular_acos': etf_inv_circular_acos,
    'etf_circular_tanh': etf_circular_tanh,
    'etf_circular_sinh': etf_circular_sinh,
    'etf_inv_circular_atanh': etf_inv_circular_atanh,
    'etf_inv_circular_asinh': etf_inv_circular_asinh,
    # ReLU
    'modrelu': modrelu,
    'crelu': crelu,
    'zrelu': zrelu,
    'complex_cardioid': complex_cardioid
}
```

Indeed, to implement an activation function, it will suffice to add it to the `act_dispatcher` dictionary to have full functionality.

In our published toolbox, there are two ways of using an activation function, either by using a string listed on `act_dispatcher` like

```
ComplexDense(units=x, activation='cart_sigmoid')
```

or by using the function directly

```
from cvnn.activations import cart_sigmoid

ComplexDense(units=x, activation=cart_sigmoid)
```

This usage also support using `tf.keras.layers.Activation` to implement an activation function directly as an independent layer.

```
    from cvnn.activations import cart_relu

    layer = tf.keras.layers.Activation('cart_relu')
    layer = tf.keras.layers.Activation(cart_relu)
```

Although complex activation functions used on CVNN are numerous [Scardapane et al., 2018, Bassey et al., 2021, Lee et al., 2022], we will mainly focus on two types of activation functions that are an extension of the real-valued functions [Kuroe et al., 2003]:

- Type-A: $\sigma_A(z) = \sigma_{\mathrm{Re}}(\mathrm{Re}(z)) + i\,\sigma_{\mathrm{Im}}(\mathrm{Im}(z))$,
- Type-B: $\sigma_B(z) = \sigma_r(|z|)\,\exp(i\,\sigma_\phi(\arg(z)))$,

where $\sigma_{\mathrm{Re}}, \sigma_{\mathrm{Im}}, \sigma_r, \sigma_\phi$ are all real-valued functions[1]. Re and Im operators are the real and imaginary parts of the input, respectively, and the $\arg$ operator gives the phase of the input. Note that in Type-A, the real and imaginary parts of an input go through nonlinear functions separately, and in Type-B, the magnitude and phase go through nonlinear functions separately.

The most popular activation functions, sigmoid, hyperbolic tangent (tanh) and Rectified Linear Unit (ReLU), are extensible using Type-A or Type-B approach. Although tanh is already defined on the complex domain for what, its transformation is probably less interesting.

Other complex-activation functions are supported by our toolbox including elementary transcentental functions (complex-valued-neural-networks.rtfd.io/en/latest/activations/etf.html) [Kim and Adali, 2001a, Kim and Adali, 2001b] or phasor activation function (complex-valued-neural-networks.rtfd.io/en/latest/activations/mvn_activation.html) such as multi-valued neuron (MVN) activation function [Ajzenberg and Tošić, 1972, Aizenberg et al., 1973] or Georgiou CDBP [Georgiou and Koutsougeras, 1992].

[1] Although not with the same notation, these two types of complex-valued activation functions are also discussed in Section 3.3 of [Hirose, 2012]

2.3.1 . Complex Rectified Linear Unit (ReLU)

Normally, σ_ϕ is left as a linear mapping [Kuroe et al., 2003, Hirose, 2012]. Under this condition, using Rectified Linear Unit (ReLU) activation function for σ_r has a limited interest since the latter makes σ_B converge to a linear function, limiting Type-B ReLU usage. Nevertheless, ReLU has increased in popularity over the others as it has proved to learn several times faster than equivalents with saturating neurons [Krizhevsky et al., 2012]. Consequently, probably the most common complex-valued activation function is Type-A ReLU activation function more often defined as Complex-ReLU or $\mathbb{C}$ReLU [Trabelsi et al., 2017, Cao et al., 2019].

However, several other ReLU adaptations to the complex domain were defined throughout the bibliography as zReLU [Guberman, 2016], defined as

$$\mathrm{zReLU}(z) = \begin{cases} z & \text{if } 0 < \arg(z) < \pi/2 \\ 0 & \text{if otherwise} \end{cases}, \tag{2.4}$$

letting the output as the input only if both real and imaginary parts are positive. Another popular adaptation is modReLU [Arjovsky et al., 2016], defined as

$$\mathrm{modReLU}(z) = \begin{cases} \mathrm{ReLU}\left(|z| + b\right) \dfrac{z}{|z|} & \text{if } |z| \geq b \\ 0 & \text{if otherwise} \end{cases}, \tag{2.5}$$

where b is an adaptable parameter defining a radius along which the output of the function is 0. This function provides a point-wise non-linearity that affects only the absolute value of a complex number. Another extension of ReLU, is the complex cardioid proposed by [Virtue et al., 2017]

$$\sigma(z) = \frac{\left(1 + \cos\left(\arg(z)\right)\right) z}{2}. \tag{2.6}$$

This function maintains the phase information while attenuating the magnitude based on the phase itself.

These last three activation functions (cardioid, zReLU and modReLU) were analyzed and compared against each other in [Scardapane et al., 2018].

The discussed variants were implemented in the toolbox documented as usual in complex-valued-neural-networks.rtfd.io/en/latest/activations/relu.html.

2.3.2 . Output layer activation function

The image domain of the output layer depends on the set of data labels. For classification tasks, real-valued integers or binary numbers are frequently used to label each class. For these cases, one option would be to cast the labels to the complex domain as done in [Zhang et al., 2017], where a transformation is done to a label $c \in \mathbb{R}$ like $T : c \to c + i\,c$.

The second option is to use an activation function $\sigma : \mathbb{C} \to \mathbb{R}$ as the output layer. A popular real-valued activation used for classification tasks is the *softmax* function [Goodfellow et al., 2016] (normalized exponential), which maps the magnitude to $[0, 1]$, so the image domain is homogeneous to a probability. There are several options on how to transform this function to accept complex input and still have its image $\in [0; 1]$. These options include either performing an average of the magnitudes $\sigma_{\text{Re}}, \sigma_{\text{Im}}$ or σ_r, σ_ϕ, using only one of the magnitudes like σ_r or apply the real-valued *softmax* to either the addition or multiplication of $\sigma_{\text{Re}}, \sigma_{\text{Im}}$ or σ_r, σ_ϕ, between other options. Most of these variants are implemented in the library detailed in this Chapter and documented in complex-valued-neural-networks.rtfd.io/en/latest/activations/real_output.html.

2.4 . Complex-compatible Loss functions

For CVNNs, the loss or cost function to minimize will have a real-valued output as one can not look for the minimum of two complex numbers. If the application is that of classification or semantic segmentation (as it is in all the cases of study of this work), there are a few options on what to do.

Some loss functions support this naturally. Reference [Hänsch, 2010] compares the performance of different type of complex input compatible loss functions. If the loss function to be used does not support complex-valued input, a popular option is to manage this through the output activation function as explained in the previous Section 2.3.2. However, a second option for non-complex-compatible loss functions such as *categorical cross-entropy* is to compare both the real and imaginary parts of the prediction independently with the labels and compute the loss function as an average of both results. Reference [Cao et al., 2019], for example, defines a loss function as the complex average cross-entropy as:

$$\mathcal{L}^{ACE} = \frac{1}{2}\left[\mathcal{L}^{CCE}\left(\text{Re}(y), d\right) + \mathcal{L}^{CCE}\left(\text{Im}(y), d\right)\right] , \tag{2.7}$$

where $\mathcal{L}^{ACE}$ is the complex average cross-entropy, $\mathcal{L}^{CCE}$ is the well-known categorical cross-entropy. y is the network predicted output, and d is the corresponding ground truth or desired output. For real-valued output $\mathcal{L}^{ACE} = \mathcal{L}^{CCE}$. This function was implemented in the published code alongside other variants, such as multiplying each class by weight for imbalanced classes or ignoring unlabeled data. All these versions are documented in complex-valued-neural-networks.rtfd.io/en/latest/losses.htm.

When the desired output is already complex-valued (regression tasks), more natural definitions can be used, such as proposed by [Bassey et al., 2021], where the loss is defined as

$$\mathcal{L} = \frac{1}{2}\sum_k e_k \overline{e_k} , \tag{2.8}$$

where $e_k(y_k, d_k)$ is a complex error computation of y_k and d_k such as a subtraction.

2.5 . Complex Batch Normalization

The complex Batch Normalization (BN) was adapted from the real-valued BN technique by Reference [Trabelsi et al., 2017]. For normalizing a complex vector, we will treat the problem as a 2D vector instead of working on the complex domain so that $z = a + i\,b \in \mathbb{C} \longrightarrow \boldsymbol{x} = (a, b) \in \mathbb{R}^2$.

To normalize a complex variable, we need to compute

$$\boldsymbol{o} = \hat{\boldsymbol{\Sigma}}^{-\frac{1}{2}}(\boldsymbol{x} - \hat{\boldsymbol{\mu}})\,, \tag{2.9}$$

where $\boldsymbol{o}$ is the normalized output, $\hat{\boldsymbol{\mu}}$ is the mean estimate of $\mathbb{E}[\boldsymbol{x}]$, and $\hat{\boldsymbol{\Sigma}} \in \mathbb{R}^{2\times 2}$ is the estimated covariance matrix of $\boldsymbol{x}$ so that

$$\hat{\boldsymbol{\Sigma}} = \begin{bmatrix} \Sigma_{rr} & \Sigma_{ri} \\ \Sigma_{ir} & \Sigma_{ii} \end{bmatrix} = \begin{bmatrix} \mathrm{Cov}(\mathrm{Re}(x)\mathrm{Re}(x)) & \mathrm{Cov}(\mathrm{Re}(x)\mathrm{Im}(x)) \\ \mathrm{Cov}(\mathrm{Im}(x)\mathrm{Re}(x)) & \mathrm{Cov}(\mathrm{Im}(x)\mathrm{Im}(x)) \end{bmatrix}. \tag{2.10}$$

During the batch normalization layer initialization, two variables $\boldsymbol{\Sigma'} \in \mathbb{R}^{2\times 2}$ (moving variance) and $\boldsymbol{\mu'} \in \mathbb{R}^2$ (moving mean) are initialized. By default, $\boldsymbol{\Sigma'} = \mathbf{I}/\sqrt{2}$ and $\boldsymbol{\mu}'$ is initialized to zero.

During the training phase, $\hat{\boldsymbol{\Sigma}}$ and $\hat{\boldsymbol{\mu}}$ are computed on the innermost dimension of the training input batch (for multi-dimensional inputs where $z \in \mathbb{C}^N \rightarrow x \in \mathbb{R}^{N\times 2}$). The output of the layer is then computed as in Equation 2.9. The moving variance and moving mean are iteratively updated using the following rule:

$$\boldsymbol{\mu'}_{k+1} = m\,\boldsymbol{\mu'}_k + (1-m)\,\hat{\boldsymbol{\mu}}_k \tag{2.11}$$

$$\boldsymbol{\Sigma'}_{k+1} = m\,\boldsymbol{\Sigma'}_k + (1-m)\,\hat{\boldsymbol{\Sigma}}_k\,, \tag{2.12}$$

where m is the momentum, a constant parameter set to 0.99 by default.

During the inference phase, that is, for example, when performing a prediction, no variance nor average is computed. The output is directly calculated using the moving variance and moving average as

$$\hat{\boldsymbol{x}} = \boldsymbol{\Sigma'}^{-\frac{1}{2}}(\boldsymbol{x} - \boldsymbol{\mu'})\,. \tag{2.13}$$

Analogously to the real-valued batch normalization, it is possible to shift and scale the output by using the trainable parameters $\boldsymbol{\beta}$ and $\boldsymbol{\Gamma}$. In this case, the output $\boldsymbol{o}$ for both the training and prediction phase will be changed to $\hat{\boldsymbol{o}} = \boldsymbol{\Gamma}\,\boldsymbol{o} + \boldsymbol{\beta}$. By default, $\boldsymbol{\beta}$ is initialized to $(0, 0)^T \in \mathbb{R}^2$ and

$$\boldsymbol{\Gamma} = \begin{pmatrix} 1/\sqrt{2} & 0 \\ 0 & 1/\sqrt{2} \end{pmatrix}. \tag{2.14}$$

2.6 . Complex Random Initialization

If we are to blindly apply any well-known random initialization algorithm to both real and imaginary parts of each trainable parameter independently, we might

lose the special properties of the used initialization. This is the case, for example, for Glorot, also known as Xavier, initializer [Glorot and Bengio, 2010].

Assuming that:

- The input features have the same variance $\mathrm{Var}\left[X_i^{(0)}\right] \triangleq \mathrm{Var}\left[X^{(0)}\right], \forall i \in [|1; N_0|]$ and have zero mean (can be adjusted by the bias input).
- All the weights are statistically centered, and there is no correlation between real and imaginary parts.
- The weights at layer l share the same variance $\mathrm{Var}\left[\omega_{i,j}^{(l)}\right] \triangleq \mathrm{Var}\left[\omega^{(l)}\right], \forall (i,j) \in [|1; N_{l+1}|] \times [|1; N_l|]$ and are statistically independent of the others layer weights and of inputs $X^{(0)}$.
- We are working on the linear part of the activation function. Therefore, $\sigma(z) \approx z$, which is the same as saying that $\sigma(z, \overline{z}) \approx z$. The partial derivatives will then be

$$\begin{cases} \dfrac{\partial \sigma}{\partial z} \approx 1 \\ \dfrac{\partial \sigma}{\partial \overline{z}} \approx 0 \end{cases} \tag{2.15}$$

so that $\dfrac{\partial \sigma\left(V_n^{(l)}\right)}{\partial V_n^{(l)}} \approx 1$ and $\dfrac{\partial \overline{\sigma\left(V_n^{(l)}\right)}}{\partial V_n^{(l)}} \approx 0$, with $V_n^{(l)}$ defined in Section 1.3.1. This is an acceptable assumption when working with logistic sigmoid or tanh activation functions.

Using the notation of Section 1.3.1, for a dense feed-forward neural network with a bias initialized to zero (as is often the case), each neuron at hidden layer l is expressed as

$$X_n^{(l)} \triangleq \sigma\left(V_n^{(l)}\right) = \sigma\left(\sum_{m=1}^{N_{l-1}} \omega_{nm}^{(l)} X_m^{(l-1)}\right), \forall n \in [|1; N_l|]. \tag{2.16}$$

Since σ is working on the linear part, from (2.16) we get that

$$\mathrm{Var}\left[X_n^{(l)}\right] = \mathrm{Var}\left[\sum_{m=1}^{N_{l-1}} \omega_{nm}^{(l)} X_m^{(l-1)}\right], \tag{2.17}$$

where $X_m^{(l-1)}$ is a combination of $\omega^{(k)}, 1 \leq k \leq l-1$ and $x^{(0)}$, so it is independent of $\omega^{(l)}$ which leads to

$$\mathrm{Var}\left[X_n^{(l)}\right] = \sum_{m=1}^{N_{l-1}} \mathrm{Var}\left[\omega_{nm}^{(l)}\right] \mathrm{Var}\left[X_m^{(l-1)}\right], \tag{2.18}$$

As the weights share the same variance at each layer,

$$\begin{aligned}\mathrm{Var}\left[X_n^{(l)}\right] &= \mathrm{Var}\left[\omega^{(l)}\right]\sum_{m=1}^{N_{l-1}}\mathrm{Var}\left[X_m^{(l-1)}\right], \\ &= \mathrm{Var}\left[\omega^{(l)}\right]\mathrm{Var}\left[\omega^{(l-1)}\right]\sum_{m=1}^{N_{l-1}}\sum_{p=1}^{N_{l-2}}\mathrm{Var}\left[X_p^{(l-2)}\right], \\ &= N_{l-1}\mathrm{Var}\left[\omega^{(l)}\right]\mathrm{Var}\left[\omega^{(l-1)}\right]\sum_{p=1}^{N_{l-2}}\mathrm{Var}\left[X_p^{(l-2)}\right]. \end{aligned} \tag{2.19}$$

We can now obtain the variance of $X_n^{(l)}$ as a function of $x^{(0)}$ by applying Equation (2.19) recursively and assuming $X_n^{(0)}, n = 1, \ldots, N_0$ sharing the same variance,

$$\mathrm{Var}\left[X_n^{(l)}\right] = \mathrm{Var}\left[x^{(0)}\right]\prod_{m=1}^{l} N_{m-1}\mathrm{Var}\left[\omega^{(m)}\right]. \tag{2.20}$$

From a forward-propagation point of view, to keep a constant flow of information, then

$$\mathrm{Var}\left[X_n^{(l)}\right] = \mathrm{Var}\left[X_n^{(l')}\right], \forall 1 \leq l < l' \leq N \tag{2.21}$$

which implies that $N_{m-1}\mathrm{Var}\left[\omega^{(m)}\right] = 1, \forall 1 \leq m \leq N$.

On the other hand,

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial V_n^{(l)}} &= \sum_{k=1}^{N_{l+1}}\frac{\partial\mathcal{L}}{\partial V_k^{(l+1)}}\frac{\partial V_k^{(l+1)}}{\partial X_n^{(l)}}\frac{\partial X_n^{(l)}}{\partial V_n^{(l)}} + \frac{\partial\mathcal{L}}{\partial\overline{V_k^{(l+1)}}}\frac{\partial\overline{V_k^{(l+1)}}}{\partial X_n^{(l)}}\frac{\partial X_n^{(l)}}{\partial V_n^{(l)}} + R, \\ &= \sum_{k=1}^{N_{l+1}}\frac{\partial\mathcal{L}}{\partial V_n^{(l+1)}}\omega_{k,n}^{(l+1)}. \end{aligned} \tag{2.22}$$

where R is the remaining terms depending on $\frac{\partial\overline{X_n^{(l)}}}{\partial V_n^{(l)}}$, which is assumed to be 0 because the activation function is working in the linear regime at the initialization, i.e. $\frac{\partial X_n^{(l)}}{\partial V_n^{(l)}} \approx 1$ and $\frac{\partial\overline{X_n^{(l)}}}{\partial V_n^{(l)}} \approx 0$. The last equality is held since $\frac{\partial V_k^{(l+1)}}{\partial X_n^{(l)}} = \omega_{k,n}^{(l+1)}$ and $\frac{\partial V_k^{(l+1)}}{\partial\overline{X_n^{(l)}}} = 0$ (see (B.60) in Appendix for the general case).

Assuming the loss variation w.r.t. the output neuron is statistically independent of any weights at any layers, then we can deduce recursively from (2.22),

$$\mathrm{Var}\left[\frac{\partial\mathcal{L}}{\partial V_n^{(l)}}\right] = \mathrm{Var}\left[\frac{\partial\mathcal{L}}{\partial V_n^{(L)}}\right]\prod_{m=l+1}^{L} N_m\mathrm{Var}\left[\omega^{(m)}\right]. \tag{2.23}$$

From a back-propagation point of view, we want to keep a constant learning flow:

$$\mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial V_n^{(l)}}\right] = \mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial V_n^{(l')}}\right], \forall 1 \leq l < l' \leq N, \tag{2.24}$$

which implies $N_m \mathrm{Var}\left[\omega^{(m)}\right] = 1, \forall 1 \leq m \leq N$.

Conditions (2.21) and (2.24) are not possible to be satisfied at the same time (unless $N_l = N_{l+1}, \forall 1 \leq l < N$, meaning all layers should have the same width) for what Reference [Glorot and Bengio, 2010] proposes the following trade-of

$$\mathrm{Var}\left[\omega^{(l)}\right] = \frac{2}{N_l + N_{l+1}}, \forall 1 \leq l < N\,. \tag{2.25}$$

If the weight initialization is a uniform distribution $\sim U$, for the real-valued case, the initialization that has the variance stated on Equation 2.25 is:

$$\omega^{(l)} \sim U\left[-\frac{\sqrt{6}}{\sqrt{N_l + N_{l+1}}}, \frac{\sqrt{6}}{\sqrt{N_l + N_{l+1}}}\right]. \tag{2.26}$$

For a complex variable with no correlation between real and imaginary parts, the variance is defined as:

$$\mathrm{Var}\left[\omega^{(l)}\right] = \mathrm{Var}\left[\mathrm{Re}\left(\omega^{(l)}\right)\right] + \mathrm{Var}\left[\mathrm{Im}\left(\omega^{(l)}\right)\right], \tag{2.27}$$

it is therefore logical to choose both variances $\mathrm{Var}\left[\mathrm{Re}\left(\omega^{(l)}\right)\right]$ and $\mathrm{Var}\left[\mathrm{Im}\left(\omega^{(l)}\right)\right]$ to be equal:

$$\mathrm{Var}\left[\mathrm{Re}\left(\omega^{(l)}\right)\right] = \mathrm{Var}\left[\mathrm{Im}\left(\omega^{(l)}\right)\right] = \frac{1}{N_l + N_{l+1}}\,. \tag{2.28}$$

With this definition, the complex variable could be initialized as:

$$\mathrm{Re}\left(\omega^{(l)}\right) = \mathrm{Im}\left(\omega^{(l)}\right) \sim U\left[-\frac{\sqrt{3}}{\sqrt{N_l + N_{l+1}}}, \frac{\sqrt{3}}{\sqrt{N_l + N_{l+1}}}\right]. \tag{2.29}$$

By comparing (2.26) with (2.29) it is concluded that to correctly implement a Glorot initialization, one should divide the real and imaginary part of the complex weight by $\sqrt{2}$.

It is also possible to define the initialization technique from a polar perspective. The variance definition is

$$\mathrm{Var}\left[\omega^{(l)}\right] = E\left[\left|\omega^{(l)} - E\left[\omega^{(l)}\right]^2\right|\right] = E\left[\left|\omega^{(l)}\right|^2\right] + \left|E\left[\omega^{(l)}\right]\right|^2. \tag{2.30}$$

By choosing the phase to be a uniform distribution between 0 and 2π and knowing the absolute is always positive, then $E\left[\omega^{(l)}\right] = 0$ and Equation 2.30 can be simplified to:

$$\mathrm{Var}\left[\omega^{(l)}\right] = E\left[\left|\omega^{(l)}\right|^2\right]. \tag{2.31}$$

It will therefore suffice to choose any random initialization, such as, for example, a Rayleigh distribution [Rayleigh, 1880], for $\left|\omega^{(l)}\right| = \rho \in \mathbb{R}_0^+$ so that

$$E\left[\rho^2\right] = \frac{2}{N_l + N_{l+1}}\,. \tag{2.32}$$

2.6.1 . Impact of complex-initialization equation application

A simulation was done for a complex multi-layer network with four hidden layers of size 128, 64, 32 and 16, respectively, with a logistic sigmoid activation function to test the impact of this constant division by $\sqrt{2}$ on a signal classification task. One-hundred and fifty epochs were done with one thousand runs of each model to obtain statistical results.

The task consisted in classifying different signals used in radar applications. Temporal and time-frequency representations of each signal are shown in Figures 2.9 and 2.10 respectively. The generated signals are

- Sweep or chirp signal. These are signals whose frequency changes over time. These types of signals are commonly applied to radar. The chirp-generated signals were of two types, either linear chirp, where the frequency changed linearly over time or S-shaped, whose frequency variation gets faster at both the beginning and the end, forming an S-shaped spectrum as can be seen in Figure 2.10.
- Phase-Shift Keying (PSK) modulated signals, a digital modulation process that conveys data by changing the phase of a constant frequency reference signal. These signals were BPSK (2-phase states) and QPSK (4-phase states)
- Quadrature Amplitude Modulation (QAM) signals, which are a combination of amplitude and phase modulation. These signals were 16QAM (4 phase and amplitude states) and 64QAM (8 phase and amplitude states).

A noise signal (null), without any signal of interest, was also used, making a total of 7 different classes.

Instead of using measured signals, and with the goal of facilitating the studies, the signals were randomly generated, which also allowed having the ground truth. These generated signals had the following properties:

- 256 samples per signal
- Peak-to-peak amplitude of 1
- Chirp signals with frequencies from 0.05 to 0.45 times the sample frequency
- Number of moments between 8 and 64 for the codes BPSK, QPSK, 16QAM and 64QAM

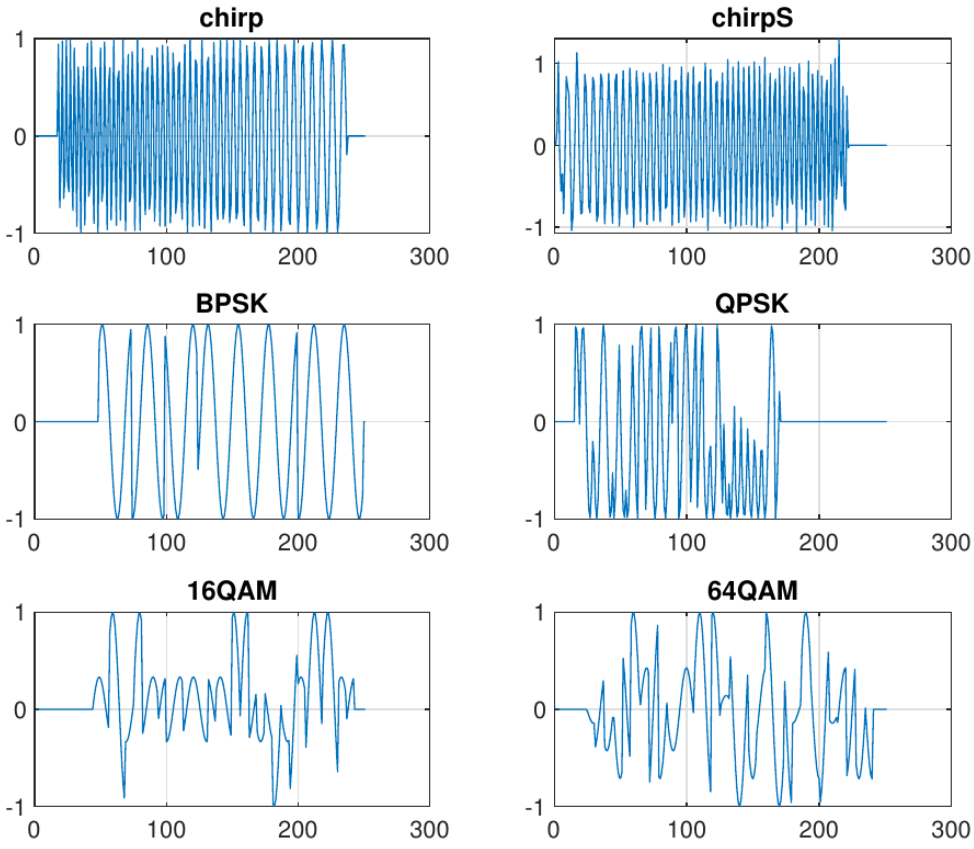


Figure 2.9: Temporal amplitude examples of used signals [Vieillard, 2018].

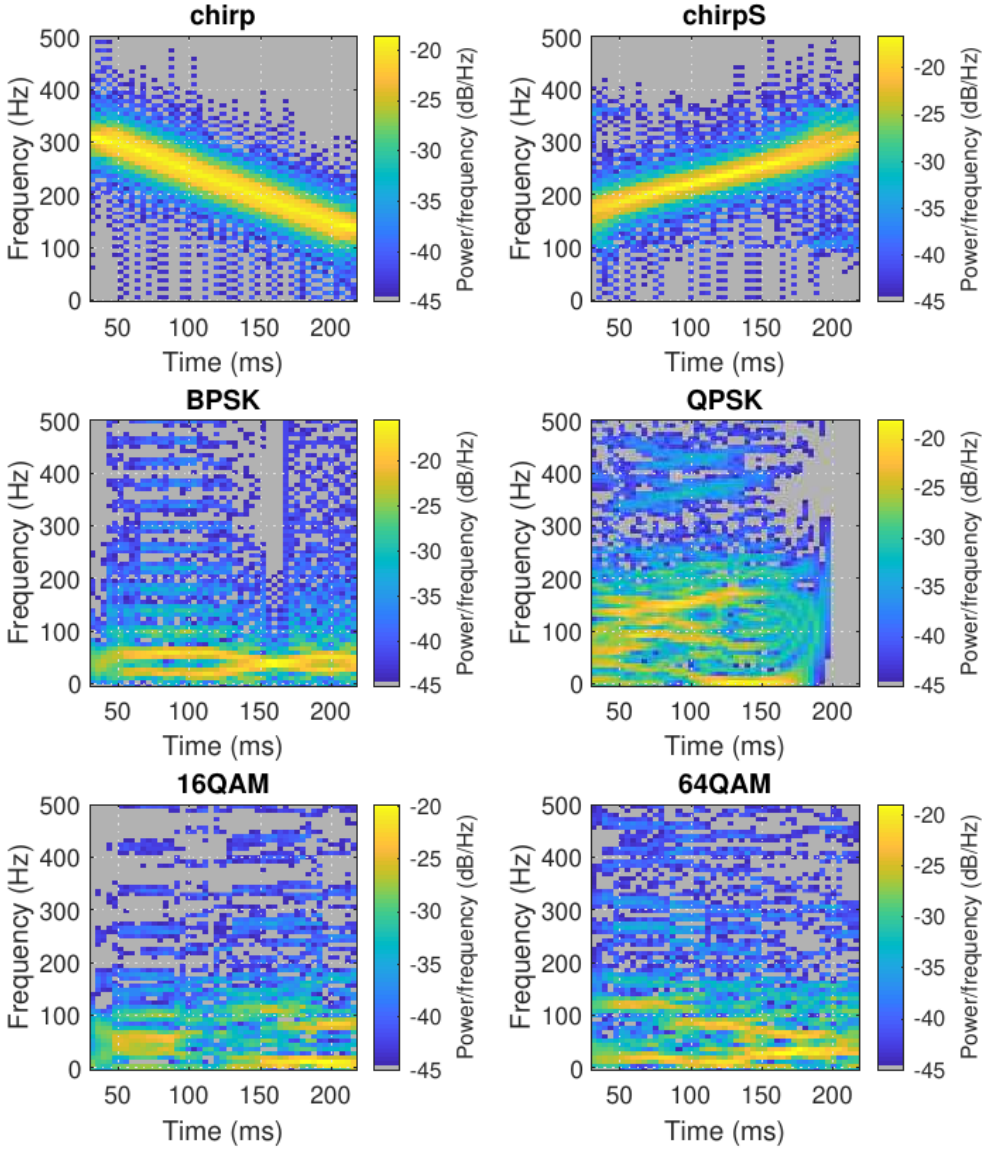


Figure 2.10: Spectrogram examples of used signals [Vieillard, 2018].

Thermal noise was added to each signal and was transformed to the complex domain using the Hilbert Transform, a popular transformation in signal processing applications [Hahn, 1996], which provides an analytic mapping of a real-valued function to the complex plane.

The Hilbert transpose has its origins in 1902 when the English mathematician Godfrey Harold Hardy (1877 – 1947) introduced a transformation that consisted in the convolution of a real function $f(s)$ [Hardy, 1902, Hardy, 1909], with the Cauchy kernel $1/\pi(t-s)$ which, being an improper integral, must be defined in terms of its principal value (p.v.) [King, 2009],

$$\mathcal{H}(f)(t) = \frac{1}{\pi}\text{p.v.}\int_{-\infty}^{+\infty}\frac{f(s)}{t-s}ds \quad = \frac{1}{\pi}\lim_{\varepsilon\to 0}\int_{\varepsilon}^{+\infty}\frac{f(t-s)-f(t+s)}{s}ds\,. \tag{2.33}$$

One of the most important properties of this transformation is that its repeated application allows for the recovery of the original function, with only a change of sign, that is,

$$g(t) = \mathcal{H}(f)(t) \Leftrightarrow f(t) = -\mathcal{H}(g)(t)\,. \tag{2.34}$$

The functions f and g that satisfy this relation are called Hilbert transform pairs, in honor of David Hilbert, who first studied them in 1904 [Hilbert, 1912]. In fact, it is for this reason that in 1924 [Hardy, 1924b, Hardy, 1924a], Hardy graciously proposed calling transformation (2.33) as Hilbert Transform.

Some examples of Hilbert transform pairs are shown in Table 2.1.

$f(t)$	$g(t) = \mathcal{H}(f)(t)$
$\sin(t)$	$\cos(t)$
$1/\left(t^2+1\right)$	$t/\left(t^2+1\right)$
$\sin(t)/t$	$\left[1-\cos(t)\right]/t$
$\delta(t)$	$1/\pi t$

Table 2.1: Examples of Hilbert transform pairs

Considering the definition of the Fourier transform of an absolutely integrable real function $f(s)$,

$$\mathcal{F}(f)(t)\,. \tag{2.35}$$

It can be shown that [King, 2009]

$$\mathcal{F}\left(\mathcal{H}(f)(t)\right) = -i\,\text{sgn}(t)\mathcal{F}(f)(t)\,. \tag{2.36}$$

This relation provides an effective way to evaluate the Hilbert transform

$$\mathcal{H}(f) = -i\,\mathcal{F}^{-1}\left[\text{sgn}(t)\mathcal{F}(f)(t)\right]\,, \tag{2.37}$$

avoiding the issue of dealing with the singular structure of the Cauchy kernel.

One of the most important properties of the Hilbert transform, at least in reference to this thesis, is that the real and imaginary parts of a function $h(z)$ that is analytic in the upper half of the complex plane are Hilbert transform pairs. That is to say that

$$\mathrm{Im}(h) = \mathcal{H}(\mathrm{Re}(h))\,. \tag{2.38}$$

In this way, the Hilbert transform provides a simple method of performing the analytic continuation to the complex plane of a real function $f(x)$ defined on the real axis, defining $h(z) = f(z) + i\,g(z)$ with $g(z) = \mathcal{H}(f)$. This property of the Hilbert transform was independently discovered by Ralph Kronig [Kronig, 1926] (1904 – 1995), and Hans Kramers [Kramers, 1927] (1894 - 1952) in 1926 in relation to the response function of physical systems, known as the Kramers-Kronig relation. At the same time, it began to be used in circuit analysis [Carson, 1926] in relation to the real and imaginary parts of the complex impedance. Through the work of pioneers such as the Nobel prize winner Dennis Gabor [Gabor, 1946] (1900 – 1979), its application in modern signal processing is wide and varied [Hahn, 1996].

The real and imaginary weights where initialized as described in (2.26) (The definition for real-valued weights, which is equivalent to multiplying the limits of Equation 2.29 by $\sqrt{2}$) and (2.29) (the original case for complex-valued weights) to compare them. An initialization that divided the limits of Equation 2.29 by two was also used to assert that smaller values will not produce a superior result either.

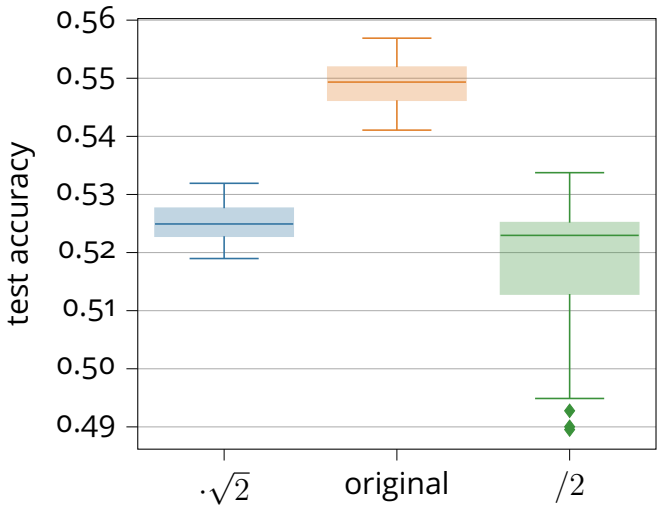


Figure 2.11: Comparison of Glorot Uniform initialization scaled by different values.

The results shown in Figure 2.11 prove the importance of the correct adaptation of Glorot initialization to complex numbers and how failing to do so will impact its performance negatively.

2.6.2 . Experiment on different trade-offs

Complex numbers enable choosing different trade-offs than the one chosen by [Glorot and Bengio, 2010] (Equation 2.25), for example, the following trade-off can also be chosen:

$$\begin{aligned} \operatorname{Var}\left[\operatorname{Re}\left(\omega^{(l)}\right)\right] &= \frac{1}{2N_l}\,, \\ \operatorname{Var}\left[\operatorname{Im}\left(\omega^{(l)}\right)\right] &= \frac{1}{2N_{l+1}}\,, \end{aligned} \tag{2.39}$$

between other options.

In a similar manner, as we did with Glorot (Xavier) initialization, the He weight initialization described in [He et al., 2015] can be deduced for complex numbers. the same dataset as the one used in the experiment of Figure 2.11 was done using Glorot Uniform (GU), Glorot Normal (GN), He Normal (HN), He Uniform (HU), and Glorot Uniform using the trade-off defined in (2.39) (GU_C).

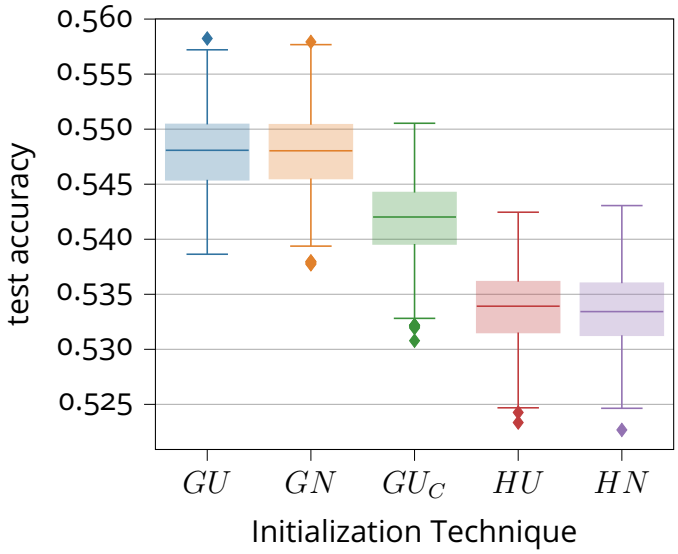


Figure 2.12: Initialization Technique Comparison

The results of Figure 2.12 show that the difference between using a normal or uniform distribution is negligible but that Glorot clearly outperforms He initialization. This is to be expected when using sigmoid activation function because He initialization was designed for ReLU activation functions. On the other hand, the trade-off proposed in (2.39) actually achieves lower performances than the one proposed in [Glorot and Bengio, 2010] for what it was not implemented in the cvnn toolbox, however, it is possible this results is only specific to this dataset and for what further research could be done with this trade-off variant. All the other initialization techniques are documented in `complex-valued-neural-networks.rtfd.io/en/latest/initializers.html` and implemented as previously described. They can be used in standalone mode as follows

```python
import cvnn
initializer = cvnn.initializers.GlorotUniform()
values = initializer(shape=(2, 2))
# Returns a complex Glorot Uniform tensor of shape (2, 2)
```

or inside a layer using an initializer object like

```python
import cvnn
initializer = cvnn.initializers.ComplexGlorotUniform()
layer = cvnn.layers.Dense(input_size=23, output_size=45,
    weight_initializer=initializer)
```

or as a string listed within `init_dispatcher` like

```python
import cvnn

layer = cvnn.layers.Dense(input_size=23, output_size=45,
    weight_initializer="ComplexGlorotUniform")
```

with `init_dispatcher` being

```python
init_dispatcher = {
    "ComplexGlorotUniform": ComplexGlorotUniform,
    "ComplexGlorotNormal": ComplexGlorotNormal,
    "ComplexHeUniform": ComplexHeUniform,
    "ComplexHeNormal": ComplexHeNormal
}
```

2.7 . Performance on real-valued data

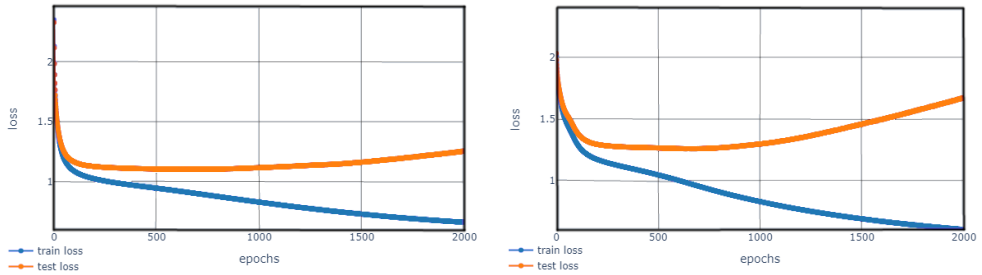
Using the same signals of previous Sections, some experiments were perform comparing Complex-Valued MultiLayer Perceptron (CV-MLP) against a Real-Valued MultiLayer Perceptron (RV-MLP). 16000 Chirp signals were created, 8000 linear and 8000 S-shaped. 80% was used for training, 10% for validation and the remaining 10% for testing. Two models (one CV-MLP and one RV-MLP) were designed and dimensioned as shown in Table 2.2.

	CV-MLP	RV-MLP
input size	256	512
hidden layers activation	Type-A Selu	Selu
1^{st} hidden layer size	25	50
2^{nd} hidden layer size	10	20
output activation	modulo *softmax*	modulo *softmax*
output size	7	7

Table 2.2: Design of MLP models

We used SGD as weight optimization and 50% dropout for both models. We performed 2000 iterations (1000 for each model) with 2000 epochs each.

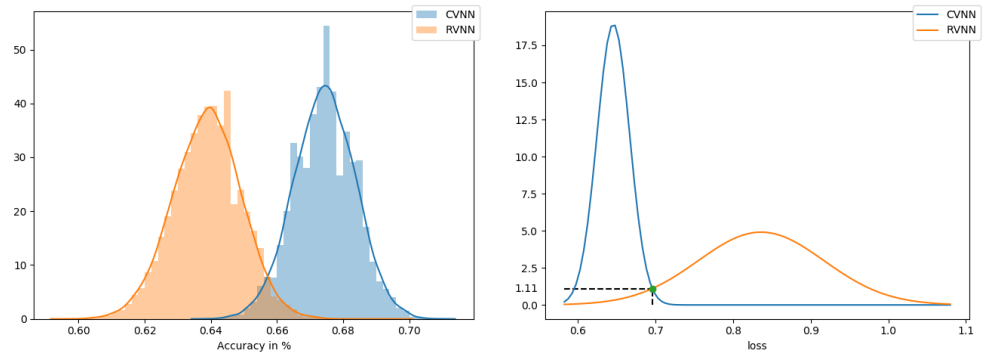
Figure 2.13 shows the mean loss per epoch of both training and validation set for CV-MLP (Figure 2.13a) and RV-MLP (Figure 2.13b). The Figures show that CV-MLP presents less overfitting than the real-valued model.



(a) CV-MLP (b) RV-MLP

Figure 2.13: Mean loss evolution per epoch.

A histogram of both models' accuracy and loss values on the test set was plotted and can be seen in Figure 2.14. It is clear that CV-MLP outperforms RV-MLP classification accuracy with around 4% higher accuracy. Regarding the loss, RV-MLP obtained higher variance.

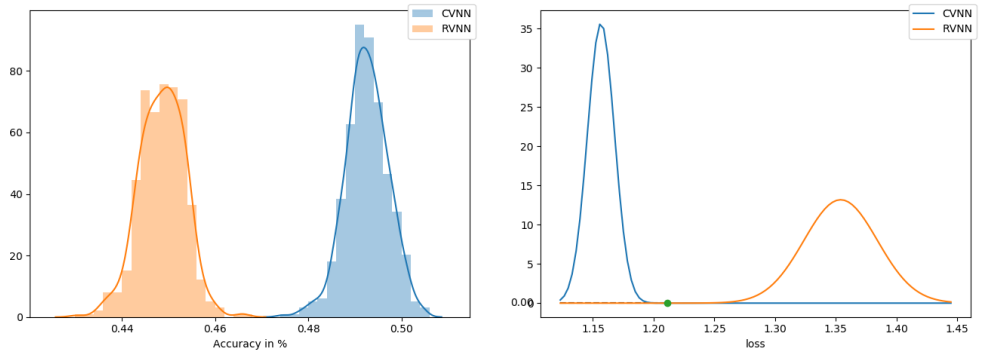


(a) Accuracy (b) Loss

Figure 2.14: Test set histogram metrics for binary classification on Chirp signals.

Finally, the simulations were performed for all seven signals obtaining similar results as before. This time, accuracy results have a higher difference with CVNN results not intersecting with the RVNN results. Again, RV-MLP had higher loss variance.

2.7.1 . Discussion



(a) Accuracy (b) Loss

Figure 2.15: Test set histogram metrics for multi-class classification for all 7 signals.

It is important to note that these networks were not optimized. The main issue is the *softmax* activation function used on the absolute value of the complex output. Although it is not generally used like this, in CVNN, it might be acceptable. However, for a RVNN, this is unconventional and penalizes their performance greatly. Furthermore, both models are not equivalent, as will be explained in Chapter 3.2.1, resulting in RVNN having a higher capacity, which may increase their performance but also may result in more overfitting.

For all these reasons, the simulations must be revised. However, if the general conclusions stand, this might indicate that CVNN can outperform RVNN even for real-valued applications when using an appropriate transformation such as the Hilbert transform.

2.8 . Conclusion

In this chapter, we described in detail the implementation of the published library, with examples of how to use the code and with reference to the documentation to be used if needed. We showed that the library had great success in the community through its increasing popularity.

We also described in detail each adaptation from conventional neural networks to the complex plane in order to be able to implement Complex-Valued Neural Networks. Each aspect was revised, and the appropriate mathematics was developed. With this Chapter it should be possible to understand and implement from a basic Complex-Valued MultiLayer Perceptron to a Complex-Valued Convolutional Neural Network and even Complex-Valued Fully Convolutional Neural Network or Complex-Valued U-Network (CV-UNET).

We performed simulations that verified the adaptation of the initialization technique and showed that correctly implementing this initialization is crucial for ob-

taining a good performance.

Finally, we show that CVNNs might be of interest even for real data by using the Hilbert transformation contrary to the work of [Mönning and Manandhar, 2018]. The classification improved around 4% when using a complex network over a real one. However, these last results should be revised as the models were not equivalent, and RVNN might have been overly penalized.

3 - Interest in CVNN for classifying non-circular data

In general, we can define a bijective map between complex-valued and real-valued data using a simple concatenation of real and imaginary part or the Hilbert transform. So we cannot derive universal rules in practice to prioritize CVNN over RVNN for complex-valued datasets. Performance highly depends on the characteristics of the complex-valued dataset. Indeed, merely taking real data as input does not profit from using CVNN [Mönning and Manandhar, 2018]. Furthermore, the hypothesis of circularity is not always satisfied, as shown in [Vasile and Totir, 2012, El-Darymli et al., 2014] for SAR images which depend on the region of interest. Therefore, non-circularity parameters are key factors in improving performance in radar estimation and classification tasks, as proposed in [Barbaresco and Chevalier, 2008, Wu et al., 2016].

We thus analyze the influence of the non-circular statistical property on the performance of both CVNN and RVNN networks. We show that particular structures of complex data, such as, for example, phase information or statistical correlation between real and imaginary parts, can notably benefit from using CVNN compared to its real-valued equivalent model. Under this context, CVNN is potentially an attractive network to obtain better classification performance on complex datasets.

The Chapter is organized as follows: In Section 3.1, we discuss the circularity property for a random variable. Section 3.2 discusses the feed-forward network architecture and the dataset used for these experiments. Section 3.3 illustrates the comparison of statistical performance obtained for CVNN and RVNN networks. In particular, the sensitivity of CVNN and RVNN results are evaluated either by changing the dataset characteristics or the network hyper-parameters.

Although CVNN is an acronym that involves numerous complex-valued neural network architectures, in this Chapter, we will always be referring to a fully connected feed-forward neural network or a MultiLayer Perceptron (MLP) in accordance with the existing bibliography [Hirose and Yoshida, 2012, Hänsch and Hellwich, 2009a, Hirose, 2012, Hirose, 2009, Amin et al., 2011, Hirose, 2013].

3.1 . Circularity

The importance of circularity for CVNNs has already been mentioned by [Hirose, 2012, Hirose, 2013]. We could define a complex random variable to be circular if, for $Z = X + iY$, $\exists\, z_0 = x_0 + i\, y_0$ from which by rotating the distribution Z using z_0 as the center, the distribution is invariant. Note that if z_0 exists, it will be the location of the distribution and unique. We can therefore use a coordinate system that its origin is coincident with the distribution location so that

$\boldsymbol{\mu} = (\mathbb{E}[X], \mathbb{E}[Y])^T = \mathbf{0}$. In this coordinate system, a complex-valued distribution Z will be circular $\Leftrightarrow$ $Z \overset{(\mathrm{d})}{=} Z \exp(i\,\theta), \forall \theta \in [0; 2\pi[$. Let us denote the vector $\boldsymbol{u} \triangleq (X, Y)^T$ as the real random vector built by stacking the real and imaginary parts of a complex Normal random variable $Z = X + i\,Y$. The probability density function (pdf) of Z can be identified through the pdf $p_{\boldsymbol{u}}$ of $\boldsymbol{u}$:

$$p_{\boldsymbol{u}}(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{\mu}, \Sigma) = \frac{1}{2\pi|\Sigma|} \exp\left[-\frac{1}{2}(\boldsymbol{u} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{u} - \boldsymbol{\mu})\right], \tag{3.1}$$

with $\boldsymbol{\mu} = (\mathbb{E}[X], \mathbb{E}[Y])^T$ and where $\boldsymbol{\Sigma}$ is the covariance matrix:

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_X^2 & \sigma_{XY} \\ \sigma_{YX} & \sigma_Y^2 \end{pmatrix}. \tag{3.2}$$

where σ_X^2 and σ_Y^2 are respectively the *variance* of X and Y. The *variance* of the complex variable Z can also be defined as a function of σ_X and σ_Y as:

$$\sigma_Z^2 \triangleq \mathbb{E}\left[|Z - \mathbb{E}[Z]|^2\right] = \sigma_X^2 + \sigma_Y^2, \tag{3.3}$$

The latter, however, does not bring information about the *covariance*:

$$\sigma_{XY} \triangleq \mathbb{E}\left[(X - \mathbb{E}[X])\,(Y - \mathbb{E}[Y])\right], \tag{3.4}$$

but this information can be retrieved thanks to the *pseudo-variance* [Ollila, 2008, Picinbono, 1996]:

$$\tau_Z \triangleq \mathbb{E}\left[(Z - \mathbb{E}[Z])^2\right] = \sigma_X^2 - \sigma_Y^2 + 2\,i\,\sigma_{XY}. \tag{3.5}$$

All σ_X^2, σ_Y^2 and σ_{XY} can be retrieved from the *variance* (σ_Z^2) and *pseudo-variance* (τ_Z)

$$\sigma_X^2 = \frac{\sigma_Z^2 + \mathrm{Re}(\tau_Z)}{2}, \tag{3.6}$$

$$\sigma_Y^2 = \frac{\sigma_Z^2 - \mathrm{Re}(\tau_Z)}{2} \tag{3.7}$$

$$\sigma_{XY} = \frac{\mathrm{Im}(\tau_Z)}{2}, \tag{3.8}$$

thus, the scalar complex random variable can be expressed as:

$$Z \sim \mathcal{CN}(\mathbb{E}[Z], \sigma_Z^2, \tau_Z).$$

For the bivariate Gaussian variable $\boldsymbol{u}$ to be circular, using the origin as the rotation axis, the mean must be zero, there should be no correlation between the real and imaginary part, and the variances of X and Y should be equal, or, in other words, a circular Normal bivariate random variable is defined as $\boldsymbol{u} = \mathcal{N}(\mathbf{0}, \sigma^2\,\mathbf{I})$, which, as it can be deduced from Equation 3.6, translated to a complex Normal

random variable, of the form $Z \sim \mathcal{CN}(0, \sigma_Z^2, 0)$. Indeed, this means that a Normal complex distribution is circular when $\tau_Z = 0$.

The *circularity quotient* $\varrho_Z \equiv \text{circ}(z) \in \mathbb{C}$ is then defined as [Ollila, 2008]:

$$\varrho_Z = \tau_Z / \sigma_Z^2 \,. \tag{3.9}$$

If Z has a vanishing *pseudo-variance*, $\tau_Z = 0$, or equivalently, $\varrho_Z = 0$, it is said to be second-order circular. Therefore, complex non-circular random datasets are generated and classified with two non-exclusive possible sources of non-circularity: X and Y have unequal variances, or X and Y are correlated [Duvaut, 1994, Chapter 10]

The *circularity quotient* can be expressed in the polar form $\varrho_Z = r_Z e^{i\theta}$, where the *circularity coefficient* is defined as $r_Z = |\varrho_Z| \in [0; 1]$, in other words, the *circularity quotient* lies on the unit disk, and $\theta = \arg \varrho_Z$ is the *circularity angle*. The *circularity angle* for a circular variable does not exist by definition. Indeed, a circular variable would have $\varrho_Z = 0$, for what this property is contemplated in the equation.

A non-circular complex Normal random variable would form an ellipse that can be described using the *circularity quotient* ϱ_Z. The orientation of the ellipse (α) would be given by the *circularity angle* as $\alpha = \arg(\varrho_Z)/2$ [Ollila, 2008]; meanwhile, the closer ϱ_Z gets to the unit circle, the more elongated the ellipse would be. On the contrary, the closer it gets to zero, the more it will look like a circle.

Finally, the *correlation coefficient* can be defined as

$$\rho = \frac{\sigma_{XY}}{\sigma_X \sigma_Y}, \tag{3.10}$$

with $\rho \leq 1$ and $\rho = 1$ when Y is a linear function of X almost surely. Under the assumption that ρ exists, it is possible to relate ρ with ϱ_Z as

$$\rho = \frac{\text{Im}(\varrho_Z)}{\sqrt{1 - \text{Re}^2(\varrho_Z)}}\,, \tag{3.11}$$

$$= \frac{r_Z \sin(\theta)}{\sqrt{1 - r_Z^2 \cos(\theta)^2}}\,, \tag{3.12}$$

as it was proved in Reference [Ollila, 2008]. Some important properties can be obtained from previous equations,

- $\varrho_Z = \pm 1 \in \mathbb{R} \Leftrightarrow Y$ equals to zero ($\varrho_Z = 1$) or X equals to zero ($\varrho_Z = -1$).
- $\rho = \pm 1 \Leftrightarrow X$ is a linear mapping of Y,
- $\varrho_Z = \pm r_Z \Leftrightarrow \rho = 0 \Leftrightarrow \theta = 0$ or $\theta = \pi$ for $0 < r_Z < 1$,
- $\varrho_Z = \pm i r_Z \Leftrightarrow \theta = \pm\pi/2 \Leftrightarrow \sigma_Y^2 = \sigma_X^2 \Rightarrow \rho = \pm r_Z$ for $0 < r_Z < 1$,

- $\rho \neq 0 \Rightarrow \varrho_Z \neq 0$ and therefore Z is not circular.

And most importantly, a complex-valued Gaussian distribution Z is circular $\Leftrightarrow$ $\varrho_Z = 0 \Leftarrow \tau_Z = 0 \Rightarrow \rho = 0$.

In the vector case, we have $\boldsymbol{X}$ and $\boldsymbol{Y}$ vectors of size N with

$$\begin{cases} \boldsymbol{\Gamma}_X = \mathbb{E}\left[(\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])\,(\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])^T\right] \\ \boldsymbol{\Gamma}_Y = \mathbb{E}\left[(\boldsymbol{Y} - \mathbb{E}[\boldsymbol{Y}])\,(\boldsymbol{Y} - \mathbb{E}[\boldsymbol{Y}])^T\right] \\ \boldsymbol{\Gamma}_{XY} = \mathbb{E}\left[(\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])\,(\boldsymbol{Y} - \mathbb{E}[\boldsymbol{Y}])^T\right] \\ \boldsymbol{\Gamma}_{YX} = \boldsymbol{\Gamma}_{XY}^T\,, \end{cases} \tag{3.13}$$

and $\boldsymbol{Z} = \boldsymbol{X} + i\,\boldsymbol{Y}$ so that

$$\begin{cases} \boldsymbol{\Gamma}_Z = \mathbb{E}\left[(\boldsymbol{Z} - \mathbb{E}[\boldsymbol{Z}])\,(\boldsymbol{Z} - \mathbb{E}[\boldsymbol{Z}])^T\right] \\ \boldsymbol{\Sigma}_Z = \boldsymbol{\Gamma}_Z = \mathbb{E}\left[(\boldsymbol{Z} - \mathbb{E}[\boldsymbol{Z}])\,(\boldsymbol{Z} - \mathbb{E}[\boldsymbol{Z}])^H\right], \end{cases} \tag{3.14}$$

where the operator ^{H} stands for complex conjugate transpose operation. With this definitions, $\boldsymbol{\Gamma}_Z = (\boldsymbol{\Gamma}_X + \boldsymbol{\Gamma}_Y) + i\,(\boldsymbol{\Gamma}_{YX} - \boldsymbol{\Gamma}_{XY})$.

In this case, $\boldsymbol{Z} \sim \mathcal{CN}(\mathbb{E}[\boldsymbol{Z}], \boldsymbol{\Sigma}_Z, \boldsymbol{\Gamma}_Z)$. $\boldsymbol{Z}$ would be circular $\Leftrightarrow$

$$\begin{cases} \boldsymbol{\Gamma}_X = \boldsymbol{\Gamma}_Y \\ \boldsymbol{\Gamma}_{XY} = -\boldsymbol{\Gamma}_{YX} = -\boldsymbol{\Gamma}_{XY}^T\,. \end{cases} \tag{3.15}$$

The last equality holds if $\boldsymbol{\Gamma}_{XY}$ is a skew-symmetric matrix. These inequalities translate on the decorrelation of $\boldsymbol{Z}$ and $\overline{\boldsymbol{Z}}$ so that

$$\mathbb{E}\left[\boldsymbol{Z}\overline{\boldsymbol{Z}}^H\right] = \boldsymbol{0}\,, \tag{3.16}$$

Finally, the pdf p_Z of a centered Gaussian circular complex vector would be [Duvaut, 1994]:

$$p_Z(\boldsymbol{Z}) = \frac{1}{\pi^N \det(\boldsymbol{\Gamma}_Z)} \exp\left(-\boldsymbol{Z}^H \boldsymbol{\Sigma}_Z^{-1} \boldsymbol{Z}\right)\,. \tag{3.17}$$

3.2 . Experimental Setup

3.2.1 . Model Architecture

Number of layers

Even though the tendency is to make the models as deep as possible for Convolutional Neural Network (CNN), this is not the case for fully-connected feed-forward neural networks, also known as MultiLayer Perceptron (MLP). For these models, one hidden layer (1HL) is usually sufficient for the vast majority of problems [Hornik et al., 1989, Stinchcombe and White, 1989]. Although some authors may argue

that two hidden layers (2HL) may be better than one [Thomas et al., 2017] until recently, most authors seemed to agree that

> « there is currently no theoretical reason to use a MLP with more than two hidden layers » - [Heaton, 2008, p. 158]

However, recent work rejects this idea for MLP as well [Montúfar et al., 2014]. Nevertheless, for computing speed, we will adhere to only one and two hidden layers.

References [Heaton, 2008] and [Kulkarni and Joshi, 2015] recommend the neurons of the hidden layer to be between the size of the input layer and the output layer. Therefore, two models will be used as default in Section 3.3, one with a single hidden layer of size 64 and one with two hidden layers of shape 100 and 40 for the first and second hidden layers respectively. In order to prevent the models from overfitting, dropout regularization technique [Srivastava et al., 2014] is used on one hidden layer and two hidden layers hidden layers. Both CVNN and RVNN are trained with a dropout factor of 0.5.

Equivalent RVNN

To define an equivalent RVNN, the strategy used in [Hirose, 2012] is adopted, separating the input z into two real values (x, y) where $z = x + i\,y$, giving the network a double amount of inputs. The same is done for the number of neurons in each hidden layer. Although this strategy keeps the same amount of features in hidden layers, it provides a higher capacity for the RVNN with respect to the number of real-valued training parameters [Mönning and Manandhar, 2018].

Loss function and optimizer

Mean square error and Cross-Entropy loss functions are mostly used for RVNN to solve regression and classification problems, respectively. The loss remains the same for CVNN since the training phase still requires the minimization over a real-valued loss function. We currently limit our optimizer to the well-known standard Stochastic Gradient Descent (SGD). The default learning rate used in this work is 0.01 as being *Tensorflow*'s default (v2.1) for its SGD implementation.

Weights initialization

For weights initialization, Glorot uniform (also known as Xavier uniform) [Glorot and Bengio, 2010] is used, and all biases start at zero as those are *Tensorflow*'s current (v2.1) default initialization methods for dense layers. Glorot initialization generates weight values according to the uniform distribution in [Glorot and Bengio, 2010, eq.16] where its boundaries depend on both input and output sizes of the

initialized layer. At this step, the authors were not yet acknowledged the complex-valued initialization technique discussed in Section 2.6, and the initialization for the complex case was done by initializing the real and imaginary parts separately and analogous to a real-valued layer, therefore, CVNN might be penalized by starting with a higher initial loss value before any training. This will only penalize the CVNN results making the conclusions of this Chapter even more meaningful.

3.2.2 . Dataset setup

As mentioned previously, to respect the equivalence between RVNN and CVNN, we use in the following input vectors of size 128 (resp. 256) for CVNNs (resp. RVNN). Each element of the feature vector is independently generated according to a non-circular Complex Normal distribution $\mathcal{CN}(0, \sigma_Z^2, \tau_Z)$. Two sources of non-circularity could occur in practice: $\sigma_X \neq \sigma_Y$ and/or $\rho \neq 0$, or equivalently $\tau_Z \neq 0$. Therefore, we propose to evaluate the classification performance of CVNN and RVNN for three types of datasets presented in Table 3.1.

	Data A		Data B		Data C	
Class	1	2	1	2	1	2
ρ	0.3	-0.3	0	0	0.3	-0.3
σ_X^2	1	1	1	2	1	2
σ_Y^2	1	1	2	1	2	1
ϱ_Z	$0.3\,i$	$-0.3\,i$	$-\frac{1}{3}$	$\frac{1}{3}$	$\frac{-0.6+i}{3}$	$\frac{0.6-i}{3}$

Table 3.1: Dataset characteristics.

Figure 3.1 shows an example of two vector samples from dataset A with a feature size of 128. It is possible to see that most features coincide even if they are from different classes, meaning that several points will yield no information or even confuse the classification algorithm.

It is important to note that the distinction between classes is entirely contained in the relationship between the real and imaginary parts. This means that removing, for example, the imaginary part of the dataset will result in both classes being statistically identical, and therefore, rendering the classification impossible.

To evaluate the difficulty of classifying this dataset, a Maximum Likelihood Estimation of τ_Z was implemented with the *prior* knowledge of the underlying Gaussian distributions used to generate the dataset. The data are then classified using a threshold on the estimate of τ_Z. The accuracy of this classifier gives an upper bound of the optimal accuracy. For a low correlation coefficient, for example, $\rho = 0.1$, this parametric classifier only achieves around 85% accuracy.

3.3 . Experimental Results

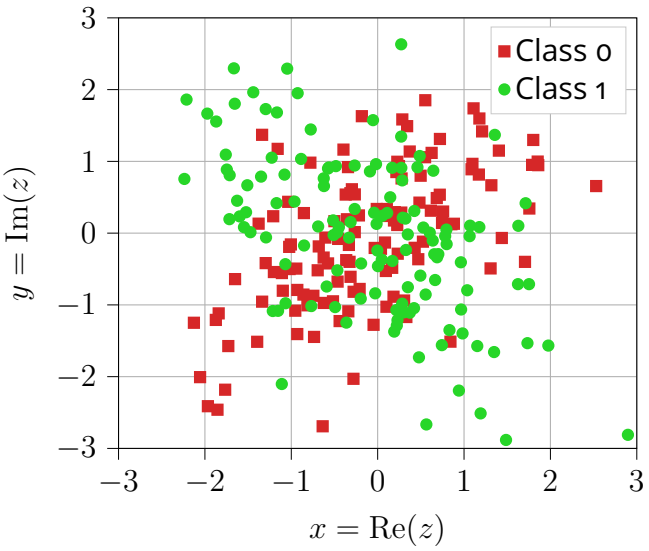


Figure 3.1: Dataset A example.

To ensure that the models do not fall short of data, 10000 samples of each class were generated using 80% for the train set and the remaining 20% for validation. Accuracy and loss of both CVNN and RVNN, defined previously in Section 3.2.1, are statistically evaluated over 1000 Monte-Carlo trials. Each trial contained 150 epochs with a batch size of 100. This number was chosen by observing that after 150 epochs, the accuracy and loss presented almost no amelioration.

3.3.1 . Baseline results

Unless said otherwise, dataset A is used as the default dataset for the results presented in this Section. Only the two hidden layers (2HL) case is illustrated in Figure 3.2 as their results are more favorable to the RVNN model. CVNN loss starts higher but decreases faster than RVNN. Both losses behave well without significant indication of overfitting. Additionally, the validation accuracy of CVNN trials stays above 95%. The validation accuracy of RVNN is lower than the CVNN one and presents more outliers.

The Table 3.2 summarizes the validation accuracy at the last epoch of one hidden layer (1HL) and two hidden layers models for all three different datasets. The median error is computed as $1.57\,\mathrm{IQR}/\sqrt{n}$ [McGill et al., 1978], where IQR is the Inter-Quartile Range, and n is the number of trials. According to [Chambers, 2018], using this error definition, if median values do not overlap, there is a 95% confidence that their values differ.

The results are skewed, meaning there is a significant difference between mean and median accuracy, as the mean is less robust to outliers. For this reason, the median would be a better measure of central tendency in the present simulations. For the complex-valued model, the outliers tend to be the bad cases, whereas, for

		Data A	
		CVNN	RVNN
1HL	median	95.00 ± 0.03	75.58 ± 0.69
	mean	94.98 ± 0.02	76.71 ± 0.27
	IQR	$94.73 - 95.23$	$69.20 - 82.93$
	full range	$93.60 - 96.08$	$64.05 - 93.15$
2HL	median	97.03 ± 0.03	92.90 ± 0.08
	mean	96.98 ± 0.02	92.37 ± 0.07
	IQR	$96.78 - 97.23$	$92.02 - 93.48$
	full range	$95.23 - 98.05$	$68.78 - 94.78$
		Data B	
		CVNN	RVNN
1HL	median	84.38 ± 0.05	58.73 ± 0.07
	mean	84.28 ± 0.03	58.89 ± 0.05
	IQR	$83.83 - 84.83$	$58.10 - 59.48$
	full range	$76.15 - 86.60$	$55.68 - 59.48$
2HL	median	69.90 ± 0.88	59.03 ± 0.33
	mean	69.48 ± 0.32	59.10 ± 0.14
	IQR	$60.89 - 78.43$	$55.80 - 62.35$
	full range	$50.03 - 87.08$	$49.98 - 71.23$
		Data C	
		CVNN	RVNN
1HL	median	96.95 ± 0.02	82.90 ± 0.78
	mean	96.96 ± 0.01	82.76 ± 0.26
	IQR	$96.78 - 97.18$	$75.49 - 91.08$
	full range	$96.00 - 97.90$	$67.35 - 95.88$
2HL	median	98.43 ± 0.02	96.10 ± 0.04
	mean	98.41 ± 0.01	96.02 ± 0.01
	IQR	$98.28 - 98.58$	$95.75 - 96.40$
	full range	$97.23 - 99.03$	$90.38 - 97.18$

Table 3.2: Circularity validation accuracy results (%).

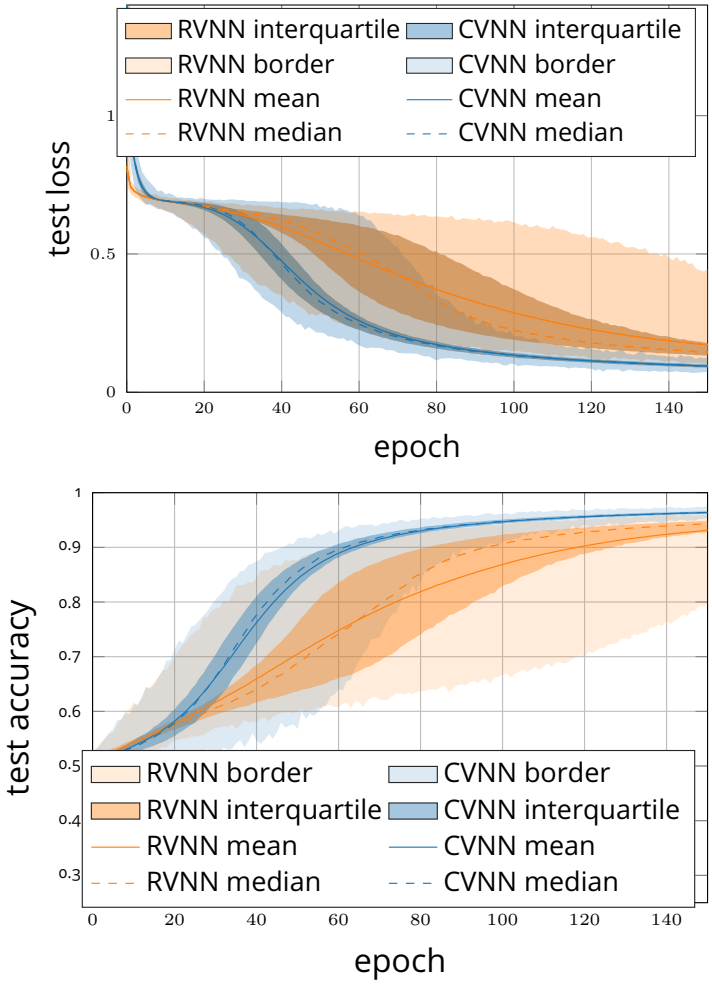


Figure 3.2: Validation loss and accuracy on dataset A for two hidden layers CVNN and RVNN with a dropout of 50%.

the real model, they are the good cases. This can be verified by the mean that is lower than the median for CVNN and higher than the median for RVNN.

For dataset B with two hidden layers, both models fail to achieve good results on average. Despite these poor performances, CVNN still proves to be superior to RVNN by far. For one hidden layer, CVNN achieves a high accuracy with a median of over 84%. Dataset C presents almost the same results as dataset A with some improvements for both architectures.

From these results, the merits of CVNNs are statistically justified by a higher accuracy than RVNNs with less overfitting and smaller variance.

In general, RVNN performed much better with two hidden layers than with one hidden layer. In the Sections that follow, two hidden layers results will be prioritized, bearing in mind that one hidden layer cases were even more favorable to CVNN.

3.3.2 . Case without dropout

The simulations were re-done with no dropout to test the model's tendency to overfit. RVNN presented very high overfitting, as can be seen in Figure 3.3. However, CVNN presented higher variance.

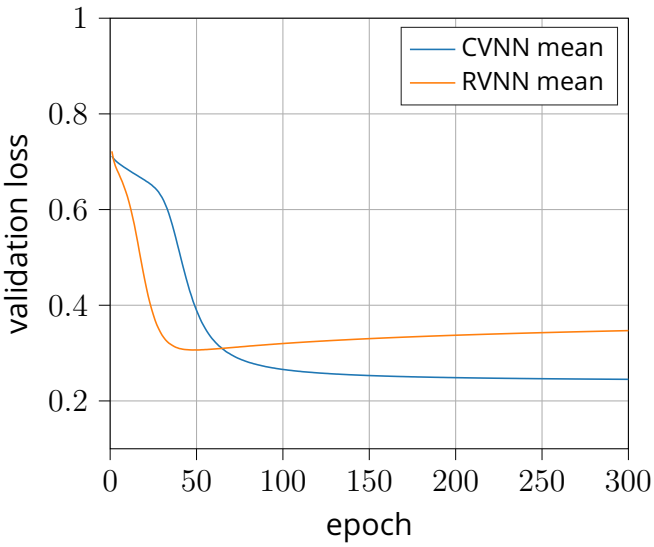


Figure 3.3: Validation loss on dataset A for two hidden layers CVNN and RVNN without dropout.

The impact was so high that the validation accuracy of RVNN drop almost 7% for two hidden layers, whereas for CVNN it dropped less than 4%. The totality of the simulations mentioned in this report was also done without dropout. In general, dropout had less impact on CVNNs performance and a huge amelioration for RVNN. However, it is worth mentioning that although on average CVNN outperformed RVNN when no dropout was used, it presented many outliers with very low accuracy. In conclusion, CVNN has less tendency to over-fit, whereas for RVNN is decisive to use dropout or at least a regularization technique.

3.3.3 . Phase and amplitude

Since the phase information could be relevant for classifying these datasets, polar-RVNN is defined where the inputs are the amplitude and phase of data. This method is tested for datasets A and B with and without dropout.

Figure 3.4 shows the results for two hidden layers tested on dataset A with dropout. It can be seen that polar-RVNN highly improves compared to the conventional RVNN, showing higher mean accuracy but also much less variance, even lower than for CVNN. However, CVNN still outperforms both real models by a wide margin.

This higher performance of polar-RVNN against RVNN can be explained by the fact that dataset type A presents more relevant information in the phase. However,

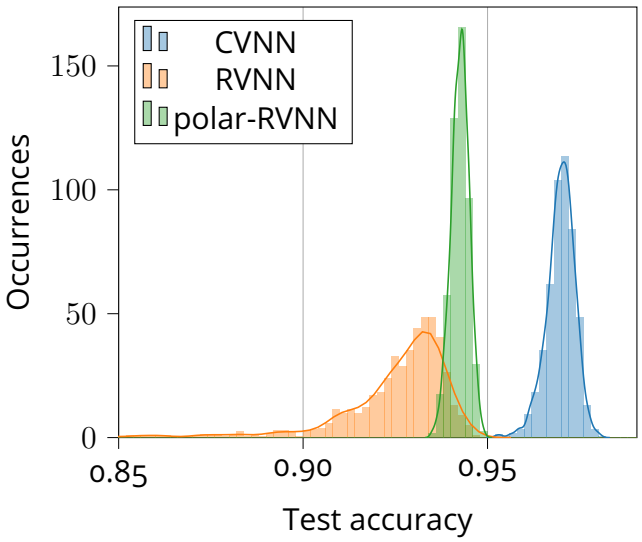


Figure 3.4: Validation accuracy histogram on dataset A of two hidden layers CVNN, polar-RVNN and RVNN.

the opposite happens with dataset type B, for which case, polar-RVNN completely fails to converge and achieves worst results than conventional RVNN for both one hidden layer, and two hidden layers models, the reason why results were omitted.

3.3.4 . Parameter sensibility study

In this Section, a swipe through several model architectures and hyper-parameters is done to assert that the results obtained are independent of specific parameters. These simulations are done for both one hidden layer and two hidden layers networks.

Other sensibility studies were done but are not presented in this work. They concern learning rate, activation function, dataset size, feature vector size, and multi-classes for all combinations of one hidden layer and two hidden layers with and without dropout and for all three types of datasets. In total, this works sum up almost 100 different combinations of experiment parameters.

Correlation coefficient

Changing the correlation coefficient of Data A has been tested for one hidden layer and two hidden layers models. Figure 3.5 shows the accuracy of two hidden layers models, in which the correlation coefficient varies from 0.1 to 0.8. As expected, for small ρ, both networks fail to distinguish between classes with accuracy values barely above 50%. Note that both models cannot possibly achieve more than 85% for a value of $\rho = 0.1$, as was explained in Section 3.2.2. As $|\rho|$ rises, CVNN merits become evident. When $|\rho|$ is close to one, then the link between real and imaginary parts is strengthened, which facilitates the classification of the data for

both models. Results for one hidden layer are even more favorable for CVNN.

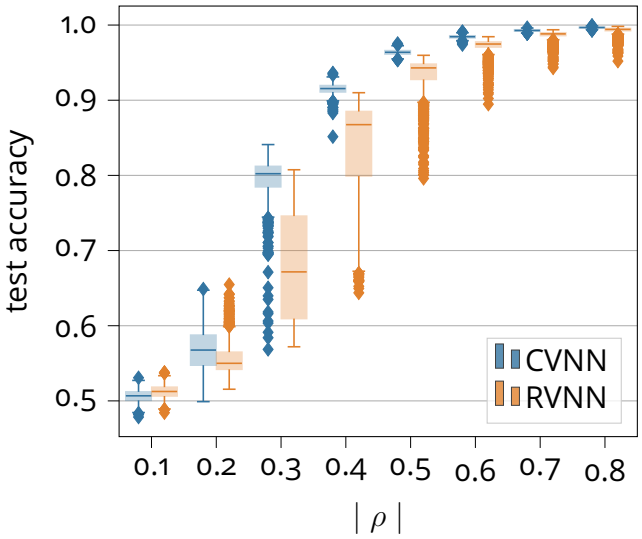


Figure 3.5: Validation accuracy box plot for different values of correlation coefficient ρ for one hidden layer model with dropout.

Hidden layer size

We evaluated the accuracy of one hidden layer for 4 sizes of the hidden layer. All these models were trained on dataset A. The median accuracy of CVNNs was always higher than the one of RVNN, no matter the number of hidden neurons. However, CVNN had low accuracy outliers for sizes 16 and 32, whereas RVNN did not. This could be explained by RVNN having higher capacity, as explained in Section 3.2.1. Fortunately, this behavior disappears when the hidden size is well-dimensioned.

3.4 . Conclusion

In this Chapter, we showed that CVNNs stand as attractive networks to obtain higher performances than conventional RVNNs on complex-valued datasets. The latter point was illustrated by several examples of non-circular complex-valued data, which cover a large amount of data types that can be encountered in signal processing and radar fields.

More than 100 simulations were done, changing and trying a different set of parameters, each one having 1000 trials of both CVNN and RVNN models. All these results have proven that CVNN out-performs RVNN for all sources of Gaussian non-circularity, which can be the correlation between real and imaginary parts (data A), unequal variances between the real and imaginary parts (data

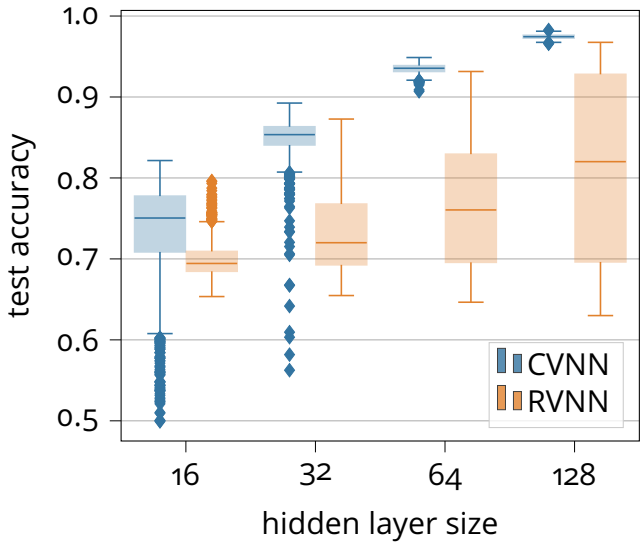


Figure 3.6: Validation accuracy box plot for different one hidden layer size with dropout.

B), or the combination of both (data C). All statistical indicators showed that CVNN clearly outperforms RVNN, presenting higher accuracy, faster convergence, smaller variance, and less overfitting, regardless of the model architecture and hyper-parameters. The larger overfitting obtained with RVNN can be partially explained by the fact that this model has higher capacity [Mönning and Manandhar, 2018, Barrachina et al., 2021c, Barrachina et al., 2022d].

Conversely, the few cases where RVNN competes with CVNN occurred when the dataset is small or the correlation coefficient $|\rho|$ is close to zero, rendering the discrimination from feature vectors nearly impossible. For these exceptions, neither CVNN and RVNN were actually of any practical use.

4 - Theory of Synthetic Aperture Radar

Synthetic Aperture Radar (SAR) is a remote sensing technique that uses ElectroMagnetic (EM) waves to generate a two-dimensional high-spatial-resolution image of the sensed Earth's surface in almost all weather conditions because

- It uses microwaves that can penetrate through clouds, and possibly foliage and surface soil
- and not being a passive method but using its own waves to illuminate the target, making it possible to take images during both day and night.

4.1 . History of SAR

Radar has been around for over one hundred years since the invention of the Telemobile, a device that used a bell and a receiver to detect remote metallic objects in the darkness, fog, and rain. In World War II, radar was essential for detecting aircraft and missiles.

Carl Wiley (1918 – 1985), Figure 4.1a, was an eccentric engineer that had been one of the first scientists to discuss solar sailing for space travel around the 1940s but wisely published under a pseudonym to avoid critics from his peers. In 1950, he came to Goodyear Aircraft Company (which later became Lockheed Martin) to begin a new scientific project. In 1951, Carly Wiley, under the motivation to drive a cruise missile (MASE) using an on-board radar, discovered the principle of SAR imaging by investigating the Doppler spectrum of the received echos when using a moving radar along a straight direction which he called Simultaneous Buildup Doppler. He was issued a patent in 1954 (Figure 4.1b), generally recognized as the SAR patent [Wiley, 1954]. Later, in 1955, the first SAR system called DOUSER flew on a C-47 making history's first SAR image with a resolution of around 150 meters [Sherwin et al., 1962].

Lockheed Martin continued to lead SAR development for decades to come. Prior to 1960, Goodyear later began the program AN/APS-73, called *Quick Check*. This program successfully proved that SAR images could be displayed live on an aircraft, allowing pilots to establish their location at all times in most weather conditions. This breakthrough

« *Created an amazing amount of excitement for the future technology of all airplanes* »

- Stephen Lasswell.

The SR-71, known as *Blackbird*, a secret supersonic spy plane, was given the go-ahead in 1960 and took four years to build. The SAR system on board the plane had a resolution of 10 meters. The classified SAR systems onboard the Blackbird would be refined and improved over the 29-years life of the aircraft.

(a) Carl A. Wiley, the inventor of SAR radar

(b) Patent of SAR [Wiley, 1954]

Figure 4.1: SAR invention. Extracted from [Lasswell, 2005].

In the early 1970s, the first earth resources radar for geo-mapping was developed, called the Goodyear Earth Mapping System (GEMS). GEMS mapped countries worldwide, particularly close to the equator, where the cloud belt and extreme weather conditions made it impossible to map with optical images.

In 1978, the National Aeronautics and Space Administration (NASA), together with the Jet Propulsion Laboratory (JPL), launched SEASAT, the first on-board satellite SAR system with a mission duration of 110 days lifetime dedicated to remote sensing of oceans and sea ice with wide ground swath [Kramer et al., 2002]. The SEASAT SAR operated at L-band (23.5 cm wavelength) with a single polarization channel HH (horizontal transmit and receive). This mission demonstrated SAR capability in general terrain discrimination and target detection leading to many follow-up space-borne SAR missions during the 80s and the 90s, such as NASAs SIR-A (1981) and SIR-B (1984), the European ERS-1 (1992) and ERS-2 (1995), the Japanese JERS-1 (1992) or the Canadian RADARSAT-1 (1995).

4.1.1 . Polarimetric

To navigate in the absence of sunshine, the Vikings used crystals to examine the *polarization* of skylights during foggy circumstances. This discovery of the phenomenon of polarized electromagnetic radiation dates back to around AD 1000. However, it was not until 1669 that Erasmus Bartolinus (1625 – 1698) produced the first quantitative study on light observation. He was succeeded by Christiaan Huygens (1629 - 1695), who made the most fundamental contributions to the subject of optics by arguing that light is a wave and by finding polarized light (1677).

Étienne-Louis Malus (1775 - 1812) established Newton's hypotheses that *polarization* is a fundamental characteristic of light (1808). An electromagnetic wave such as light consists of a coupled oscillating electric field and magnetic field, which are always perpendicular to each other; the *polarization* of electromagnetic waves refers to the direction of the electric field. In the interaction of electromagnetic waves with material objects and the propagation medium, the complex direction of the electric field vector (*polarization*) is crucial [Lee and Pottier, 2009]. This polarization transformation behavior characterisation is known as *polarimetry* in radar and SAR. Polarimetry SAR deals with the full vector nature of polarized electromagnetic waves.

From the 1940s until the 1960s, early polarimetric radar imaging research concentrated on employing polarized radar echoes to describe aircraft targets. Substantial contributions on the topic were made by Huynen, Sinclair, and Kennaugh [Lee and Pottier, 2009]. Later, Ulaby and Fung showed the significance of polarimetry in the estimate of geophysical parameters, while Valenzuela, Plant, and Alpers illustrated the utility of multiple polarization SAR and scatterometers in their investigations of ocean wave and current remote sensing. At the vanguard of radar polarimetry research, Boerner improved the target decomposition work of Kennaugh and Huynen and presented numerous polarization descriptors, including polarization ratios. Boerner has been crucial, persistently promoting polarimetric radar imaging around the world.

In 1985, JPL successfully implemented the first Polarimetric Synthetic Aperture Radar (PolSAR) AIRSAR at L-Band (1.225 GHz). Creating backscattering power and relative polarimetric phases of any polarization state combinations using quad-polarizations (HH, HV, VH, VV). Later on, NASA-JPL built a PolSAR system that flew on AIRSAR, which used three frequencies, P-, L-, and C-Band on the same pass. Most open-source PolSAR images used for this work come from the AIRSAR campaign that lasted 20 years.

Space-borne PolSAR era started in 1994 when the SIR-C/X-SAR was successfully launched. SIR-C acquired at C-band (5.4 cm in wavelength), L-band (23.5 cm) and single polarization X-band SAR simultaneously.

In the mid-2000, JPL suffered a steady decline in PolSAR-related research and AIRSAR stopped its operations. This decline was compensated by Europe's increasing interest in PolSAR applications since the 1990s, mainly through the European Space Agency (ESA). Indeed, we can see from Figure 4.3 that PolSAR missions are on the rise on all frequency bands.

Indeed, ONERA, the French Aerospace Research Agency, quickly invested, in 1980, a powerful radar airborne platform called RAMSES. This device was known for covering a large electromagnetic spectrum from P- to W-band for a total of eight bands, six of which were fully polarimetric. In 2008, RAMSES reached the end of its life; however, previewing this, ONERA acquired a new airborne remote sensing system called SETHI, which performed its first test campaign in

(a) RAMSES onboard Transall C160

(b) SETHI onboard Falcon 20

Figure 4.2: ONERAs airborne sensors extracted from [Baqué et al., 2019].

2007 [Dreuillet et al., 2006]. SETHI was developed to be a very flexible and multi-function research airborne platform. SETHI made an effort to reduce the cabin space and re-designing the pod due to the change in strategy for the antennas implementation because of the inability to open the door in flight [Baqué et al., 2019].

The first fully PolSAR satellite ALOS was launched in 2006 by the Japanese enterprise JAXA. BIOMASS would be the first P-band space-borne mission [Taillade, 2020].

> *« We have arrived at the door-step of the golden age of polarimetric radar imaging »* - [Lee and Pottier, 2009]

A more detailed list of recent PolSAR airborne and space-borne systems can be found in Section 1.3 of Reference [Lee and Pottier, 2009].

4.2 . SAR background

Synthetic Aperture Radar (SAR) is an active radar system used for acquiring a two-dimensional high spatial resolution image of the desired terrain in most weather conditions.

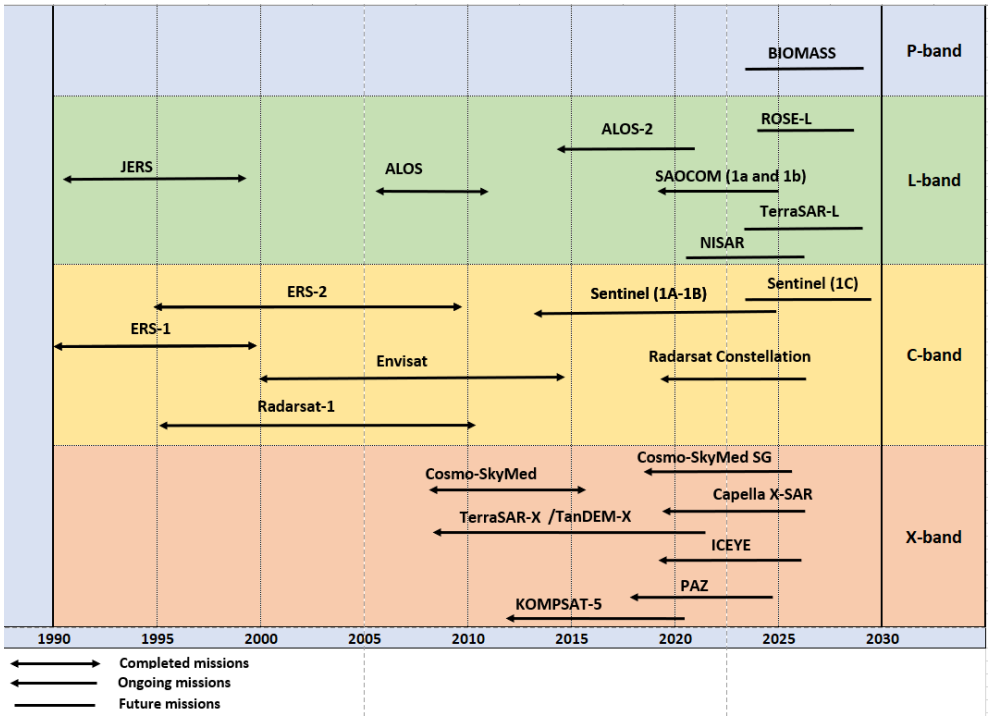


Figure 4.3: Past, current and future SAR missions extracted from [Taillade, 2020].

4.2.1 . Frequency Bands

The terrain is illuminated with coherent ElectroMagnetic (EM) microwave pulses. Such an active operating mode makes this kind of sensor independent of solar illumination and thus allows day and night imaging. The microwave operates in different spectral regions, usually between P-band and Ka-band, depending on the desired properties of the image. For example, below the S-band (see Figure 4.4), one can avoid the effects of clouds, fog, rain, and other weather characteristics, whereas S-, C-, and X-band are also used for cloud, and precipitation imaging [Lee and Pottier, 2009].

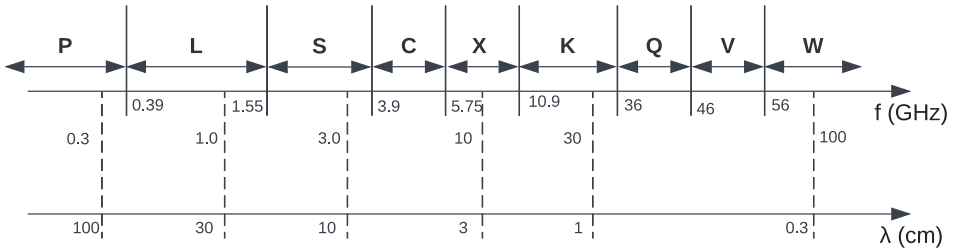


Figure 4.4: Pertinent microwave section of the EM spectrum.

4.2.2 . Geometry

SAR is usually mounted on a moving platform such as an airplane, space shuttle, or satellite that moves in the y direction (Figure 4.5), also known as the *azimuth* direction. An antenna is oriented perpendicular to the flight direction towards the ground at an angle of incidence θ_0.

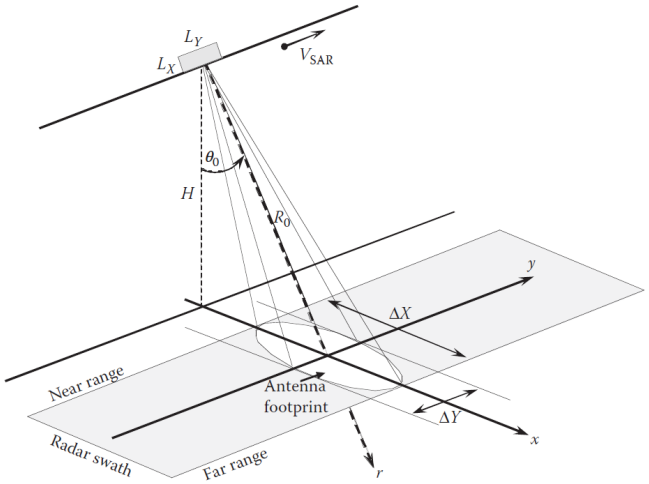


Figure 4.5: SAR imaging geometry in strip-map mode extracted from [Lee and Pottier, 2009].

The radial axis or radar-line-of-sight (RLOS) is known as the *slant range*. The area covered by the antenna beam is called the *antenna footprint*, which is defined from the antenna apertures θ_X, θ_Y given by:

$$\theta_X = \frac{\lambda}{L_X}, \text{ and } \theta_Y = \frac{\lambda}{L_Y}, \tag{4.1}$$

where L_X and L_Y correspond to the physical dimensions of the antenna and λ is the wavelength corresponding to the carrier frequency of the transmitted signal. The *antenna footprint* can then be defined using the range swath (ΔX) and azimuth swath (ΔY) which are defined as

$$\Delta X \approx \frac{R_0 \theta_X}{\cos\theta_0}, \text{ and } \Delta Y \approx \frac{R_0 \theta_Y}{\cos\theta_0}, \tag{4.2}$$

where R_0 is the distance between the radar and the *antenna footprint* center (Figure 4.6). The area scanned by the *antenna footprint* is named *radar swath*.

4.2.3 . Spatial Resolution

One of the most essential characteristics of the SAR image is the spatial resolution, which indicates the ability to distinguish between two closely spaced scatters. On the range distance, this is achieved by using the lower pulse duration possible [Lee and Pottier, 2009]. However, short pulses normally have lower energy,

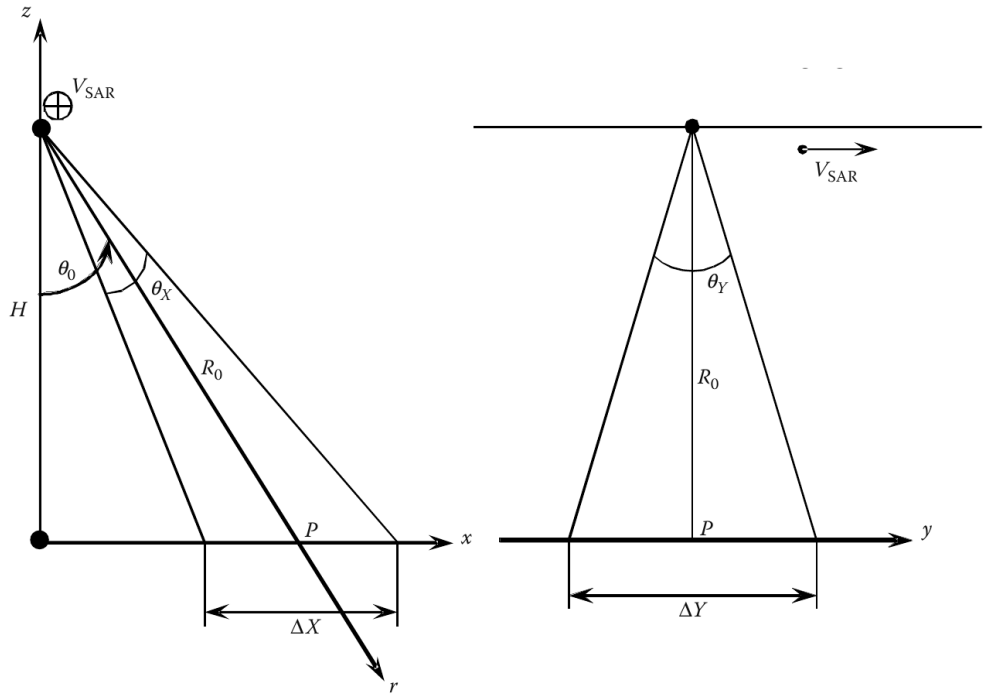


Figure 4.6: Broadside geometry domain.

causing a weak signal-to-noise ratio (SNR). The equipment needed to generate such short, high-energy pulses is challenging to achieve. It is possible to achieve a resolution comparable to those short pulses by using the *pulse compression* method, which sweeps the frequency linearly through a band B, known as *chirp*. The received signal is then processed with a matched filter that compresses the long pulse into a short one with an effective duration equal to $1/B$.

The range resolution is then defined as:

$$\delta x = \frac{c}{2B}\frac{1}{\sin\theta_0}\,, \tag{4.3}$$

where c is the speed of light, note that the range resolution also depends on θ_0, which means that the ground range resolution varies non-linearly.

Normally on the azimuth axe, two scatters can be differentiated if their distance is larger than the beam width. Therefore, the azimuth resolution would be:

$$\delta y = \frac{R_0\lambda}{L_y}\,, \tag{4.4}$$

for what large antennas will be needed to increase the azimuth resolution. The solution behind *synthetic aperture* is based on the fabrication of a longer effective antenna by shifting the antenna along the flight direction, providing a way to obtain high resolution without using a large antenna. When a scattered signal is

coherently integrated throughout the flight path at a given range R0, the azimuth resolution is equal to

$$\delta y = \frac{L_y}{2} \,. \tag{4.5}$$

It's noteworthy that range and wavelength have no bearing on azimuth resolution, which is solely controlled by the synthetic antenna size of the radar system.

For an orbital SAR imaging system, when the platform altitude (H) becomes comparable with the earth radius (R_E), these variables must be taken into consideration in the azimuth resolution expression (Equation 4.5) as follows

$$\delta y = \frac{R_E}{R_E + H} \frac{L_y}{2} \,. \tag{4.6}$$

Because the cross-talk dimension in SAR imaging is determined by a time measurement associated with the distance r (*slant range*) from the radar to the surface, SAR imaging presents an inherent resolution difference as the horizontal (x) distance (or *ground range*) is different from the *slant range* distance.

4.2.4 . Speckle Noise

One of the main drawbacks that appear when working with SAR images is the phenomenon known as *Speckle Noise*. This signal-dependent noise is generated by the coherent nature of the reflected waves from many elementary scatters and its interaction with the roughness of the terrain, causing a pixel-to-pixel variation in intensities manifesting as a granular noise pattern [Goodman, 1976]. The presence of this type of noise decreases the usefulness of these images for both human interpretation and automatic interpretation, such as segmentation or classification algorithms.

The statistical model for speckle noise often starts with the presumption that the resolution cell contains a significant amount of scattered radiations with a wavelength that is comparable to the degree of terrain roughness [Goodman, 1976]. Indeed, Reference [Dalsasso et al., 2022a, Goodman, 1976] describes the speckle noise as a physical phenomenon that is caused by the coherent sum of the contributions from different elementary scatterers within the same resolution cell, which the radar cannot resolve. The phase differences induce fluctuations in the complex summation and then in the observed amplitude, which produces in the observed image a granular behavior.

The received signal can then be represented by the addition of N independent scatters as

$$V = \sum_{k=1}^{N} V_k \exp\left(i\phi_k\right) = V_e \exp\left(i\phi\right) = V_X + i\, V_Y \,. \tag{4.7}$$

When N is sufficiently large, V_X and V_Y will follow a Gaussian distribution due to the Central Limit Theorem. It can be proven that they will have zero means and will be uncorrelated [Mascarenhas, 1997]. It is also possible to conclude that V_e

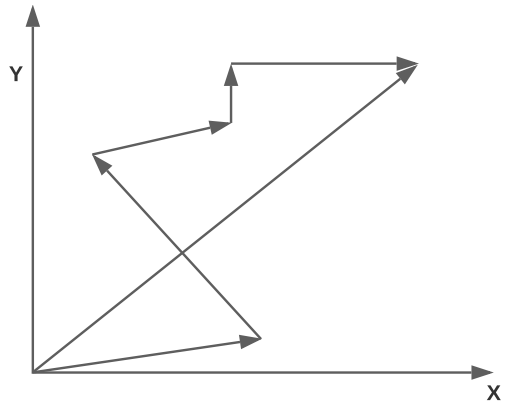


Figure 4.7: Addition of the phasor contributions of different scatters.

will have a Rayleigh distribution and ϕ will have a uniform distribution between 0 and 2π.

The most common technique to mitigate the speckle noise is to perform an averaging filter. This filter replaces a center pixel with the average of itself and all the neighboring pixels inside the boxcar filter of size 3×3 or larger. This method is very effective in speckle noise reduction for homogeneous areas, it is also simple to apply and preserves the mean value [Lee and Pottier, 2009]. However, this method reduces the spatial resolution due to the averaging of pixels from heterogeneous media. Other filter-based techniques are Lee filter [Lee, 1981], Frost filter [Frost et al., 1982], Kuan filter [Kuan et al., 1987], scattering-based-model filter [Lee et al., 2006], or Maximum a posteriori (MAP) estimation techniques [Walessa and Datcu, 2000, Lopes et al., 1993]. In particular, Reference [Walessa and Datcu, 2000] proposed a Gauss Markov random field (GMRF) model for textured areas and allows an adaptive neighborhood system for edge preservation between uniform areas.

More sophisticated techniques have been developed based on Non-Local techniques [Deledalle et al., 2009, Deledalle et al., 2011, Deledalle et al., 2010, Parrilli et al., 2012] attempted to exploit self-similarities and contextual information, generalizing on NL-SAR [Deledalle et al., 2015], a fully automatic algorithm that handles any SAR single- or multi-look images, by performing several Non-Local estimations to restore speckle-free data. Wavelet-based methods [Xie et al., 2002, Argenti and Alparone, 2002] enabled multi-resolution analysis. Later a combination of the Non-Local approach and wavelet domain shrinkage, such as SAR-block-matching 3D (SAR-BM3D) [Parrilli et al., 2012], or SAR-oriented version of BM3D [Dabov et al., 2007] emerged.

A general speckle reduction technique called MuLoG (MUlti-channel LOgarithm with Gaussian denoising) is proposed by Reference [Deledalle et al., 2017] to include Gaussian denoisers originally designed for additive Gaussian noise within an iterative speckle removal procedure for a multi-channel SAR.

Deep learning techniques make use of pairs of speckled / speckle-free images to train on ground truth, leading to results of unprecedented quality in the field of image restoration [Wang et al., 2017, Zhang et al., 2018, Lattari et al., 2019, Rasti et al., 2022]. However, Speckle-free images do not exist, for what these methods resort to synthetic data generated from optical images. Reference [Chierchia et al., 2017] propose averaging long time series of SAR images can reduce speckle to obtain such images. However, the acquisition of multi-temporal data can be challenging and is most of the time not available. The speckled versions of these speckle-free images can be obtained by generating synthetic speckle according to Goodman's model [Goodman, 1976]. These simulations generally do not account for the spatial correlations of speckle observed in actual SAR images, leading to a domain shift between network training and the application to SAR images that requires an additional image pre-processing to reduce the correlations. Without adequate pre-processing, networks trained on white noise lead to strong artifacts when applied to correlated speckle [Dalsasso et al., 2022a].

Therefore, self-supervised neural network models become of interest. SAR2SAR despeckling technique [Dalsasso et al., 2021] uses a deep semi-supervised neural network that learns to restore SAR images by only looking at noisy acquisitions by extending the noise2noise [Lehtinen et al., 2018] in order to take into account the peculiarities of SAR data. Another example of self-supervised models is Speckle2void [Molini et al., 2022], which uses a Bayesian despeckling method inspired by Reference [Laine et al., 2019] and Noise2Void algorithm [Krull et al., 2019].

Reference [Dalsasso et al., 2022b] proposes their algorithm MERLIN, which proposes to train using only one SAR image. Using the real part as the input to the network and the imaginary part as the ground truth. These three semi-supervised neural networks are explained and compared in [Dalsasso et al., 2022a].

A good starting point to learn more about desplecking techniques could be [Mascarenhas, 1997] or Chapter 5 of [Lee and Pottier, 2009].

4.3 . PolSAR data representation

Polarimetric Synthetic Aperture Radar (PolSAR) classification algorithms generally make use of signal coherence (or equivalently phase and local phase variance) existing on a single look complex data channel vector $\boldsymbol{S}$ measured from two orthogonal polarimetric transmitted signals on two orthogonal polarimetric received signals. Here we use the horizontal (H) and vertical (V) polarisation, and, as with monostatic radar, the cross channels are equal; the useful received vector is:

$$\boldsymbol{S} = \left(\boldsymbol{S}_{HH}, \sqrt{2}\,\boldsymbol{S}_{HV}, \boldsymbol{S}_{VV}\right)^{T} . \tag{4.8}$$

For each pixel of the PolSAR image, this backscattering vector is usually ex-

pressed in the Pauli basis and reshaped onto one single complex vector $\in \mathbb{C}^3$:

$$\boldsymbol{k} = \frac{1}{\sqrt{2}} \left(\boldsymbol{S}_{HH} + \boldsymbol{S}_{VV}, \boldsymbol{S}_{HH} - \boldsymbol{S}_{VV}, 2\,\boldsymbol{S}_{HV} \right)^T . \tag{4.9}$$

The Hermitian so-called coherency matrix is then formally built according to

$$\boldsymbol{T} = \frac{1}{n} \sum_{j}^{n} \boldsymbol{k}_j \, \boldsymbol{k}_j^H , \tag{4.10}$$

where the operator ^{H} stands for complex conjugate transpose operation and where n is the number of pixels chosen in a boxcar located in each local area of the SAR image.

Since $\boldsymbol{T}$ is Hermitian symmetric, its lower triangle, excluding the diagonal, is usually discarded as it provides no additional information. As the diagonal is real-valued, the data is extended to the complex plane by adding a zero imaginary part which leads to a total of six complex values per pixel, or nine real values for the RVNN architectures. Unless said otherwise, the coherency matrix $\boldsymbol{T}$ will be the input representation for the Machine Learning models.

4.4 . Available datasets and pre-processing

All the datasets used in this Chapter are summarized in Table 4.1, each will be described in detail at the respective moment. All the images obtained were presented in different file formats. PolSARPro [Pottier and Ferro-Famil, 2012] was used to obtain a common file format for all datasets (`.bin` and `.bin.hdr`).

For all datasets (when possible), three different representations were generated to be used if needed, these are the scattering vector $\boldsymbol{S}$, the Pauli vector $\boldsymbol{k}$ and the coherency matrix T, the later is computed at the pixel level with a boxcar of size 3×3. Since for a covariance matrix of size $p \times p$, we need at least $2p$ samples to compute a good estimate of this matrix. This property is also held to compute a coherency matrix which explains the choice of 3×3 for the boxcar size. Some datasets where found directly in the coherency matrix format with the same boxcar filter as explained before. In these cases, retrieving the scattering vector $\boldsymbol{S}$ or the Pauli vector $\boldsymbol{k}$ was not possible.

Deep learning methods require a large number of labeled training examples, which even if there were enough images for training, these are normally not annotated, so they cannot be used for supervised training tasks. Images are normally manually labeled [Xia et al., 2018, Zhang et al., 2017], which is very time-consuming and therefore costs a lot of human resources. Some work is being done for automatic or semi-automatic annotation [Lienou et al., 2010, Huang et al., 2021, Bratasanu et al., 2011, Inisan et al., 2022].

Furthermore, PolSAR images have several parameters that make images very different from each other, causing a low degree of generalization across datasets,

often known as the bias problem [Xia et al., 2018, Torralba and Efros, 2011]. Indeed, this causes extensive overfitting [Marmanis et al., 2016a]. Some methods, such as transfer learning [Marmanis et al., 2016a], can be used to alleviate the problem.

For all these reasons, more than one PolSAR image is rarely used in research works. However, PolSAR images usually have a high resolution that allows the generation of smaller image patches from which the training, validation and test set can be obtained. These smaller image patches are generated using the sliding window operation [Li et al., 2018b]. This method generates smaller image patches by sliding a window through the image with a given stride. The same parameters used in Reference [Cao et al., 2019] were used for the sliding window operation method, generating images of size 12×12 for the MLP and CNN models and 128×128 for the FCNN architecture.

These patches can be used for both segmentation and classification. CNN and MLP model architectures are designed to perform classification tasks, whereas FCNN is designed to perform segmentation task. In the first case, a prediction of the whole image is made, e.g., which digit is in the image (MNIST) or if the picture is of a cat or a dog but may find difficulties if both a cat and a dog are present in the image. Segmentation tasks provide a pixel-wise classification, where the distinction between dog and cat is done per pixel and can, therefore, predict an image where both the cat and dog are present (see Figure 4.8 for a practical example). Segmentation tasks provide an output of the same resolution as the input image, where the class of each pixel is represented, whereas classification models output only one global prediction class to characterize the whole image. Although MLP or CNN are designed for classification tasks, it is possible to use such models for SAR semantic segmentation problems by predicting each pixel separately with MLP or CNN and re-constructing a pixel-wise classification prediction. Note that only the central pixel is being predicted per image patch, so having several classes within the image will not pose a problem in this case. Therefore, we consider pixel-wise classification analogous to semantic segmentation tasks for which both terms will be used interchangeably.

So for the segmentation model (FCNN), label patches were generated in the same way that the input data, by using the sliding window operation and obtaining ground truth maps of $128 \times 128 \times c$ where c is the total number of classes. For the classification models (MLP or CNN), only the central pixel is used as the label (see Figure 4.8 for a practical example).

For our classification models, the input image is smaller than for the segmentation model (about 10 times smaller). These images are of size 12×12, which generates a much larger amount of image patches. For these cases, there is no need to use all the patches as it will greatly increase the computation time while not increasing by much the final accuracy. References [Hänsch and Hellwich, 2009a] and [Hänsch and Hellwich, 2010] used about 2% of the image pixels for training

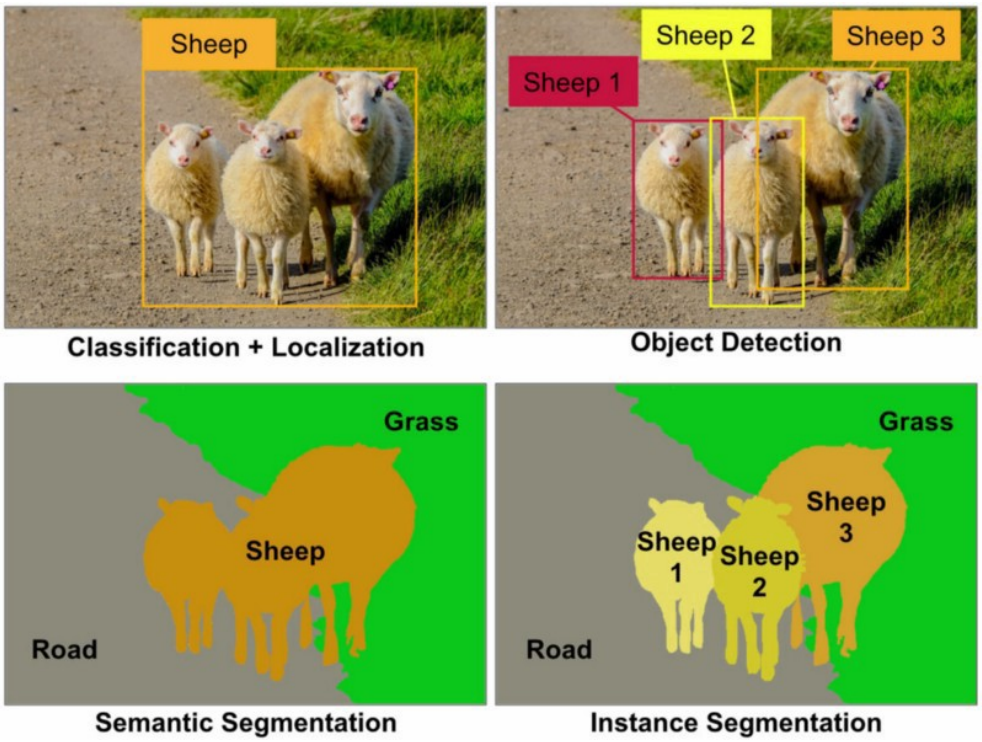


Figure 4.8: Image Classification vs. Semantic Segmentation vs. Instance Segmentation example image.

whereas [Hou et al., 2016] and [Jiao and Liu, 2016] used 5%. In [Guo et al., 2015], the authors adopted 10%. Finally, Reference [Zhang et al., 2017] tested different sampling rates and proposed, based on the results, to use a 10% sampling rate for both training and validation set together. We, therefore, adopt by default 8% for training, 2% for validation and the remaining pixels for testing.

Location	Size	Classes	Labeled pixels	Mission	Frequency	Figure
Oberpfaffenhofen (Ge)	1300x1200	3	1,311,618	E-SAR	L-Band	5.4
Flevoland (Nl)	750x1024	15	157,296	AIRSAR	L-Band	5.8
San Francisco (US)	900x1024	5	802,302	AIRSAR	L-Band	5.12
Bretigny (Fr)	1533x3392	4	2,871,080	RAMSES	X-Band	5.17

Table 4.1: Dataset parameters.

5 - Comparing CVNN against RVNN for PolSAR applications

In the previous Chapter 3, we showed the merits of using CVNN for non-circular data. Knowing that SAR and, more specifically, PolSAR data is known to be non-circular [El-Darymli et al., 2014, Vasile and Totir, 2012], we can know that there is a potential interest in using these networks on PolSAR image classification. Indeed, the common belief that the phase of the SAR image is random and therefore bears no information no longer holds for typical high-resolution targets, previously considered to be points, now provides phase information that can be exploited [El-Darymli et al., 2015]. Indeed, in urban areas, a limited number of human-made scatters with near-regular shapes and sub-meter size leads to correlated phase patterns [Datcu et al., 2007]. Taking into account phase information can boost detection accuracy, and recognition of targets [El-Darymli et al., 2013].

Nevertheless, the research is widely concentrated on using RVNN to tackle this kind of task [Ben Hamida et al., 2018, Marmanis et al., 2018, Marmanis et al., 2016b, Parikh et al., 2020, Konishi et al., 2021, Chen et al., 2016, Hou et al., 2016, Zhou et al., 2016, Fix et al., 2021] as the motivation to use CVNN is not yet clear.

Although some research has been done to classify PolSAR images using CVNN, the comparison to evaluate their gain over an equivalent RVNN has been limited on PolSAR applications. The limitations of current work make it impossible to assert with confidence that classification accuracy does improve when using a CVNN over an equivalent-RVNN. For the existing publications, these limitations are at least one of the following

- No comparison was made against a conventional real-valued neural network.
- CVNN and RVNN not being capacity equivalent as it will be explained in Section 5.1.
- Intersecting or lacking mean or median confidence intervals.

In References [Hänsch and Hellwich, 2010] and [De et al., 2017], the authors tested CV-MLP on a PolSAR database but did not provide a comparison with RV-MLP. In [Hänsch and Hellwich, 2009a], a comparison was performed for both types of networks but did not offer a confidence interval and used a different input representation for each model. Although, in [Hänsch, 2010], the same authors suggested giving the same amount of input representation to get a more precise comparison between the models. Although [Cao et al., 2019] added some neurons to the RV-MLP compared to the CV-MLP which made it an acceptable comparison, they do not specify the criteria they used, and it was not yet enough to make it equivalent to the complex network by neither of the two criteria to be analyzed

in this Section, which are the equal number of *real-valued trainable parameters* (tp) or *real-valued neurons parameters* (np). In [Cao et al., 2019], even though CV-MLP performed better than RV-MLP, confidence intervals intersect, leaving room for doubt about CV-MLP out-performance.

Works using complex Convolutional Neural Network (CNN) have been published for PolSAR applications. Reference [Wilmanski et al., 2016] uses a CV-CNN for Synthetic Aperture Radar (SAR) classification, but the real-valued network is not only under-dimensioned by making both real and complex networks having the same number of filters but also provides the RV-CNN with only the magnitude information. Reference [Oyama and Hirose, 2018] uses a CV-CNN for Interferometric Synthetic Aperture Radar (InSAR) classification but compares it against CV Markov random field (CMRF) model and not a RV-CNN. References [Zhang et al., 2017, Shang et al., 2019] compares a CV-CNN with a real-valued model close to an equivalent definition as explored in the following subsections but lacking confidence intervals. Other recent works [Sun et al., 2019, Zhao et al., 2019a, Zhao et al., 2019b] use a CV-CNN for PolSAR applications but without comparing its result with a real-valued model. In Reference [Cao et al., 2019], both a CV-CNN and CV-FCNN are compared against a real-valued networks that contains the same amount of kernels for each layer making it unfair for the real models as they will have a lower capacity [Mönning and Manandhar, 2018].

In this Chapter, we propose a framework that enables the fair comparison of two deep learning models (one real- and one complex-valued), and we run simulations using that framework for classification and segmentation of PolSAR data.

5.1 . Real Equivalent Network

To assess whether a Complex-Valued Neural Network (CVNN) is actually of interest, it is necessary to compare it with a Real-Valued equivalent network. However, the equivalence between both networks is not straightforward. Indeed, a criterion to create Complex-Valued Neural Network (CVNN) and Real-Valued Neural Network (RVNN) with equivalent capacities remains missing resulting in an unbalanced comparison. The mapping between CVNN and its real equivalent is in general not unique since there are too many degree of freedom in the network architecture, e.g. number of layers and neurons, activation functions, training loss, optimizers, etc. In our work, we only focus on real equivalent networks that have a similar architecture with the CVNN meaning that both networks should have the same number of layers, similar activation functions, training loss and optimizers. Most parameters are naturally transformed into a complex plane. That is true, for example, optimizers, or activation functions (please refer to Section 2). However, if we keep the same amount of neurons (for fully-connected layers) or kernels (for convolutional layers), it will result in the CVNN having higher capacity than their opposed RVNN as we can consider that the complex plane $\mathbb{C}$ isomorphic to $\mathbb{R}^2$

meaning that one complex-valued parameter ($p_{\mathbb{C}}$) is equivalent to two real-valued parameters ($p_{\mathbb{R}}$) so that $p_{\mathbb{C}} = 2\,p_{\mathbb{R}}$. The superscript $\mathbb{C}$ and $\mathbb{R}$ indicate whether it corresponds to the CVNN or RVNN respectively.

In this Section, therefore, we develop a formal definition for capacity equivalent CVNN and RVNN [Barrachina et al., 2021c, Barrachina et al., 2022d] providing a framework under which both networks would be equivalent and therefore its comparison can be made.

5.1.1 . Multilayer Perceptron

To preserve the same amount of *real-valued neuron parameters* (np) per layer on MLP architectures, it will suffice to double the neurons of each hidden layer within the RV-MLP with respect to CV-MLP [Hänsch and Hellwich, 2010, Hirose, 2012, Hirose, 2009]. However, as Reference [Mönning and Manandhar, 2018] points out, this design leads to a bigger number of *real-valued trainable parameters* (tp) for the RVNN. Indeed, ignoring the layer biases that are generally added at the end, a CVNN with two consecutive hidden layers of size 10 each will result in $10 \times 10 = 100$ complex-valued weights for connecting them, which is equivalent to a total of $tp_{\mathbb{C}} \triangleq 200$ *real-valued trainable parameters*. Using the described technique, an equivalent-RVNN will have two consecutive hidden layers of size 20 each, needing a total of $20 \times 20 = 400$ real-valued weights to connect them and, therefore, $tp_{\mathbb{R}} \triangleq 400$. Leading to the latter potentially having a higher capacity if this method is followed.

The global number of tp for a generic CV-MLP and RV-MLP with K hidden layers is provided in [Mönning and Manandhar, 2018] through the formula:

$$\begin{aligned} tp_{\mathbb{C}} &= 2\,N_0^{\mathbb{C}}\,N_1^{\mathbb{C}} + 2\sum_{i=1}^{K-1} N_i^{\mathbb{C}}\,N_{i+1}^{\mathbb{C}} + 2\,N_K^{\mathbb{C}}\,N_L^{\mathbb{C}}\,, \\ tp_{\mathbb{R}} &= N_0^{\mathbb{R}}\,N_1^{\mathbb{R}} + \sum_{i=1}^{K-1} N_i^{\mathbb{R}}\,N_{i+1}^{\mathbb{R}} + N_K^{\mathbb{R}}\,N_L^{\mathbb{R}}\,, \end{aligned} \tag{5.1}$$

where N_i is the number of neurons for layer $i \in 1, ..., K$. N_0 corresponds to the number of features or input size and N_L to the output size.

The task to solve directly determines the input and output sizes of the real network so that $N_0 = N_0^{\mathbb{R}} = 2\,N_0^{\mathbb{C}}$ and

$$N_L = N_L^{\mathbb{R}} = \begin{cases} 2\,N_L^{\mathbb{C}}, & \text{regression task} \\ N_L^{\mathbb{C}}, & \text{classification task} \end{cases}. \tag{5.2}$$

Reference [Mönning and Manandhar, 2018] argues that a real-valued equivalent model must have the same tp capacity as the complex one: $tp_{\mathbb{C}} = tp_{\mathbb{R}} = tp$. To accomplish this, they propose to alternate between doubling or not the number of neurons of the real-valued model hidden layers with respect to the complex-valued model. However, this strategy only works when the number of hidden layers is

even for classification tasks and an odd number for regressions tasks. To address this problem, we propose designating one hidden layer as:

$$N_i^{\mathbb{R}} = 2\,\frac{N_{i-1}^{\mathbb{C}} + N_{i+1}^{\mathbb{C}}}{N_{i-1}^{\mathbb{R}} + N_{i+1}^{\mathbb{R}}}\,N_i^{\mathbb{C}}\,. \tag{5.3}$$

Another proposition in [Mönning and Manandhar, 2018] is to make all layers the same size. Nevertheless, this solution will not maintain the same *aspect ratio* for both CVNN and RVNN models. As exemplified in Figures 5.1a and 5.1c, performing classification with a CV-MLP with two hidden layers of sizes 10 and 5 will be converted to a RV-MLP where both hidden layer sizes are 10. This means converting a network where the first hidden layer doubles the size of the second to one where both hidden layers are the same size.

In this paper, we propose to maintain the same *aspect ratio* for each hidden layer, i.e., the number of hidden layer neurons of RVNN is proportional to the one of RVNN, which leads to the following equation:

$$N_i^{\mathbb{R}} = r\,N_i^{\mathbb{C}},\ \forall i \in 1,\ldots,K\,, \tag{5.4}$$

with r a positive constant real value. Replacing (5.4) in (5.1) we obtain the following second-order polynomial equation in the variable r:

$$tp = \left(\sum_{i=1}^{K-1} N_i^{\mathbb{C}}\,N_{i+1}^{\mathbb{C}}\right) r^2 + \left(N_0\,N_1^{\mathbb{C}} + N_K^{\mathbb{C}}\,N_L\right) r. \tag{5.5}$$

Since r should be positive as well as all parameters tp, $N_i^{\mathbb{R}}$, N_L and N_0, the only possible solution to our problem is therefore:

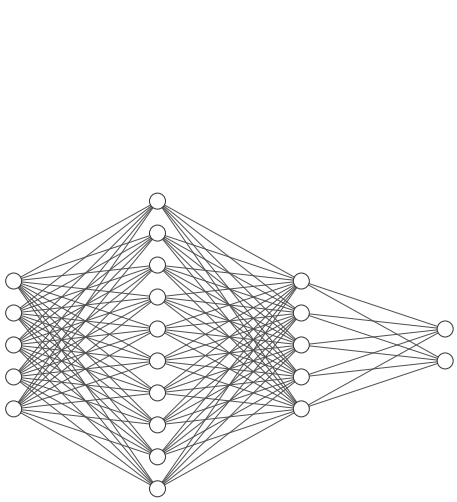
$$r = \frac{-b + \sqrt{b^2 - 4\,a\,(-tp)}}{2\,a}\,, \tag{5.6}$$

where $a = \sum_{i=1}^{K-1} N_i^{\mathbb{C}}\,N_{i+1}^{\mathbb{C}}$, $b = N_0\,N_1^{\mathbb{C}} + N_k^{\mathbb{C}}\,N_L$.

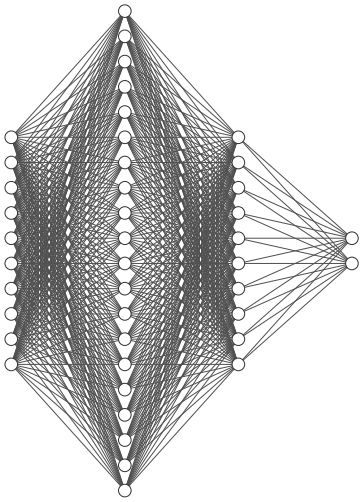
In conclusion, there are two possible definitions for an equivalent-RV-MLP. Either by setting the same *real-valued trainable parameters* (tp) or by its *real-valued neuron parameters* (np) per hidden layer (Figure 5.1b). The former can be done by creating a RV-MLP where each hidden layer size is given by Equation 5.4 with r being defined by (5.6); this will result in an equivalent-RV-MLP in terms of the *real-valued training parameters* that maintain the same aspect ratio that the CVNN hidden layers (Figure 5.1d).

If $0 < r < 1$,

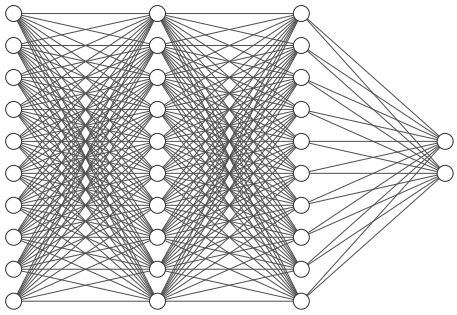
$$\begin{aligned} &a\,r^2 + b\,r - tp = 0 < r\,a + r\,b - tp\,,\\ \Rightarrow\ &0 < r\,a + r\,b - 2\,a - \beta \leq a\,(r-2) + b\,(r-1)\,, \end{aligned} \tag{5.7}$$



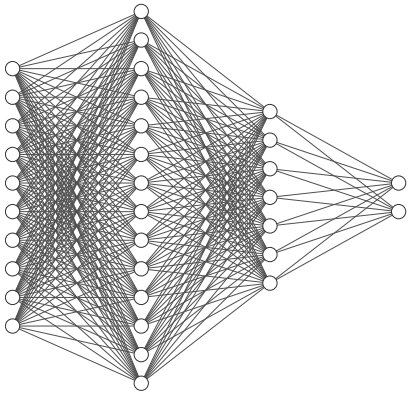
(a) Original complex model



(b) np-equivalent model



(c) tp-equivalent model proposed by [Mönning and Manandhar, 2018]



(d) Proposed tp-equivalent model [Barrachina et al., 2021c]

Figure 5.1: Real equivalent MLP models example. Figures generated using alexlenail.

where $tp = 2\,a + \beta$ with $b \leq \beta = 2\,N_0^{\mathbb{C}}\,N_1^{\mathbb{C}} + 2\,N_k^{\mathbb{C}}\,N_L^{\mathbb{C}} < 2\,b$ (Equation 5.1). As both a and b are positive, Equation 5.7 is absurd, which is expected as it implies that real-valued models will never have fewer neurons than the complex-valued models. On the other hand, for $r \geq 1$:

$$\begin{aligned} a\,r^2 + b\,r - tp = 0 &\geq r\,a + r\,b - tp\,, \\ \Rightarrow 0 \geq a\,(r-2) + b\,r - \beta &> a\,(r-2) + b\,(r-2)\,. \end{aligned} \tag{5.8}$$

Again, as a and b are positive, Equation 5.8 is absurd if $r \geq 2$. Because of inequalities (5.7) and (5.8), we conclude that $1 \leq r < 2$, meaning that the equivalent-RVNN should have at least the same dimension as CVNN and at most double. In particular, $r = 2$ corresponds to the case for the same value of np. Proving that

it is not possible to reach both conditions simultaneously and one must choose between setting an equal value for np or tp.

For single hidden layer models, $a = 0$ and therefore, r will be:

$$r = \frac{\beta}{b} = 2\,\frac{N_0^{\mathbb{C}} + N_L^{\mathbb{C}}}{N_0 + N_L}\,. \tag{5.9}$$

As it can be derived from (5.9), $r = 1$ for regressions tasks while for classifications tasks, $1 < r < 2$ depending on the relationship between N_0 and N_L. Finally, as the number of hidden neurons gets bigger with respect to the input and output, or in other words, $a \gg b$, it will tend $r \to \sqrt{2}$.

Note that, the extra terms $2\sum_{i=1}^{K} N_i^{\mathbb{C}}$ and $\sum_{i=1}^{K} N_i^{\mathbb{R}}$ should be added to Equation (5.1) in order to take into account the bias. This extra term will lead to a slight variation of r by changing the value of b but does not change its boundary $1 \leq r < 2$. Naturally, when multiplying r by $N_i^{\mathbb{C}}$ to obtain the size of each real-valued layer (5.4), it will most likely not yield an integer. In these cases, the value was rounded up to make sure the real-equivalent network never has lower capacity than their complex-valued counterpart.

5.1.2 . Convolutional Neural Networks

For convolutional layers, the equation of the *real-valued trainable parameters* is defined by:

$$\begin{aligned} tp_{\mathbb{C}} &= 2\sum_{i=1}^{K} C_i^{\mathbb{C}}\, W_i^{\mathbb{C}} H_i^{\mathbb{C}} F_i^{\mathbb{C}}\,, \\ tp_{\mathbb{R}} &= \sum_{i=1}^{K} C_i^{\mathbb{R}}\, W_i^{\mathbb{R}} H_i^{\mathbb{R}} F_i^{\mathbb{R}}\,, \end{aligned} \tag{5.10}$$

with $\{C_i\}^{\mathbb{C},\mathbb{R}}$ the channels presented on the input of the convolutional layer, $\{W_i\}^{\mathbb{C},\mathbb{R}}$ and $\{H_i\}^{\mathbb{C},\mathbb{R}}$ the filter width and height respectively and $\{F_i\}^{\mathbb{C},\mathbb{R}}$ the amount of filters or kernels of the layer i.

For convolutional layers, there are a few options on how to maintain the same amount of *real-valued trainable parameters*, either by extending the kernel sizes or increasing the number of kernels. The second method may seem more logical as the transformation from the complex to the real plane does not change the resolution of the image and we maintain the same size of the receptive field for both networks. By adopting the second method then $H_i^{\mathbb{C}} = H_i^{\mathbb{R}}$ and $W_i^{\mathbb{C}} = W_i^{\mathbb{R}}$. By definition, $C_i = F_{i-1}$ with F_0 the input image channel dimension. Without further restrictions, the solution would become evident with $r = \sqrt{2}$ as shown in Figure 5.2. However, as before, the mapping of the complex data into the real domain will result in the real-valued input having twice as channels as the complex case so that $F_0^{\mathbb{R}} = 2F_0^{\mathbb{C}}$. In order to have the same *real-valued trainable parameters*, we need to solve Equation 5.11 for $r \in \mathbb{R}$ in order to have $F_i^{\mathbb{R}} = r\,F_i^{\mathbb{C}}$,

$\forall i \geq 1$.

$$tp = 2 \sum_{i=1}^{K} F_{i-1}^{\mathbb{C}} W_i H_i F_i^{\mathbb{C}} = 2r F_0^{\mathbb{C}} W_1 H_1 F_1^{\mathbb{C}} + r^2 \sum_{i=2}^{K} F_{i-1}^{\mathbb{C}} W_i H_i F_i^{\mathbb{C}} . \quad (5.11)$$

From the above equation, the solution for r is:

$$r = \frac{-b}{2\,a} + \sqrt{2 + \frac{b}{a} + \frac{b^2}{4\,a^2}}\,, \quad (5.12)$$

with $b = 2F_0^{\mathbb{C}} W_1 H_1 F_1^{\mathbb{C}}$ and $a = \sum_{i=2}^{K} F_{i-1}^{\mathbb{C}} W_i H_i F_i^{\mathbb{C}}$. The extreme case of a single convolutional layer makes $r = 1$. Similarly as for MLP, when $a >> b$, that happens for example with deep the convolutional neural networks, $r \to \sqrt{2}$.

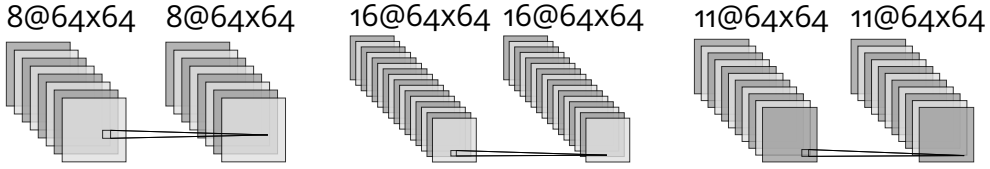
If taking into account the bias, Equation 5.10 changes to:

$$\begin{aligned} tp_{\mathbb{C}} &= 2 \sum_{i=1}^{K} \left(C_i^{\mathbb{C}} W_i^{\mathbb{C}} H_i^{\mathbb{C}} + 1 \right) F_i^{\mathbb{C}}, \\ tp_{\mathbb{R}} &= \sum_{i=1}^{K} \left(C_i^{\mathbb{R}} W_i^{\mathbb{R}} H_i^{\mathbb{R}} + 1 \right) F_i^{\mathbb{R}} . \end{aligned} \quad (5.13)$$

With this changes, Equation 5.12 changes to

$$r = \frac{-b}{2\,a} + \sqrt{2 + \frac{b}{a} + \frac{b^2}{4\,a^2} + \frac{\sum_{i=1} F_i}{a}}\,, \quad (5.14)$$

where a is as before but b changes to $b = 2F_0^{\mathbb{C}} W_1 H_1 F_1^{\mathbb{C}} + \sum_{i=1} F_i$.



(a) Original complex model (b) np-equivalent model (c) tp-equivalent model

Figure 5.2: Real equivalent convolution example of a middle hidden layer. Figures generated using alexlenail.

5.2 . Neural Network architectures used for the experiments

MLP [Hänsch and Hellwich, 2009a], CNN [Zhang et al., 2017] and FCNN [Cao et al., 2019] model architectures are used throughout this Chapter, both on the

complex and real domain respecting the equivalence definitions discussed on Section 5.1. In this Section, we will give a detailed description of those models. Some small modifications were made compared to the model's respective References with *state-of-the-art* parameters not popular or known at the time of those publications. References [Hänsch and Hellwich, 2009a] and [Zhang et al., 2017] use Stochastic Gradient Descent (SGD) as an optimizer whereas Reference [Cao et al., 2019] use a more modern optimizer known as Adam [Kingma and Ba, 2014] which might allow models to find a lower optimal minimum. Adam was, therefore, used as the optimizer for all models. Learning rate and momentum were tweaked for each model independently according to the results. As well as the number of epochs.

Although [Hänsch and Hellwich, 2009a] use $\tanh$ activation function for the MLP model, we decided in this work to use Rectified Linear Unit (ReLU). Indeed, both activation functions were tested for the MLP architecture showing an interesting increase in performance when using Rectified Linear Unit (ReLU). This small optimization, plus using the Adam optimizer, made the MLP architecture go from a median average accuracy per class (based on 15 iterations) of $83.75 \pm 0.13\%$ to $85.25 \pm 0.05\%$ for the complex model and from $83.31 \pm 0.11\%$ to $84.38 \pm 0.16\%$ for the real model. For the output layer, the *softmax* activation function [Goodfellow et al., 2016] has been used. For the complex-valued hidden layers, activation functions are separately applied to both real and imaginary parts so that $\mathbb{C}\mathrm{ReLU}(z) = \mathrm{ReLU}(\mathrm{Re}(z)) + i\,\mathrm{ReLU}(\mathrm{Im}(z))$. These type of activation functions are known as complex Type A activation function according to [Barrachina et al., 2021b, Kuroe et al., 2003] (also explained in Section 2.3).

A Normal weight initialization by K. He in [He et al., 2015] was used, and the bias was initialized as zero. The adaptation for complex-valued weights initialization is described in Section 2.6 and [Trabelsi et al., 2017, p. 6], which has to be done with care to keep the benefits of the K. He initialization on the complex domain.

A categorical cross-entropy loss function was used for all models. For complex models, the loss is computed twice, using first the real part and then the imaginary part as the prediction result. An average of the two error values is then calculated to be optimized. This loss function is known as complex average categorical cross-entropy ($\mathcal{L}^{ACE}$) and was explained in Section 2.4.

Both Complex- and Real-Valued MLP architectures had two hidden layers. For the CV-MLP, 96 and 180 neurons were used for the first and second hidden layers, respectively, as presented in [Cao et al., 2019]. The hidden layers sizes of the RV-MLP were dimensioned to have the same amount of *real-valued training parameters* with the same aspect ratio as explained in Section 5.1 (Figure 5.1d). The MLP models presented some overfitting for what dropout with 50% rate was used, which ameliorated the performance.

Throughout literature, CV-CNNs are the most popular CVNN architecture used for PolSAR. All References [Zhang et al., 2017, Sun et al., 2019, Zhao et al.,

2019a, Zhao et al., 2019b, Qin et al., 2021] identically dimensioned the model with the same amount of layers and kernels. Therefore, we decided to use the same architecture, which presents two convolutional layers, with 6 and 12 kernels, respectively, for the complex model. Again, for the real model, their size was dimensioned as explained in Section 5.1. All kernels were of size 3×3. The real model was dimensioned as explained in Section 5.1. Conventional arithmetic average pooling was used between both convolutional layers, whose extension to the complex domain is explained in Section 2.1.1. The model presents a fully connected layer at the end to perform the classification.

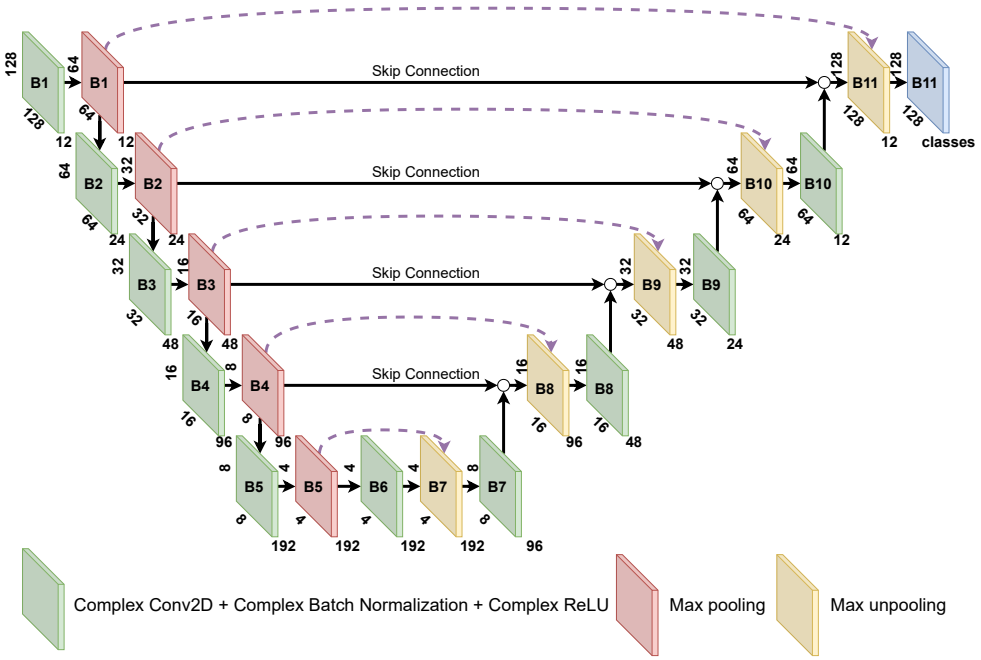


Figure 5.3: Complex-Valued Fully Convolutional Neural Network (CV-FCNN) diagram.

Finally, CV-FCNN (Figure 5.3) was implemented as described on [Cao et al., 2019]. which is composed of the downsampling or feature extraction part and the upsampling part. The downsampling part presents several blocks (B1, B2, B3, B4, B5, and B6). Each block has two sub-modules that are represented in Figure 5.3 in green and red colors. The upsampling part presents blocks B7, B8, B9, B10, and B11, which, in terms, are a combination of the other two sub-modules, the second one being the same green sub-module present in the downsampling section. The first sub-module (yellow) is a max-unpooling [Zafar et al., 2018] module as explained in Section 2.1.2.

The green sub-module is a combination of a convolution layer, a Batch Normalization (BN) (explained in Section 2.5) and Complex-Rectified Linear Unit (ReLU).

Reference [Cao et al., 2019] mentions using dropout but does not indicate at which points nor their rate. Different dropout rates were tested at several stages, such as the downsampling or upsampling part, without any appreciable amelioration (and sometimes the opposite). For this reason, no Dropout was used for this model. This can be explained as BN also acts as a regularizer, in some cases eliminating the need for Dropout [Srivastava et al., 2014]. The convolutional filter present on each layer was of size 3×3, and the number used for each layer is represented in Figure 5.3 for the complex model. As usual, the definition in Section 5.1 was used to dimension the real-valued model.

The red sub-module is a max pooling layer whose main objective is to shrink the image into smaller ones by keeping only the maximum value within a small window, in our case, of size 2×2. For the complex case, the absolute value of the complex number is used for comparison as proposed in [Zhang et al., 2017]. This layer complements the max-unpooling sub-module (yellow), which receives the locations where the maximum value was found. The max-unpooling layer (explained in Section 2.1.1) enlarges the input image by placing their pixels according to the maxed locations received from the corresponding max-pooling layer [Zafar et al., 2018].

The last blocks of the downsampling and upsampling parts (B6 and B11) have some differences with respect to the other blocks. B6 removes the max-pooling layer (red) completely. B11, on the other hand, replaces the ReLU activation function with a *softmax* activation function to be used for the output layer.

Each model was evaluated over 50 Monte-Carlo trials to be able to extract statistical analysis.

5.3 . Experiments and results

Throughout this and all following Sections of the Chapter, the median error was computed as in [McGill et al., 1978]; if median intervals do not overlap, there is a 95% confidence that their values differ [Chambers, 2018]. The confidence interval of the mean is calculated for a confidence level of 99%.

This Section implements several CVNN models, all of them described in the previous Section, with the coherency matrix as input representation, providing a thorough analysis that involves several independent trials for each network in order to infer appropriate errors and statistics. The dataset format used for each model was described in detail in Section 4.4. We also implement all the equivalent-RVNN [Barrachina et al., 2021c, Barrachina et al., 2022d] as described in Section 3.2.1. This definition asserts the same quantity of *real-valued trainable parameters* for both complex- and real-valued models maintaining the *aspect ratio* for each hidden layer.

We then show which model architecture is best suited for PolSAR applications and prove that using a Complex-Valued model is desirable regardless of the chosen

(a) Oberpfaffenhofen image (b) Ground Truth

Figure 5.4: Oberpfaffenhofen image and ground truth. **A** Built-up Area; **B** Woodland; **C** Open Area.

architecture for two PolSAR datasets.

5.3.1 . Oberpfaffenhofen dataset

E-SAR is an airborne experimental synthetic aperture radar system that is operated by the Microwaves and Radar Institute in cooperation with the German Aerospace Center (DLR) flight facilities. E-SAR delivered its first images in 1988 in its basic system configuration. Since then, the system has been continuously upgraded to become what it is today: a versatile and reliable workhorse in airborne Earth observation with applications worldwide [German Aerospace Center (DLR), 1988b]. The measurement modes include single-channel operation with either one wavelength and polarisation at a time or SAR Interferometry and SAR Polarimetry. The system is polarimetrically calibrated in L- and P-band. Single-pass SAR Interferometry is operational in X-band (along or cross-track), whereas repeat Pass SAR Interferometry works in L- and P-band, especially in combination with polarimetry [German Aerospace Center (DLR), 1988a].

Figure 5.4a shows the Oberpfaffenhofen, Germany database taken by E-SAR at L-band. The dataset can be downloaded at the European Space Agency (ESA) website. It can be seen in Figure 5.4b the ground truth for 3 different classes, built-up areas (25.01%), woodland (18.81%) and open areas (56.18%) for a total of 1,311,618 labeled pixels. These labels were obtained from [Zhang et al., 2017] as with the Flevoland ground truth.

Experiments were done with all np, alternate-tp and ratio-tp real equivalent models, analogously as those shown in Figure 5.1. The MLP architecture was used for the Oberpfaffenhofen dataset pixel-wise classification. Figure 5.5 shows the

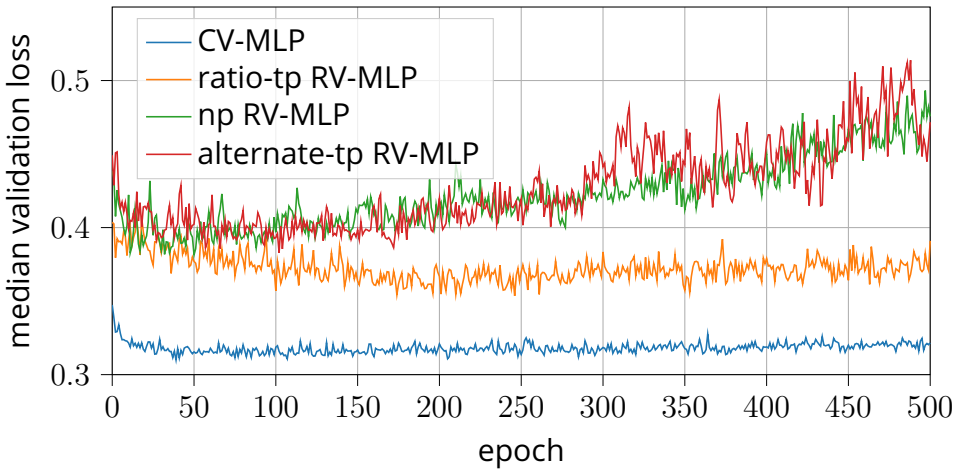


Figure 5.5: Real-Valued Equivalent-MLP comparison.

first 500 epochs for the validation loss value of these simulations. It can be seen that both the np and alternate-tp models presented overfitting. In contrast, this was not present (or to a very limited extent) in the ratio-tp and complex models. Therefore, we can conclude that using the proposed ratio-tp technique works best for this case of study, giving proof that our method might be the one to be favored in these cases for which we will only use this equivalent network definition for the rest of our simulations.

Simulations were done using the Oberpfaffenhofen dataset for the three main neural networks mentioned in Section 5.2. Both complex- and real-valued model architectures were implemented for each of them. Each model was evaluated over 50 Monte-Carlo trials to be able to extract statistics analysis.

Statistical indicators of both the OA and AA are summarized in Table 5.1 for the 6 experimental models. From Table 5.1, it is evident that FCNN outperforms CNN which, at the same time, outperforms MLP. For all three model architectures, CVNN outperforms their real-valued equivalent model in both OA and AA metrics. The highest achieved accuracy was achieved by the CV-FCNN architecture with an OA of 98.55% and an AA of 98.14%. Achieving the highest mean or median does not guarantee the most performing trained model. Indeed, we can argue that the maximum obtained value may be more important than the average accuracy as, in most cases, they will only use the most performing model for their end-user application. In this case, CV-FCNN was also the model that obtained both the upper 75% of cases and the maximum highest accuracy.

Unfortunately, the dataset is highly imbalanced, having much more occurrences of class C (Open Areas) than the other classes. In particular, class C always obtained a significantly higher accuracy than class B (Woodland) as it can be appreciated in Figure 5.6 which caused the OA to be higher than the AA. Because of application purposes, it is normally desired that a model performs better on

		Overall Accuracy (OA)			
		median	mean	IQR	range
FCNN	CV	**98.55 ± 0.21**	98.42 ± 0.09	$97.99 - 98.94$	$99.91 - \mathbf{99.44}$
	RV	98.23 ± 0.15	98.30 ± 0.08	$98.02 - 98.69$	$96.83 - 99.28$
CNN	CV	96.45 ± 0.04	96.45 ± 0.02	$96.36 - 96.52$	$96.21 - 96.68$
	RV	96.32 ± 0.04	96.32 ± 0.02	$96.24 - 96.44$	$95.89 - 96.65$
MLP	CV	88.87 ± 0.03	88.86 ± 0.02	$87.78 - 88.93$	$88.61 - 89.13$
	RV	88.03 ± 0.13	87.94 ± 0.06	$87.64 - 88.24$	$86.90 - 88.91$

		Average Accuracy (AA)			
		median	mean	IQR	range
FCNN	CV	**98.14 ± 0.28**	97.68 ± 0.23	$97.38 - 98.68$	$90.97 - \mathbf{99.41}$
	RV	97.79 ± 0.30	97.38 ± 0.22	$96.93 - 98.31$	$91.54 - 99.00$
CNN	CV	95.69 ± 0.05	95.68 ± 0.02	$95.57 - 95.81$	$95.27 - 96.00$
	RV	95.50 ± 0.06	95.47 ± 0.03	$95.34 - 95.63$	$94.82 - 95.93$
MLP	CV	85.25 ± 0.05	85.24 ± 0.04	$85.13 - 85.38$	$84.60 - 86.03$
	RV	84.38 ± 0.16	84.25 ± 0.08	$83.92 - 84.62$	$82.59 - 85.42$

Table 5.1: Test accuracy results (%) on Oberpfaffenhofen dataset.

classifying classes equally without any preference due to frequent occurrences for what we decided to favor AA over OA.

Figure 5.7 shows the median model prediction per model[1]. The performance difference between FCNN, CNN and MLP remains clear based on the predicted image. However, the better generalization gained when using a complex model is much harder to visualize, although the difference can be seen in particular sections of the image.

5.3.2 . Flevoland dataset

Flevoland dataset is a subset of an L-band PolSAR agriculture area of the Netherlands acquired by NASA/JPL (Jet Propulsion Laboratory) AIRSAR in 1989 during the MAESTRO-1 Campaign (Figure 5.8). This dataset contains 157,296 labeled pixels distributed among the following 15 classes Steambeans (3.88%),

[1]Note that is a single prediction given by the median OA model and not an average of all simulations

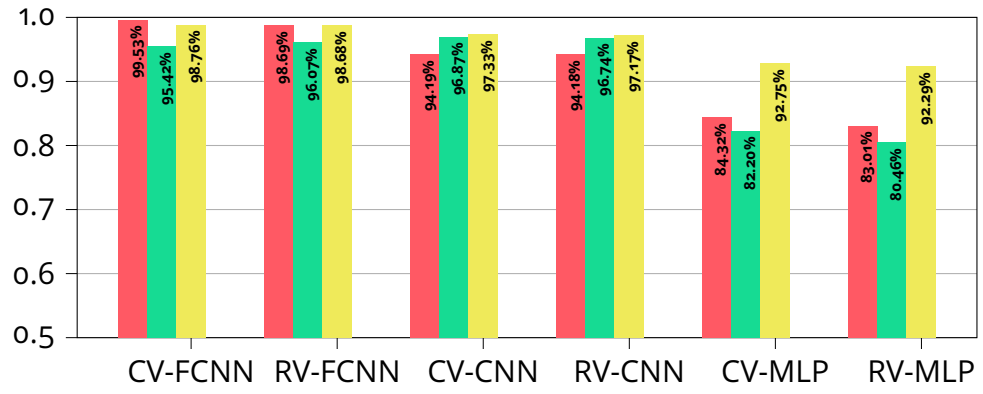
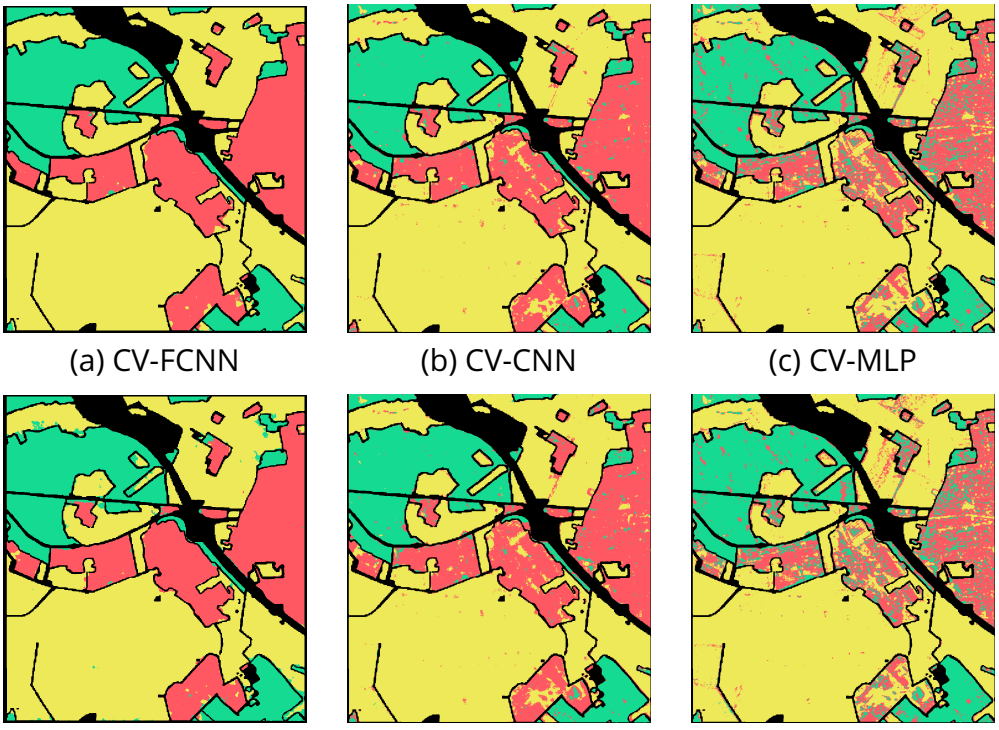


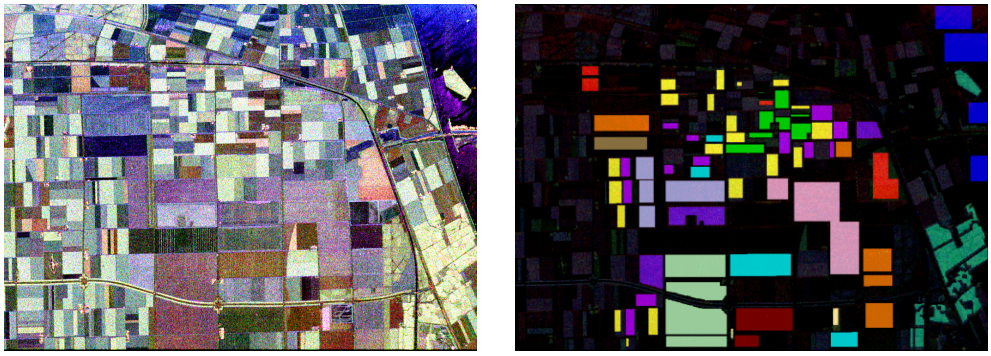
Figure 5.6: Accuracy per class for all models on Oberpfaffenhofen dataset. **A** Built-up Area; **B** Woodland; **C** Open Area.



(a) CV-FCNN (b) CV-CNN (c) CV-MLP

(d) RV-FCNN (e) RV-CNN (f) RV-MLP

Figure 5.7: Median Overall Accuracy model predictions of Oberpfaffenhofen dataset. **A** Built-up Area; **B** Wood Land; **C** Open Area.



(a) Flevoland image (b) Ground truth

Figure 5.8: Flevoland image and ground truth. **A** Steambeans; **B** Peas; **C** Forest; **D** Lucerne; **E** Wheat; **F** Beet; **G** Potatoes; **H** Bare Soil; **I** Grass; **J** Rapeseed; **K** Barley; **L** Wheat 2; **M** Wheat 3; **N** Water; **O** Buildings.

Peas (5.79%), Forest (9.50%), Lucerne (6.02%), Wheat (10.99%), Beet (6.39%), Potatoes (9.72%), Bare Soil (1.96%), Grass (3.99%), Rapeseed (8.07%), Barley (4.55%), Wheat 2 (6.73%), Wheat 3 (13.54%), Water (8.57%) and Buildings (0.30%). Labels were from Reference [Zhang et al., 2017].

Simulations on both CV-FCNN and RV-FCNN architectures, are performed on the Flevoland dataset. The simulations were repeated 50 times each in order to infer proper statistical analysis of the results. On each trial, the train, validation, and test sets are randomly sampled, so no two simulations have the same dataset split. Each training includes 1000 epochs.

Statistical indicators of both the Overall Accuracy (OA), which is the ratio of the number of correctly predicted pixels with respect to the total pixels, and the Average Accuracy (AA), which is an average of the accuracy for each class independently, are summarized in Table 5.2 for the test set. Although high accuracy makes the performance difference between the models seem small, confidence intervals remain far apart, which allows concluding that CV-FCNN generalizes better than RV-FCNN. 75% of CV-FCNN simulations achieve more than 99.74% OA whereas only 25% of RV-FCNN iterations achieve it.

The validation accuracy progression over epochs is shown in Figure 5.9; only the first 300 epochs are depicted, as the graph does not change much after that. It can be appreciated that complex-valued models converge quicker and with less variance than real-valued ones.

Figure 5.10 shows the box plot for both OA and AA. These plots allow a better appreciation of the outliers, which is not possible to appreciate in Table 5.2. Finally, Figure 5.11 shows the median[2] OA models prediction. Those correctly predicted

[2]Note that is a single prediction given by the median OA model and not an average

		CV-FCNN	RV-FCNN
OA	median	**99.80** $\pm$ 0.02	99.67 $\pm$ 0.03
	mean	99.79 $\pm$ 0.01	99.66 $\pm$ 0.02
	IQR	99.74 $-$ 99.84	99.58 $-$ 99.74
	full range	99.58 $-$ 99.91	99.38 $-$ 99.88
AA	median	**98.55** $\pm$ 0.38	98.25 $\pm$ 0.44
	mean	98.35 $\pm$ 0.19	97.87 $\pm$ 0.23
	IQR	97.84 $-$ 99.52	97.08 $-$ 99.10
	full range	94.20 $-$ 99.87	93.07 $-$ 99.75

Table 5.2: FCNN test accuracy results (%) on Flevoland dataset.

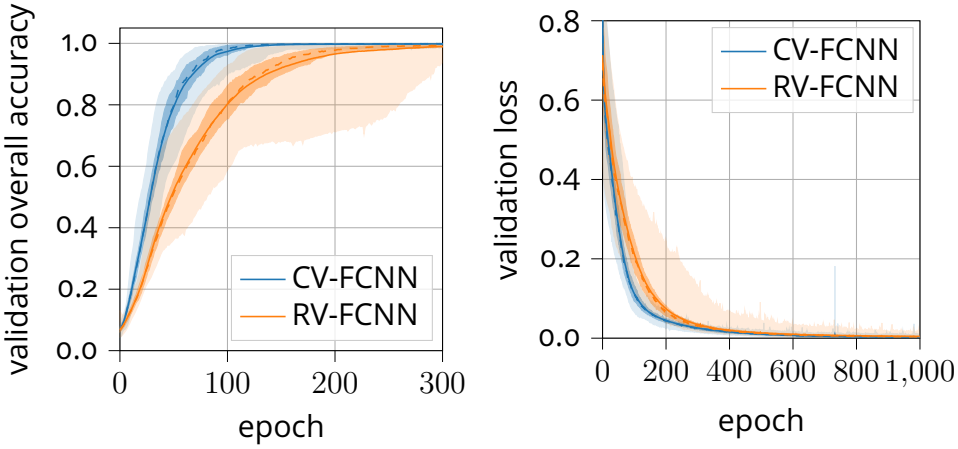


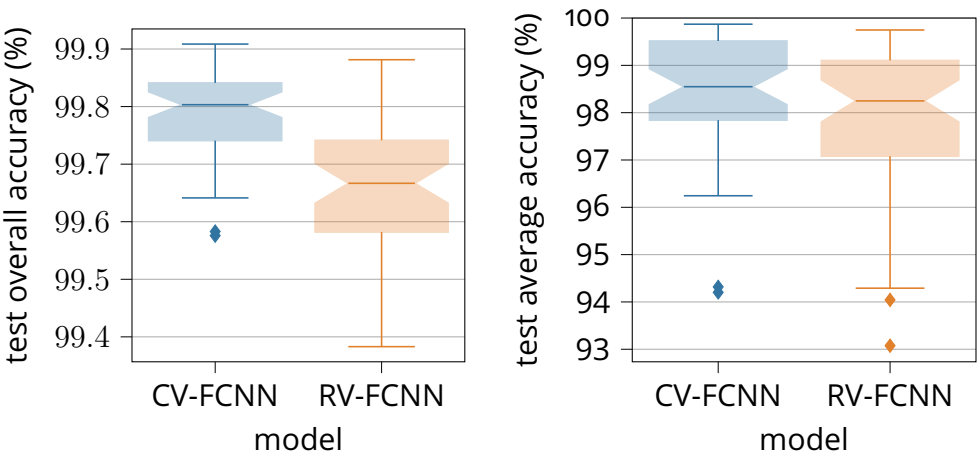
Figure 5.9: Validation evolution per epoch on the Flevoland dataset.

pixels are depicted in white, and the wrong predictions are in red.

5.4 . Studies on input representation

To our best knowledge, all the work mentioned above used the coherency matrix as their network input representation regardless of the PolSAR application. Even though the coherency matrix is a well-known attribute in unsupervised PolSAR image classification used for e.g. H-α, Wishart classifier, etc., it might not be adapted to the supervised learning framework for pixel-wise segmentation tasks. The averaging operation performed on Equation 4.9, whose main objective is to reduce noise at the expense of losing resolution, mixes values of adjacent pixels, which is not favorable for pixel-wise classification. The averaging algorithm can be viewed as a non-trainable convolution operation on $\boldsymbol{k}\,\boldsymbol{k}^H$ with identical kernel

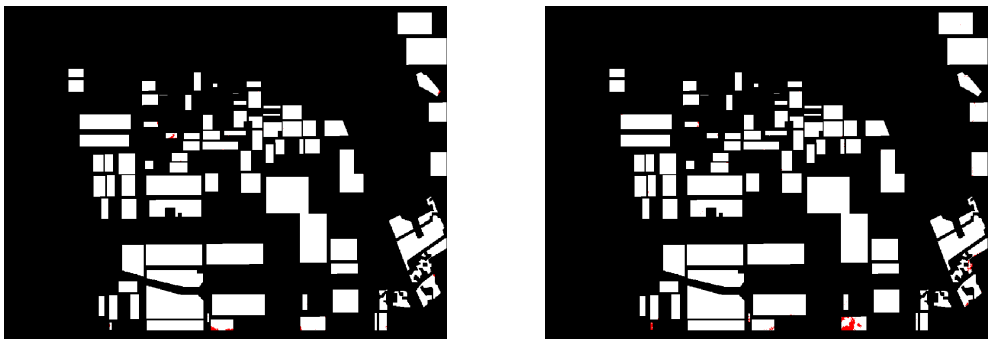
of all simulations



(a) Test Overall Accuracy.

(b) Test Average Accuracy.

Figure 5.10: Flevoland test accuracy box plot.



(a) CV-FCNN

(b) RV-FCNN

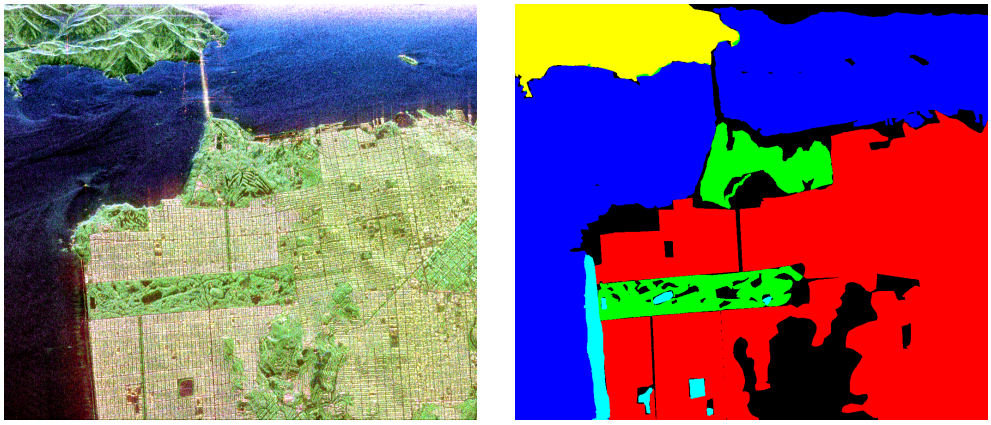
Figure 5.11: Median Overall Accuracy model predictions of Flevoland dataset.

weights $\frac{1}{n}$. Letting these kernels be trainable could enhance the performance of classification and segmentation.

Additionally, the diagonal elements of the coherency matrix are real-valued, which is a desirable property in certain cases, but that has no interest when using CVNNs as they can deal with complex-valued data naturally. Therefore, we propose to use Pauli vector $\boldsymbol{k}$ as CVNN input whenever this data format is available.

5.4.1 . Analysis on San Francisco dataset

To validate our statement, we propose to evaluate the performance of both complex- and real-valued FCNN networks. Unfortunately, both the Flevoland and Oberpfaffenhofen dataset used in Chapter 5 are in the form of the coherency matrix, the Pauli vector is not retrievable from the coherency matrix due to the



(a) SF-AIRSAR image (b) Ground truth

Figure 5.12: San Francisco image and ground truth. **A** Mountain; **B** Water; **C** Urban; **D** Vegetation; **E** Bare Soil.

boxcar averaging, and we were not able to find the Pauli vector or any prior format of the data online. Therefore, we decided to run our simulations on the San Francisco AIRSAR dataset for which the Sinclair values S_{HH}, S_{HV}, S_{VV} and S_{VH} are available.

The Airborne Synthetic Aperture Radar (AIRSAR) was designed and built by the Jet Propulsion Laboratory (JPL), which also manages the AIRSAR project [Jet Propulsion Laboratory (JPL), 2008]. AIRSAR served as a NASA radar technology for demonstrating new radar technology and acquiring data for the development of radar processing techniques and applications. As part of NASA's Earth Science Enterprise, AIRSAR first flew in 1988 and flew its last mission in 2004. Two images from this mission were used in this work.

San Francisco AIRSAR (Airborne Synthetic Aperture Radar) L-band dataset (Figure 5.12a) taken on august 1989. With a spatial resolution of 10m. The ground truth was obtained from [Liu et al., 2019]. The dataset presents 802,302 labeled pixels with five classes with different occurrences, which are Mountain (7.81%), Water (41.08%), Urban (42.73%), Vegetation (6.67%), and Bare soil (1.71%), as it can be seen on Figure 5.12b.

5.4.2 . Experiments and results

Four simulations were done using both CV-FCNN and RV-FCNN architectures, discussed in Section 5.2, with the two discussed input representations. These four simulations were performed 50 times each in order to be able to infer statistical analysis over the results. On each trial, the train, validation and test sets were randomly sampled, so no two simulations have the same dataset split. For each training, 400 epochs were made.

		CV-FCNN		RV-FCNN	
		Pauli vector	Coherency matrix	Pauli vector	Coherency matrix
AA	median	**98.00** $\pm$ 0.27	96.80 ± 0.25	96.75 ± 0.32	95.20 ± 0.44
	mean	97.55 ± 0.15	96.54 ± 0.12	96.39 ± 0.18	94.98 ± 0.21
	full range	$93.90 - 98.79$	$93.44 - 98.63$	$92.37 - 97.69$	$91.06 - 97.64$
OA	median	**99.64** $\pm$ 0.01	99.45 ± 0.02	99.40 ± 0.02	99.19 ± 0.03
	mean	99.64 ± 0.01	99.44 ± 0.01	99.40 ± 0.01	99.18 ± 0.02
	full range	$99.53 - 99.70$	$98.91 - 99.61$	$99.16 - 99.53$	$98.76 - 99.43$

Table 5.3: FCNN test accuracy results (%) on San Francisco dataset.

Figure 5.13 shows the validation accuracy evolution over 250 epochs, after which the difference in performance can no longer be appreciated on that graph. It can be seen that the complex-valued models converge faster than the real-valued models. However, the final achieved accuracy at epoch 400 can be better appreciated on the box plot in Figure 5.14. Note that the use of the Pauli vector representation increases the accuracy and achieves a lower variance than using the coherency matrix input for both real and complex architectures.

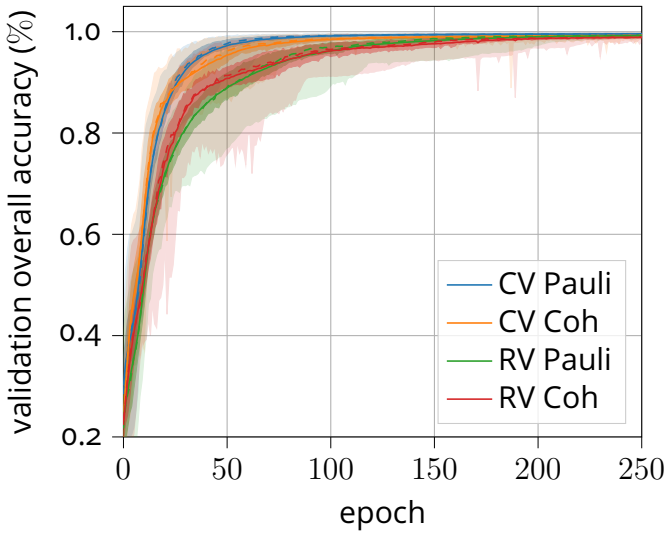


Figure 5.13: Validation Overall Accuracy evolution per epoch on the San Francisco dataset.

Table 5.3 depicts the results obtained with the test set. The high accuracy obtained mainly for OA makes it harder to discern between the model accuracies, although confidence intervals remain far apart. When comparing the AA, however, the median accuracy between the input representation methods presents more than a 1% difference which, above 95%, is highly significant.

Both using a complex-valued architecture and Pauli vector input representation clearly increase accuracy. However, although using a complex-valued architecture may seem to be slightly more significant, the confidence intervals do not allow us to assert such conclusions.

There was a significant difference between the OA and AA. This was due to a highly different classification accuracy per class, as can be appreciated in Figure 5.15. The Figure also makes evident that using CVNN achieves a significant amelioration over using RVNN. Finally, Figure 5.16 shows the median predicted image for all models and input representations tested.

5.5 . SAR image splitting

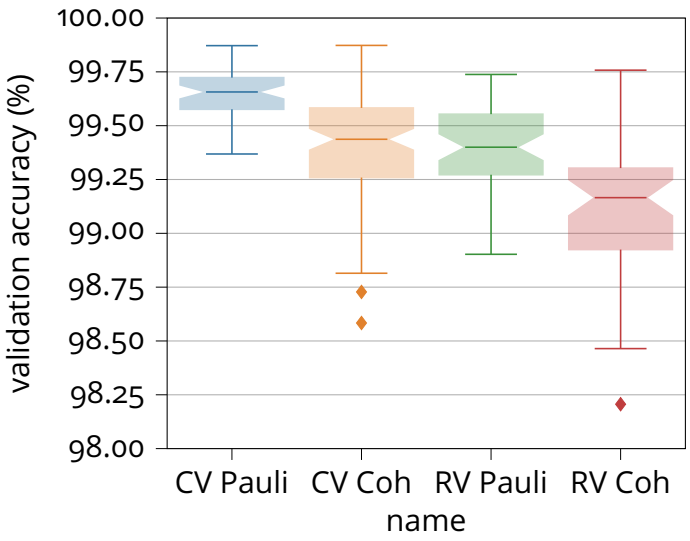


Figure 5.14: Validation Overall Accuracy box plot [Williamson et al., 1989]

PolSAR classification or segmentation tasks must use the same image for train, validation and testing. This is because not only is it hard to find open-source PolSAR images but from the existing cases, the radar properties such as frequency, spatial resolution or aperture. Even in the unlikely case that there are at least two images with the same parameters, they are usually not labeled, and if they are, labels might be different. For example, in the two images explored in Chapter 5, one had three classes, one of them Open Fields, whereas the second image has 15 classes which can all be grouped under the Open Fields label as they are all cases of different crops.

Most existing works, end even ourselves in previous Sections of this Chapter, obtain different image patches through the sliding window operation [Li et al., 2018b]. This method generates smaller image patches by sliding a window through the image with a given stride. In particular, if the stride is smaller than the window size, as happened with Reference [Cao et al., 2019], the generated image patches will share pixels and ground truth. Several of the mentioned articles divide training and test sets randomly from the generated images. The major drawback is that this overlap will be present between the train and test set. Even if the stride is bigger than the window size, a test set image can be spatially next to a training set image for what there might be some overfitting happening without notice.

Current *state-of-the-art* PolSAR classification methods obtain an accuracy over 95%, making it harder to compare new models, but this high accuracy might be the result of this unfortunate dataset generation.

To prevent this issue and evaluate its impact on the test set accuracy, we propose to first divide the image vertically into three sub-images for training, validation

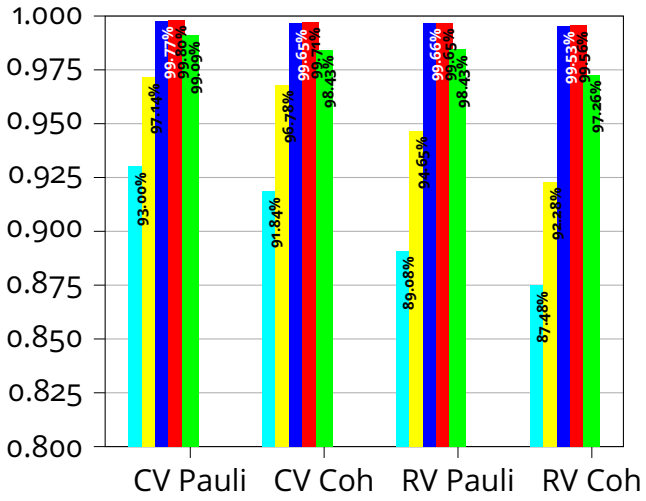
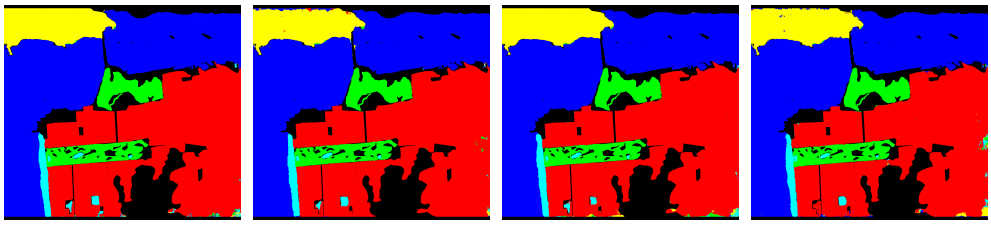


Figure 5.15: Test accuracy per class for all models on San Francisco dataset.



(a) CV-FCNN Pauli representation (b) CV-FCNN Coherency matrix (c) RV-FCNN Pauli representation (d) RV-FCNN Coherency matrix

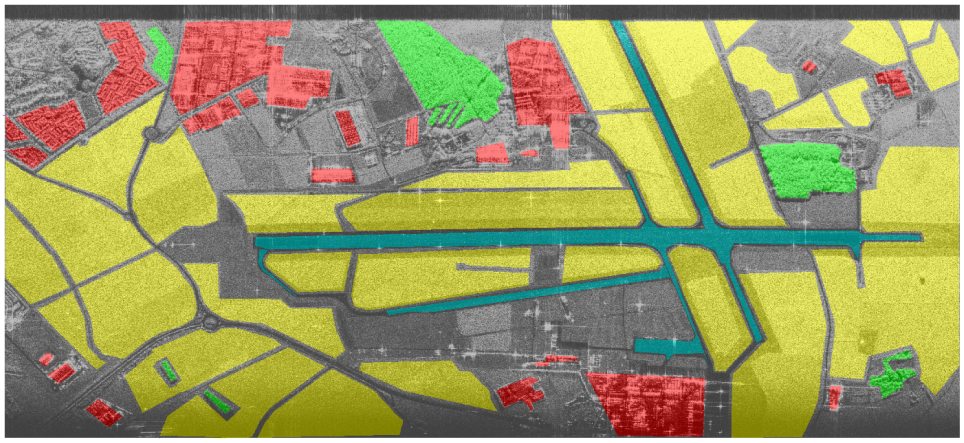
Figure 5.16: Median Overall Accuracy model predictions using both Pauli and coherency matrix representation.

and test, respectively, before applying the sliding window operation to extract the smaller image patches. Using this method, the subsets will be farther apart, and a small stride can be used to have more image patches without the risk of overlap between the different subsets. Preferably, the image should be split in the azimuth direction as pixels in a different swath value would have different range resolution, as it was explained in Equation 4.3.

The problem with this method is twofold, it requires the split direction (preferably the azimuth) to have some minimum length for the dataset to be large enough. This is normally not an issue as the azimuth direction is normally the larger axis for being the direction the airborne moves to capture the image. However, the real limitation is that all classes should be present in all three sub-images.

5.5.1 . Bretigny dataset

(a) Bretigny image



(b) Ground truth

Figure 5.17: Bretigny image and ground truth. **A** Built-up Area; **B** Wood Land; **C** Open Area; **D** Runway

None of the previously used datasets could be easily divided into three sub-images that presented all classes. For this reason, we decided to use ONERA's proprietary PolSAR image, which presented these features.

The Electromagnetic and Radar Science Department (DEMR) of ONERA, the French Aerospace Research Agency developed the RAMSES (Radar Aéroporté Multi-spectral d'Etude des Signatures) PolSAR system in 2002 with funding from the DGA (Direction Générale de l'Armement) and CNES (Centre National d'Études Spatiales). RAMSES was developed mainly as a test bench for new technologies and to provide specific data for TDRI (Target Detection, Recognition, and Identification) algorithm evaluation. It is flown on a Transall C160 platform operated by the CEV (Centre d'Essais en Vol).

RAMSES can be configured with three bands picked among P-(430 MHz),

L-(1.3 GHz), S-(3.2 GHz), C-(5.3 GHz), X-(9.5 GHz), Ku-(14.3 GHz), Ka-(35 GHz), and W-(95 GHz) bands totaling for eight different bands. From those eight, six (all but Ka and W) operate in fully polarimetric mode. The associated bandwidth and waveforms can be adjusted to meet the data acquisition objectives, and the incidence angles can be set from $30°$ to $85°$. The X-band and the Ku-band systems are interferometric and can collect PolInSAR mode imagery in multi-baseline configurations either along-track, cross-track, or both. [Dubois-Fernandez et al., 2002].

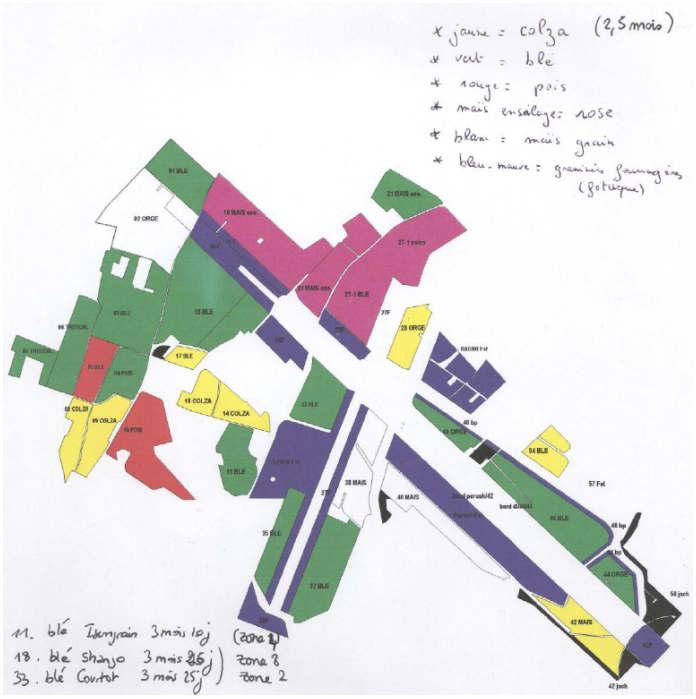
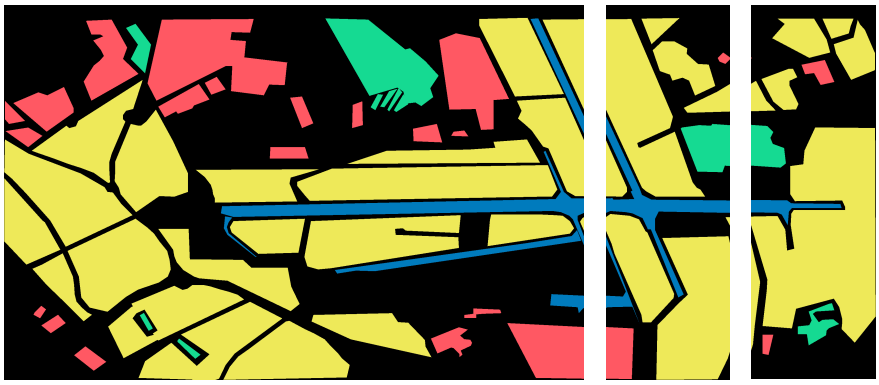


Figure 5.18: Ground truth of different crops of Bretigny area.

ONERA's proprietary PolSAR image of Bretigny, France [Formont et al., 2010] whose area is shown in Figure 5.17a. This image was measured with RAMSES at X-band with a resolution of 1.3m. The image has a spatial resolution of 2m, an incidence angle of $30°$, and an X frequency band. 2,871,080 labeled pixels of four classes, which are Open Area (73.20%), Wood Land (5.76%), Built-up Area (14.43%), and Runway (6.61%), were manually labeled. However, although there was a single class for the fields (Open Area), there are different types of crops, as shown in Figure 5.18, this can impact the prediction accuracy negatively. Indeed, using *k-means* to group classes automatically performs poorly when applied directly on the coherency matrix representation, for example clustering some open field crops with the forest. On the other hand, the algorithm performs better when eight classes are provided, so the class of the Open Field is divided into more sub-classes [Inisan et al., 2022].

The dataset was split as shown in Figure 5.19. 70% of the image was used as a training set, and 15% was used for both validation and test set. Note that the four classes are present in each sub-image as shown in Figure 5.19. The sliding window operation is then applied to each sub-image to generate train, validation, and test sets separately. We used a stride of 25 for the sliding window and an input image size of 128×128. This method not only avoids the ground-truth overlap but also prevents training and validation patches from being spatially next to each other, reducing correlation between datasets.



(a) train (b) val (c) test

Figure 5.19: Split of Bretigny dataset; 70% as the training set, 15% as the validation set, and 15% as the test set. **A** Built-up Area; **B** Wood Land; **C** Open Area; **D** Runway.

5.5.2 . Experiments and results

Five semantic segmentation Monte Carlo trials, using CV-FCNN and RV-FCNN models, were performed for the following results, each involving 150 epochs and a batch size of 30. As results in Section 5.4 show that it is better to use the Pauli vector for the FCNN model architecture, we employ that as the input representation.

In Figure 5.20, we can see the accuracy and loss curves for both the training and validation sets. A solid line represents the mean value, whereas the colored area is the interquartile range. In this Figure, it is possible to appreciate that CV-FCNN generalized better during the ensemble of the training.

Validation loss was used to select the best model checkpoint for each trial. The final performance was then computed using the test set whose results are displayed in Table 5.4 with their associated confidence interval. According to the values shown in Table 5.4, the number of iterations was enough to assert that CV-FCNN merits are statistically justified as confidence intervals for both mean and median do not overlap. Even more, these simulations where the first case of results not overlapping in any moment, which means the higher obtained accuracy

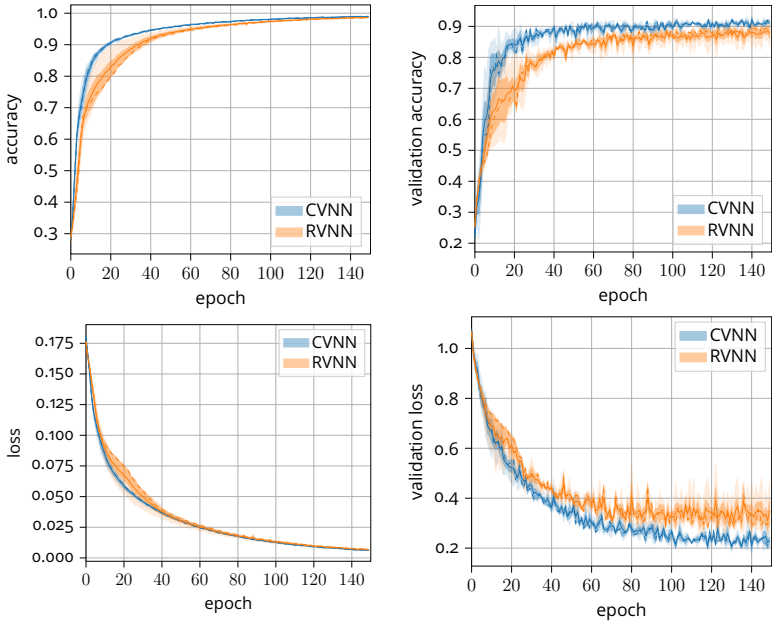


Figure 5.20: CV-FCNN vs RV-FCNN accuracy and loss evolution per epoch

of RV-FCNN remained under the lowest obtained accuracy of the CV-FCNN model as it can be better appreciated on Figure 5.21. This indicates that CVNN merits might be higher than expected and that the close performance might be an issue of a saturated problem where accuracy is already very high.

By running the simulation analogous to previous simulations (without the dataset split), CV-FCNN architecture obtained 99.83% Overall Accuracy and RV-FCNN had 99.69%. These results make clear the impact of the dataset splitting and at which point the task was saturated.

	CV-FCNN	RV-FCNN
median	92.76 ± 0.36	89.86 ± 0.96
mean	92.77 ± 0.46	89.92 ± 1.23
full range	$93.17 - 92.37$	$91.02 - 88.89$

Table 5.4: FCNN test Accuracy results (%) on Bretigny dataset.

Figure 5.22 shows the predicted image of a randomly chosen CV-FCNN (5.22a) and RV-FCNN (5.22b) models. The Figure allows appreciate the effect of the dataset split method detailed in Section 5.5 as we see how both models achieve a better representation on the left of the image (training set) and lower performance on the right (validation and test set). On the other hand, it is possible to appreciate that CV-FCNN does better work predicting the ground truth.

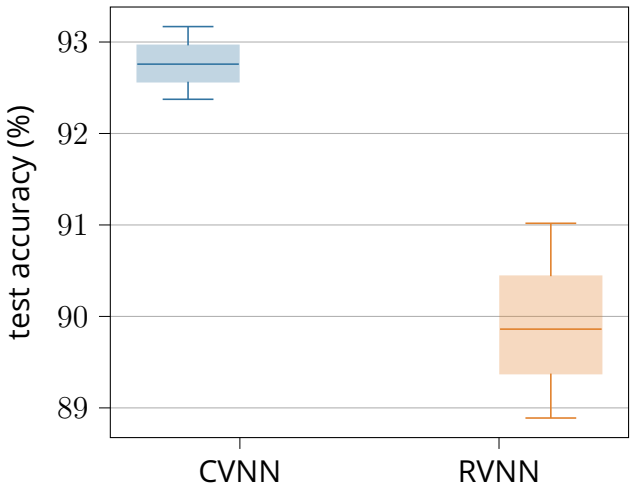


Figure 5.21: Bretigny split method test Average Accuracy box plot.

5.6 . Conclusion

Until recently, the limitations of existing work could not clear doubts about CVNN out-performance against RVNN for PolSAR classification. In this Chapter, we cleared the doubts by performing semantic segmentation on two different PolSAR images.

The experiments were done for three structurally different models indicating that using complex-valued architectures improves the performance of a conventional neural network regardless of the chosen base model.

We make sure that the comparison was fair by dimensioning both the CVNN and RVNN to be capacity equivalent in terms of trainable parameters that maintain the same aspect ratio.

We performed an exhaustive comparison between CVNN and RVNN architectures for PolSAR semantic segmentation for both Flevoland and Oberpfaffenhofen images. We showed that CVNN outperforms RVNN for all types of model architectures. We also proved that Fully Convolutional Neural Network generalizes better than Convolutional Neural Network and MultiLayer Perceptron models for this application.

In this Chapter, we demonstrated on well-known San Francisco AIRSAR PolSAR database that the CV-FCNN architecture, implemented using our open-source toolbox [Barrachina, 2021], achieves a better performance than their equivalent-RV-FCNN for segmentation application.

We also showed that using the Pauli vector as input features increases the segmentation performance for both complex-valued and real-valued architectures. We, therefore, encourage the community to favor this form of input representation instead of the widely popular coherency matrix.

Furthermore, it is worth mentioning that the models used in the experiment

(a) CVNN prediction

(b) RVNN prediction

Figure 5.22: Median Overall Accuracy model predictions of Bretigny dataset.

were dimensioned for a coherency matrix dataset meaning there might be room for improvement in the Pauli vector representation. Yet, it is this last one that performed better. Furthermore, since $\boldsymbol{k}$ is a three-dimensional vector, in contrast, with the coherency matrix $\boldsymbol{T}$ which leads to a 6th-dimensional vector, the dataset would take less memory space when using the Pauli vector representation.

We later performed a statistical comparison of a CVNN against an equivalent RVNN on a new PolSAR dataset. We proposed splitting the dataset to reduce the overfitting of the test set for what the task was less saturated. Furthermore, this method allowed to use of a smaller stride for generating more training image patches. Results were lower than when using previous methods for dataset splitting, confirming this dataset split method generates a less saturated task. This showed a higher accuracy difference between CVNN and RVNN, obtaining non-overlapping results making clear the out-performance of CVNN over RVNN with both higher accuracy and lower variance. In comparison with the other results from this work and even the existing bibliography, this last method allowed showing a higher difference between the models and made CVNN more evident.

The next step of this simulation will be to obtain different PolSAR images for all the sub-sets. This would allow asserting the true merits of these segmentation methods and the possibility of adding a live predictor on board.

6 - Conclusion

With this work, we hope to motivate further work on CVNN by providing the means to implement them in practice with our published toolbox. We implemented complex layers, activation functions, losses and initialization techniques. For the latter, we propose and perform an experiment that proves the importance of the correct extension of the algorithms to the complex domain for better performance of the network.

Although the model design must be revised to make CVNN and RVNN equivalent, we compared that CVNN might even achieve higher results on real-valued data when using the Hilbert transform to cast the values to the complex domain.

We showed the potential interest of CVNN for data that presents a non-circular property. The wide extension of the experiments showed that CVNN performs better with less overfitting and variance regardless of the data properties, input representation and model dimension. This breakthrough on the potential fields of application for CVNN was the most upvoted paper of week 116 by the machine learning subreddit community, which accounts for two and a half million people.

In particular, we performed a thorough statistical analysis on PolSAR dataset. This study presented a framework for dimension and design models that are equivalent to be certain the comparison is fair. We tested three base models (MLP, CNN and FCNN) on four PolSAR images (Flevoland, Oberpfaffenhofen, San-Francisco and Bretigny). In all cases, CVNN presented higher accuracy and, in general, less variance and less overfitting. With this we hope to prove CVNN merits in practice for PolSAR classification and segmentation tasks.

We showed that using the Pauli vector as input representation to a FCNN architecture performs better than using the coherency matrix, the most popular input representation across research works. This is regardless if the model is complex- or real-valued, although, as usual, CV-FCNN out-performs RV-FCNN. Using the Pauli vector as input representation has the added value of needing less memory.

Particular attention was given to the SAR image splitting process, which might introduce a correlation between training, validation and test set if it is not done correctly. Notably, the validation and the test score can be too optimistic in single PolSAR image segmentation tasks We have proposed to split a PolSAR in a way to reduce this correlation and obtain lower accuracy that allows for a better appreciation between the difference of CVNN and RVNN performance with no intersection of results with CV-FCNN model less performing results from a total of 50 trials achieving 92.37% accuracy while highest achieved accuracy for RV-FCNN of 91.02%.

6.1 . Perspectives

The next natural step of this research will be to analyze the generalization potential of CVNN algorithms for PolSAR semantic segmentation tasks using different PolSAR images taken at different dates and/or different places but with the same classes. This will evaluate the actual accuracy of performing segmentation on newly acquired data in real-time.

The results on the temporal signal show even a potential interest where CVNN might even outperform RVNN for experimental data if a pertinent transformation is applied, like in our case, the Hilbert transform. This matter should be revised, and if proven to be true, it will open an endless field of application of CVNN.

CVNN opens the possibilities of many complex-valued activation functions, one of the most challenging parts when designing the network [Lee et al., 2022]. Although most of these alternatives were implemented in the published toolbox, the simulations to prove and compare their merits were left mostly unexplored in this work [Scardapane et al., 2018], limited to the functions described in Reference [Kuroe et al., 2003]. These types of activation functions are known as spit activation functions; however, fully complex activation functions may be more favorable to represent both magnitude and phase components [Lee et al., 2022]. It was not discussed in this work the possibility of using complex-valued learning rates as discussed in [Zhang and Mandic, 2016]. Even a complex-valued loss function could have numerical convergence when using Wirtinger calculus.

We argued that the main advantage of the coherency matrix representation is that, generally, an averaging filter is used to reduce speckling noise. Deep enough convolutional networks can easily learn to perform this despeckling technique themselves so that this averaging is indeed not necessary and indeed results in lower performance due to the loss of information involved when averaging the neighboring pixels. However, we do not test the impact of several more advanced despeckling techniques, which might have a much higher impact on the performance and potentially increase it to new *state-of-the-art* results.

In this thesis, we faced the problem of not having enough data to exploit in algorithms that are usually known as *data hungry* [Castro, 2020]. Generative Adversarial Network (GAN)s are often employed for data augmentation, also known as Data Augmentation GAN (DAGAN) [Antoniou et al., 2017]. This has even been explored for PolSAR datasets [Bargsten and Schlaefer, 2020]. However, using a complex-valued GAN model can play an important role when generating complex-valued data such as PolSAR images [Li et al., 2020].

Other PolSAR application tasks can be explored using CVNN such as super-resolution, change detection, object and target detection, style transfer, complex-valued autoencoder.

Apart from PolSAR, there are many other potential radar applications of CVNN, such as drone classification, where the input for the deep learning technique can be the complex spectrum of the micro-Doppler signal [Gérard et al., 2021]. Reference [Gérard et al., 2021] tries several input representations of the dataset

to assert which mode would be better for drone classification. These experiments could be re-done for a CVNN by casting the real-valued representation by means of, for example, the Hilbert transform.

A - Synthèse en français

Dans la communauté de l'apprentissage profond, la plupart des réseaux neurones sont entrainés pour inférer sur des données à valeurs réelles (signaux vocaux, images RVB, vidéos, etc.). Or, la communauté du traitement du signal a souvent besoin des théories basant sur l'analyse complexe des signaux. En effet, les signaux à valeurs complexes sont très présents dans une grande variété d'applications telles que les données biomédicales, issues de la physique, des communications et des radars. Toutes les données susmentionnées nécessitent les outils du traitement du signal [Schreier and Scharf, 2010], qui sont pour la plupart basés sur des représentations et des opérations à valeurs complexes telles que les transformées de Fourier et d'ondelettes, les filtres de Wiener et les filtres adaptés, etc

Ainsi, les réseau de neurones à valeurs complexes (CVNNs pour ses sigles en anglais) apparaissent comme un choix naturel pour traiter et apprendre de ces caractéristiques à valeurs complexes puisque l'opération effectuée à chaque couche des CVNNs peut être interprétée comme un filtrage ou des multiplications complexes. Notamment, les CVNNs sont plus adaptés que les RVNNs pour extraire les informations de phase [Hirose and Yoshida, 2012]. Pour cette raison, plusieurs domaines d'étude du traitement du signal utilisent déjà les CVNNs pour leurs expériences, comme le traitement du signal radiofréquence dans les communications sans fil [Gong et al., 2017, Zhang and Wu, 2017, Liu et al., 2017, Ding and Hirose, 2014, Marseet and Sahin, 2017], traitement d'images dans le domaine de la vision par ordinateur, telles que les tâches de classification et de segmentation [Akramifard et al., 2012, Hafiz et al., 2015, Popa, 2017, Amilia et al., 2015, Liu et al., 2014, Olanrewaju et al., 2011, Gu and Ding, 2018, Matlacz and Sarwas, 2018, Popa, 2018], traitement du signal audio [Al-Nuaimi et al., 2012, Kinouchi and Hagiwara, 1996, Kataoka et al., 1998, Hayakawa et al., 2018, Tsuzuki et al., 2013, Tsuzuki et al., 2013], prévision des vents [Sepasi et al., 2017, Çevik et al., 2018, Mandic et al., 2009], transformateurs de puissance [Chistyakov et al., 2011, Minin et al., 2012], contrôle des signaux de trafic [Nishikawa et al., 2005, Nishikawa et al., 2006], cryptographie [Dong and Huang, 2019], détection de spam [Hu et al., 2008], mémoire associative [Jankowski et al., 1996], applications médicales telles que le diagnostic de l'épilepsie [Peker et al., 2015] ou le traitement du signal IRM [Virtue et al., 2017] et en particulier les applications radar (notre domaine d'étude).

Au début des années 90, des travaux sur la rétropropagation complex pour les réseaux CVNNs ont été publiés [Hirose, 1992, Georgiou and Koutsougeras, 1992, Benvenuto and Piazza, 1992, Leung and Haykin, 1991], laissant le champ libre à sa mise en œuvre. Depuis, les travaux sur CVNN ont commencé à augmenter au milieu des années 1990 et 2000 ; [Jankowski et al., 1996, Kim and Adali, 2001b, Kim and Adali, 2001a, Kim and Adali, 2002, Miyauchi and Seki, 1992, Miyauchi et al., 1993, Jianping et al., 2002, Cha and Kassam, 1995] et en 2003, Akira Hirose, très

probablement le précurseur de CVNN a publié un premier livre sur le sujet en 2003 : [Hirose, 2003] et deux autres par la suite : [Hirose, 2013, Hirose, 2012]. La référence [Nitta, 2004] explique le potentiel d'un neurone à valeur complexe pour le concept de frontière orthogonale où l'intersection de deux hyper-surfaces peut diviser la frontière de décision en quatre régions, révélant ainsi la puissance de calcul potentielle des CVNNs par rapport aux RVNN. A. Hirose a expliqué que les mérites des CVNN résident principalement dans les propriétés de la multiplication complexe qui peut être vue comme une rotation de phase, et une modulation d'amplitude préconisant une réduction de liberté avantageuse : [Hirose, 2013, Hirose, 1992]. Des références [Hirose, 2009, Hirose, 2011, Hirose, 2010] ont discuté des mérites de CVNN en mettant en relation les expressions mathématiques des nombres complexes avec le domaine du traitement du signal. Deux études récentes ont été publiées récemment sur CVNN *état de l'art* des avancées [Bassey et al., 2021, Lee et al., 2022]. Ces deux travaux mentionnent la motivation biologique des CVNN qui représentent mieux leur comportement.

Récemment, nous avons montré que les CVNN sont plus performants dans la classification des données gaussiennes non circulaires que leur équivalents réelles, ce qui signifie que les CVNNs sont plus sensibles à l'extraction des informations contenue dans la phase que les RVNNs. Pour ce faire, nous comparons des vecteurs de données non circulaires aléatoires et montrons que les CVNN peuvent profiter de cette caractéristique et en extraire tout le potentiel en obtenant une précision plus élevée, moins de sur apprentissage et une variance plus faible que les RVNN. Nos résultats ont également été cités par la référence [Ko et al., 2022] pour justifier certaines propriétés de leurs résultats obtenus qui observaient des carectéristiques similaires aux nôtres.

Les techniques d'apprentissage profond sont de plus en plus populaires et se sont étendues à la classification d'images radar et PolSAR [Fix et al., 2021, Marmanis et al., 2018, Marmanis et al., 2016b, Parikh et al., 2020, Konishi et al., 2021, Chen et al., 2016, Hou et al., 2016, Zhou et al., 2016]. Habituellement, ces réseaux sont alimentés par les informations d'amplitude de l'image PolSAR sans utiliser les données de phase.

Récemment, certaines publications ont commencé à utiliser les CVNNs comme alternative aux Real-Valued Neural Network (RVNN) conventionnels pour les applications radars [Hirose, 2013, Bassey et al., 2021] puisque les données radars sont généralement à valeurs complexes. Sachant que les données Synthetic Aperture Radar (SAR) sont non circulaires [El-Darymli et al., 2014, Vasile and Totir, 2012] et que par conséquent les informations de phase jouent un rôle crucial dans leur représentation [Datcu et al., 2007, El-Darymli et al., 2013, El-Darymli et al., 2015], il n'est pas étonnant que les Complex-Valued Neural Network soient de plus en plus utilisés pour les applications SAR, PolSAR ou InSAR [Wilmanski et al., 2016, Oyama and Hirose, 2018, Gleich and Sipos, 2018].

La référence [Hänsch and Hellwich, 2009a], a été l'une des premières à mettre

en œuvre un Complex-Valued MultiLayer Perceptron (CV-MLP) pour les applications PolSAR en 2009. Bien qu'une comparaison ait été faite avec le Real-Valued MultiLayer Perceptron (RV-MLP), aucun intervalle de confiance n'a été donné, ce qui empêche d'affirmer les mérites du CV-MLP. De plus, une représentation d'entrée différente a été utilisée pour chaque modèle, ce qui en fait une comparaison non pertinente. Plus tard, les mêmes auteurs ont suggéré de donner la même représentation d'entrée pour obtenir une comparaison plus précise entre les modèles sur [Hänsch, 2010]. Les références [Hänsch and Hellwich, 2010] et [De et al., 2017] ont également utilisé les CV-MLP sur une base de données PolSAR mais n'ont pas fourni de comparaison avec RV-MLP. La référence [Cao et al., 2019] a comparé CV-MLP à une RV-MLP mais même si CV-MLP a obtenu de meilleures performances que RV-MLP, les intervalles de confiance se croisent, ce qui laisse planer un doute sur la surperformance de CV-MLP.

Des travaux utilisant Complex-Valued Convolutional Neural Network (CV-CNN) ont été publiés pour des applications PolSAR. La référence [Zhang et al., 2017] compare un CV-CNN avec des RV-CNNs mais manque d'intervalles de confiance. D'autres travaux récents [Sun et al., 2019, Zhao et al., 2019a, Zhao et al., 2019b, Qin et al., 2021, Dong et al., 2020] utilisent un CV-CNN pour les applications PolSAR mais sans comparer son résultat avec un RV-CNN.

Les références [Xie et al., 2020] et [Shang et al., 2019] ont ajouté de la complexité à l'architecture CNN en utilisant un Complex-Valued Convolutional Neural Network récurrent et un Complex-Valued Convolutional AutoEncoder (CV-CAE) pour obtenir des résultats plus précis. Récemment, des références [Cao et al., 2019, Li et al., 2018a] ont atteint des performances *state-of-the-art* en utilisant une architecture de modèle Complex-Valued Fully Convolutional Neural Network (CV-FCNN). Tous les travaux précédemment cités d'applications CVNN sur PolSAR effectuent une tâche de classification par pixel, qui peut être considérée comme une tâche de segmentation sémantique. Il n'est donc pas surprenant qu'un modèle FCNN atteigne une précision supérieure puisqu'il effectue une segmentation sémantique par conception.

Dans tous les cas cités ici, les dimensions du modèle RVNN ne sont pas justifiées car le modèle RVNN a une capacité d'apprentissage généralement inférieure à celle du modèle CVNN ; [Mönning and Manandhar, 2018, Barrachina et al., 2021c, Barrachina et al., 2022d]. Pour cette raison, les mérites de CVNN par rapport à RVNN pour la classification PolSAR et la segmentation sémantique sont souvent biaisées.

Dans ce travail, nous avons fourni une méthode rigoureuse pour concevoir et dimensionner correctement un RVNN afin qu'il soit équivalent au réseau CVNN avec lequel nous voulons le comparer. Dans ce cadre, nous effectuons des tâches de classification sur deux ensembles de données PolSAR pour trois architectures différentes et leurs réseaux équivalents réels correspondants et nous montrons que pour la même tâche et la même architecture de modèle, les réseaux complexes sont

statistiquement plus performants.

La plupart des travaux cités ci-dessus utilisent la matrice de cohérence pour les données SAR polarimétrique comme la représentation d'entrée des réseaux. Cependant, la représentation du vecteur de Pauli est en général plus riche mais aussi plus bruité principalement à cause du spekle. Nous avons donc comparer les résultats des réseaux de neurones en utilisant soit le vecteur de Pauli ou la matrice de cohérence en entrée du réseau. Et nous montrons que les deux réseaux Complex-Valued Fully Convolutional Neural Network (CV-FCNN) et son modèle équivalent réel sont plus performants lorsqu'ils utilisent le vecteur de Pauli comme représentation d'entrée.

Les études PolSAR présentent une difficulté supplémentaire qui est la rareté des images SARs annotées. Pour cette raison, les ensembles d'apprentissage, de validation et de test sont souvent extraits de la même image bien qu'elle peut être volumineuse (10 1000 millions de pixels). Ceci peut induire trop de similitudes qui biaise l'erreur de généralisation et donne des résultats de test apparents plus élevés que ceux obtenus avec des vraies nouvelles données. Nous proposons donc une nouvelle méthode pour diviser l'ensemble de données afin de réduire cet effet et d'évaluer son impact.

Les difficultés d'implementation des modèles CVNN dans la pratique ont ralenti le développement du domaine [Mönning and Manandhar, 2018]. Pour combler ce lacune, nous avons mis en place une librairie *Python* permettant la mise en œuvre rapide de CVNN. Cette dernière est mise en ligne et bien documentée afin que la communauté puisse s'en servir. En effet, le besoin de la communauté pour cette librairie a fait l'objet du succès de ce code qui peut être vu par les métriques de *GitHub*, téléchargements, et citations. Additionnellement, nous réalisons une expérience qui justifie l'adaptation de l'initialisation pour les couches complexes définie par la référence [Trabelsi et al., 2017], et nous parvenons à montrer l'importance d'utiliser correctement cette adaptation.

Le chapitre 1 introduit la théorie des réseaux neurones classiques. Il présente l'histoire et le développement des réseaux de neurones jusqu'aux percées actuelles des algorithmes de pointe. Nous expliquons ensuite le fonctionnement de base d'un réseau de neurones et la phase de formation. Enfin, nous expliquons en détail chaque partie de l'algorithme du réseau de neurones et ses variantes. Les réseaux de neurones étant un domaine très vaste, nous limitons le chapitre aux avancées et théories pertinentes pour notre travail.

Le chapitre 2 explique l'adaptation de ces réseaux réels conventionnels au domaine complexe et introduit les bases de l'implémentation Complex-Valued Neural Network (CVNN). Nous proposons deux expériences qui prouvent l'importance de l'adaptation correcte de la technique d'initialisation au domaine complexe. En parallèle, nous décrivons la librarie publiée qui permet l'implémentation de CVNN en utilisant *Tensorflow* comme back-end. Nous terminons le chapitre par une expérience qui montre que les signaux temporels à valeurs réelles peuvent bénéficier d'un

CVNN lors de l'utilisation de la transformée de Hilbert pour obtenir de meilleures performances que lors de l'utilisation d'un modèle conventionnel à valeurs réelles.

Le chapitre 3 justifie les mérites de CVNN par rapport à RVNN des ensembles de données gaussiennes à valeurs complexes non circulaires, caractérisées par une corrélation entre les parties réelle et imaginaire ou une variance non égale pour la partie réelle et la partie imaginaire. Pour ce faire, nous générons aléatoirement plusieurs classes de vecteurs de données non circulaires qui ont la propriété de changer leur distribution si nous faisons tourner le générateur aléatoire, contrairement aux données circulaires, qui sont invariantes à la rotation. Classez-les en utilisant à la fois CVNN et RVNN, en effectuant plusieurs fois la simulation pour en déduire des résultats statistiques. Cette expérience a été réalisée dans différentes variantes, telles que différentes sources de non-circularité, différentes représentations d'entrée, différentes architectures de modèle, hyperparamètres et dimensions, etc., montrant que CVNN surpasse statistiquement RVNN en général et non pour un cas particulier seulement.

Le chapitre 4 explique la théorie de Synthetic Aperture Radar (SAR) afin de préparer le terrain pour le chapitre suivant, qui traite de ces images radar comme un ensemble de données d'entrée pour les tâches de classification et de segmentation sémantique.

Le chapitre 5 détaille les expériences et les résultats obtenus sur la classification et la segmentation PolSAR. Nous proposons d'abord un cadre permettant d'affirmer que la comparaison entre CVNN et RVNN est équitable et prouvons que CVNN est plus performant pour trois architectures de modèles différentes et deux images PolSAR. Nous soutenons ensuite que la représentation d'entrée utilisée dans la plupart des tâches de classification PolSAR peut être non optimale dans tous les cas et que les couches convolutives peuvent être plus performantes en utilisant une autre représentation. Nous étayons notre argument en effectuant les simulations correspondantes. Deuxièmement, nous montrons que la classification d'images PolSAR est un cas saturé et que cela est principalement dû à une corrélation entre les ensembles d'entraînement, de validation et de test, et nous proposons une méthode pour réduire cette corrélation et diminuer la saturation des applications *state-of-the-art*.

Nous clôturons ce rapport de thèse avec les conclusions du chapitre 6, et nous discutons des perspectives de travaux futurs.

Publications

Journaux

- Barrachina, J. A., Ren, C., Morisseau, C., Vieillard, G., and Ovarlez, J.-P. (2022d). Comparison Between Equivalent Architectures of Complex-Valued and Real-Valued Neural Networks - Application on Polarimetric SAR Image Segmentation. *Journal of Signal Processing Systems*, pages 1–10

Actes

- Barrachina, J. A., Ren, C., Vieillard, G., Morisseau, C., and Ovarlez, J.-P. (2022e). Real- and Complex-Valued Neural Networks for SAR image segmentation through different polarimetric representations. In *2022 IEEE International Conference on Image Processing (ICIP)*

- Barrachina, J. A., Ren, C., Morisseau, Vieillard, G., C., and Ovarlez, J.-P. (2022c). Merits of Complex-Valued Neural Networks for PolSAR image segmentation. In *GRETSI XXVIIIème Colloque Francophone de Traitement du Signal et des Images*

- **3MT finalist**. Barrachina, J. A., Ren, C., Morisseau, Vieillard, G., C., and Ovarlez, J.-P. (2022a). Complex-Valued Neural Networks for Polarimetric SAR segmentation using Pauli representation. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*

- **Ranked top 15% reviewer score**. Barrachina, J. A., Ren, C., Vieillard, G., Morisseau, C., and Ovarlez, J.-P. (2021c). About the Equivalence Between Complex-Valued and Real-Valued Fully Connected Neural Networks - Application to PolInSAR Images. In *2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6

- **r/MachineLearning (reddit) WAYR week 116**. Barrachina, J. A., Ren, C., Morisseau, C., Vieillard, G., and Ovarlez, J.-P. (2021a). Complex-Valued vs. Real-Valued Neural Networks for Classification Perspectives: An Example on Non-Circular Data. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2990–2994

Sans actes

- **Workshop presentation and publication**. Barrachina, J. A., Ren, C., Morisseau, Vieillard, G., C., and Ovarlez, J.-P. (2022b). Complex-Valued Neural Networks for Polarimetric SAR segmentation using Pauli representation. 5th SONDRA Workshop

- **Poster session**. Barrachina, J. A. (2020). A comparison between complex and real-valued fully connected neural networks on non-circular complex data. XXII Giambiagi Winter School: Artificial intelligence and deep learning in physics

- **Code**. Barrachina, J. A. (2021). NEGU93/cvnn. https://doi.org/10.5281/zenodo.4452131
  Barrachina, J. A. (2019). Complex-Valued Neural Networks (CVNN). https://github.com/NEGU93/cvnn

B - Mathematical Background

For further reading and properties of complex numbers, see Reference [Haykin, 2005].

B.1 . Complex Identities

Definition B.1.1 (Hermitian transpose)**.** Given $\boldsymbol{A} \in \mathbb{C}^{n\times m}, m, n \in \mathbb{N}^{+}$. The *Hermitian transpose* of a matrix is defined as:

$$\boldsymbol{A}^{H} = \overline{\boldsymbol{A}^{T}}\,. \tag{B.1}$$

The upper line on $\overline{A}$ refers to the conjugate value of a complex number, that is, a number with an equally real part and an imaginary part equal in magnitude but opposite in sign.

Definition B.1.2 (differential rule)**.**

$$\partial f = \frac{\partial f}{\partial z}\partial z + \frac{\partial f}{\partial \overline{z}}\partial \overline{z}\,. \tag{B.2}$$

Definition B.1.3 (conjugation rule)**.**

$$\begin{aligned} \overline{\left(\frac{\partial f}{\partial z}\right)} &= \frac{\partial \overline{f}}{\partial \overline{z}}\,, \\ \overline{\left(\frac{\partial f}{\partial \overline{z}}\right)} &= \overline{\frac{\partial f}{\partial z}}\,. \end{aligned} \tag{B.3}$$

When $f : \mathbb{C} \longrightarrow \mathbb{R}$ the expression can be simplified as:

$$\begin{aligned} \overline{\left(\frac{\partial f}{\partial z}\right)} &= \frac{\partial f}{\partial \overline{z}} \\ \overline{\left(\frac{\partial f}{\partial \overline{z}}\right)} &= \frac{\partial f}{\partial z}\,. \end{aligned} \tag{B.4}$$

Theorem B.1.1. Given $f(z) : \mathbb{C} \longrightarrow \mathbb{C}, z = x + i\,y \Rightarrow \exists u, v : \mathbb{R}^2 \longrightarrow \mathbb{R}$ such that $f(z) = u(x, y) + i\,v(x, y)$

B.1.1 . Complex differentiation rules

Given $f, g : \mathbb{C} \longrightarrow \mathbb{C}$ and $c \in \mathbb{C}$:

$$\begin{aligned} (f+g)'(z_0) &= f'(z_0) + g'(z_0)\,, \\ (f\,g)'(z_0) &= f'(z_0)\,g(z_0) + f(z_0)\,g'(z_0)\,, \\ \left(\frac{f}{g}\right)'(z_0) &= \frac{f'(z_0)\,g(z_0) - f(z_0)\,g'(z_0)}{g^2(z_0)}, \quad g(z_0) \neq 0\,, \\ (c\,f)'(z_0) &= c\,f'(z_0)\,. \end{aligned} \tag{B.5}$$

B.1.2 . Properties of the conjugate

Given $z, w \in \mathbb{C}$:

$$\begin{aligned}
\overline{z \pm w} &= \overline{z} \pm \overline{w}\,, \\
\overline{z\,w} &= \overline{z}\,\overline{w}\,, \\
\overline{\left(\frac{z}{w}\right)} &= \frac{\overline{z}}{\overline{w}}\,, \\
\overline{z^n} &= \overline{z}^n, \forall n \in \mathbb{C}\,, \\
|z|^2 &= z\,\overline{z}\,, \\
\overline{z}\,z &= z\,\overline{z}\,, \\
\overline{\overline{z}} &= z \text{ (involution)}\,, \\
\overline{z} &= z \Leftrightarrow z \in \mathbb{R}\,, \\
z^{-1} &= \frac{\overline{z}}{|z|^2}, \forall z \neq 0\,.
\end{aligned} \tag{B.6}$$

B.2 . Holomorphic Function

An *holomorphic* function is a complex-valued function of one or more complex variables that is, at every point of its domain, complex differentiable in a neighborhood of the point. This condition implies an *holomorphic* function is Class C^∞ (analytic).

Definition B.2.1. Given a complex function $f : \mathbb{C} \longrightarrow \mathbb{C}$ at a point z_0 of an open subset $\Omega \subset \mathbb{C}$ is *complex-differentiable* if exists a limit such as:

$$f'(z_0) = \lim_{z \to z_0} \frac{f(z) - f(z_0)}{z - z_0}\,. \tag{B.7}$$

As stated before, if a function is complex-differentiable at all points of Ω it is called *holomorphic* [Mönning and Manandhar, 2018]. The relationship between real differentiability and complex differentiability is stated on the *Cauchy-Riemann equations*

Theorem B.2.1 (Cauchy-Riemann equations)**.** Given $f(x + i\,y) = u(x, y) + i\,v(x, y)$ where $u, v : \mathbb{R}^2 \longrightarrow \mathbb{R}$ real-differentiable functions, f is complex-differentiable if satisfies:

$$\begin{aligned}
\frac{\partial u}{\partial x} &= \frac{\partial v}{\partial y}\,, \\
-\frac{\partial u}{\partial y} &= \frac{\partial v}{\partial x}\,.
\end{aligned} \tag{B.8}$$

Proof. Given $f : \mathbb{C} \longrightarrow \mathbb{C}$:

$$f(x + iy) = u(x, y) + i\,v(x, y)\,, \tag{B.9}$$

with $u, v : \mathbb{R} \longrightarrow \mathbb{R}$ real-differentiable functions.

For f to be complex-differentiable, then:

$$f'(z) = \lim_{\Delta x \to 0} \frac{f(z + \Delta x) - f(z)}{\Delta x} = \lim_{i\Delta y \to 0} \frac{f(z + i\,\Delta y) - f(z)}{i\,\Delta y}\,. \tag{B.10}$$

Replacing (B.10) with (B.9) and doing some algebra:

$$f'(z) = \frac{\partial u}{\partial x} + i\,\frac{\partial v}{\partial x} = \frac{\partial v}{\partial y} - i\,\frac{\partial u}{\partial y}\,. \tag{B.11}$$

By comparing real and imaginary parts from the latest function, the Cauchy-Riemann equations are evident. □

B.3 . Chain Rule

Theorem B.3.1 (real multivariate chain rule with complex variable)**.** Given $f : \mathbb{R}^2 \longrightarrow \mathbb{R}, z \in \mathbb{C}$ and $x(z), y(z) : \mathbb{C} \longrightarrow \mathbb{R}$

$$\frac{\partial f}{\partial z} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial z} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial z}\,. \tag{B.12}$$

This chain rule is analogous to that of the multivariate chain rule with real values. The need for stating this case arises from a demonstration that will be done for Theorem B.17.

Theorem B.3.2 (complex chain rule over real and imaginary part)**.** Given $f : \mathbb{C} \longrightarrow \mathbb{C}, z \in \mathbb{C}$ and $x(z), y(z) : \mathbb{C} \longrightarrow \mathbb{R}$

$$\frac{\partial f}{\partial z} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial z} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial z}\,. \tag{B.13}$$

Proof. By using theorem B.1.1, we can write $f = u + i\,v$ so that:

$$\frac{\partial f}{\partial z} = \frac{\partial u}{\partial z} + i\,\frac{\partial v}{\partial z}\,. \tag{B.14}$$

Using theorem B.3.1 we can apply the chain rule with u and v:

$$\frac{\partial f}{\partial z} = \left[\frac{\partial u}{\partial x}\frac{\partial x}{\partial z} + \frac{\partial u}{\partial y}\frac{\partial y}{\partial z}\right] + i\left[\frac{\partial v}{\partial x}\frac{\partial x}{\partial z} + \frac{\partial v}{\partial y}\frac{\partial y}{\partial z}\right]\,. \tag{B.15}$$

Rearranging the terms leads to:

$$\begin{aligned}
\frac{\partial f}{\partial z} &= \left[\frac{\partial u}{\partial x}\frac{\partial x}{\partial z} + i\,\frac{\partial v}{\partial x}\frac{\partial x}{\partial z}\right] + \left[\frac{\partial u}{\partial y}\frac{\partial y}{\partial z} + i\,\frac{\partial v}{\partial y}\frac{\partial y}{\partial z}\right]\,,\\
&= \left[\frac{\partial u}{\partial x} + i\,\frac{\partial v}{\partial x}\right]\frac{\partial x}{\partial z} + \left[\frac{\partial u}{\partial y} + i\,\frac{\partial v}{\partial y}\right]\frac{\partial y}{\partial z}\,,\\
&= \left[\frac{\partial u + i\,\partial v}{\partial x}\right]\frac{\partial x}{\partial z} + \left[\frac{\partial u + i\,\partial v}{\partial y}\right]\frac{\partial y}{\partial z}\,,\\
&= \left[\frac{\partial (u + i\,v)}{\partial x}\right]\frac{\partial x}{\partial z} + \left[\frac{\partial (u + i\,v)}{\partial y}\right]\frac{\partial y}{\partial z}\,,\\
&= \frac{\partial f}{\partial x}\frac{\partial x}{\partial z} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial z}\,.
\end{aligned} \tag{B.16}$$

□

Corollary B.3.2.1. See that the proof of Theorem B.3.1 is indistinct weather $z \in \mathbb{C}$ or $z \in \mathbb{R}$, so that both (B.3.1) and (B.3.2) are also valid for $z \in \mathbb{R}$

Definition B.3.1 (complex chain rule)**.** Given $h, g : \mathbb{C} \longrightarrow \mathbb{C}, z \in \mathbb{C}$. The chain rule in complex numbers is now given by:

$$\begin{aligned} \frac{\partial h(g)}{\partial z} &= \frac{\partial h}{\partial g}\frac{\partial g}{\partial z} + \frac{\partial h}{\partial \overline{g}}\frac{\partial \overline{g}}{\partial z} \\ \frac{\partial h(g)}{\partial \overline{z}} &= \frac{\partial h}{\partial g}\frac{\partial g}{\partial \overline{z}} + \frac{\partial h}{\partial \overline{g}}\frac{\partial \overline{g}}{\partial \overline{z}} \,. \end{aligned} \tag{B.17}$$

Proof. As we make no assumption of h or g being *holomorphic*, we will use Wirtinger calculus (B.33) (having in mind that *holomorphic* functions are special cases of the later):

$$\frac{\partial\,(f \circ g)}{\partial z} = \frac{1}{2}\left(\frac{\partial\,(f \circ g)}{\partial z_{\mathrm{Re}}} - i\,\frac{\partial\,(f \circ g)}{\partial z_{\mathrm{Im}}}\right) . \tag{B.18}$$

Using (B.3.2.1):

$$\frac{\partial\,(f \circ g)}{\partial z} = \frac{1}{2}\left(\left(\frac{\partial f}{\partial g_{\mathrm{Re}}}\frac{\partial g_{\mathrm{Re}}}{\partial z_{\mathrm{Re}}} + \frac{\partial f}{\partial g_{\mathrm{Im}}}\frac{\partial g_{\mathrm{Im}}}{\partial z_{\mathrm{Re}}}\right) - i\left(\frac{\partial f}{\partial g_{\mathrm{Re}}}\frac{\partial g_{\mathrm{Re}}}{\partial z_{\mathrm{Im}}} + \frac{\partial f}{\partial g_{\mathrm{Im}}}\frac{\partial g_{\mathrm{Im}}}{\partial z_{\mathrm{Im}}}\right)\right) , \tag{B.19}$$

$$\begin{aligned} &= \frac{1}{4}\left(\frac{\partial f}{\partial g_{\mathrm{Re}}}\frac{\partial (g+\overline{g})}{\partial z_{\mathrm{Re}}} - i\,\frac{\partial f}{\partial g_{\mathrm{Im}}}\frac{\partial (g-\overline{g})}{\partial z_{\mathrm{Re}}}\right) \\ &\qquad - i\,\frac{1}{4}\left(\frac{\partial f}{\partial g_{\mathrm{Re}}}\frac{\partial (g+\overline{g})}{\partial z_{\mathrm{Im}}} - i\,\frac{\partial f}{\partial g_{\mathrm{Im}}}\frac{\partial (g-\overline{g})}{\partial z_{\mathrm{Im}}}\right) \end{aligned} \tag{B.20}$$

$$\begin{aligned} &= \frac{1}{4}\frac{\partial f}{\partial g_{\mathrm{Re}}}\left(\frac{\partial (g+\overline{g})}{\partial z_{\mathrm{Re}}} - i\,\frac{\partial (g+\overline{g})}{\partial z_{\mathrm{Im}}}\right) \\ &\qquad - \frac{1}{4}\frac{\partial f}{\partial g_{\mathrm{Im}}}\left(i\,\frac{\partial (g-\overline{g})}{\partial z_{\mathrm{Re}}} + \frac{\partial (g-\overline{g})}{\partial z_{\mathrm{Im}}}\right) . \end{aligned} \tag{B.21}$$

Using Wirtinger calculus definition again (Equation B.33):

$$\frac{\partial\,(f \circ g)}{\partial z} = \frac{1}{2}\left(\frac{\partial f}{\partial g_{\mathrm{Re}}}\frac{\partial (g+\overline{g})}{\partial z} - i\frac{\partial f}{\partial g_{\mathrm{Im}}}\frac{\partial (g-\overline{g})}{\partial z}\right) . \tag{B.22}$$

Using (B.5.1.1):

$$\begin{aligned}
\frac{\partial\left(f\circ g\right)}{\partial z} &= \frac{1}{2}\left(\left(\frac{\partial f}{\partial g}+\frac{\partial f}{\partial \overline{g}}\right)\frac{\partial(g+\overline{g})}{\partial z}+\left(\frac{\partial f}{\partial g}-\frac{\partial f}{\partial \overline{g}}\right)\frac{\partial(g-\overline{g})}{\partial z}\right)\,,\\
&= \frac{1}{2}\left(\frac{\partial f}{\partial g}\left(\frac{\partial(g+\overline{g})}{\partial z}+\frac{\partial(g-\overline{g})}{\partial z}\right)+\frac{\partial f}{\partial \overline{g}}\left(\frac{\partial(g+\overline{g})}{\partial z}-\frac{\partial(g-\overline{g})}{\partial z}\right)\right)\,,\\
&= \frac{1}{2}\left(\frac{\partial f}{\partial g}\left(\frac{\partial(g+\overline{g})+\partial(g-\overline{g})}{\partial z}\right)+\frac{\partial f}{\partial \overline{g}}\left(\frac{\partial(g+\overline{g})-\partial(g-\overline{g})}{\partial z}\right)\right)\,,\\
&= \frac{1}{2}\left(\frac{\partial f}{\partial g}\left(\frac{\partial(g+\overline{g}+g-\overline{g})}{\partial z}\right)+\frac{\partial f}{\partial \overline{g}}\left(\frac{\partial(g+\overline{g}-g+\overline{g})}{\partial z}\right)\right)\,,\\
&= \left(\frac{\partial f}{\partial g}\frac{\partial g}{\partial z}+\frac{\partial f}{\partial \overline{g}}\frac{\partial \overline{g}}{\partial z}\right)\,.
\end{aligned} \tag{B.23}$$

□

B.4 . Liouville Theorem

Liouville graduated from the École Polytechnique in 1827. After some years as an assistant at various institutions including the École Centrale Paris, he was appointed as a professor at the École Polytechnique in 1838.

Definition B.4.1. In complex analysis, an *entire function*, also called an *integral function*, is a complex-valued function that is *holomorphic* at all finite points over the whole complex plane.

Definition B.4.2. Given a function f, the function is *bounded* if $\exists M \in \mathbb{R}^+ : |f(z)| < M$.

Theorem B.4.1 (Cauchy integral theorem)**.** Given f analytic through region **D**, then the contour integral of $f(z)$ along any close path C inside region **D** is zero:

$$\oint_C f(z)dz = 0\,. \tag{B.24}$$

Theorem B.4.2 (Cauchy integral formula)**.** Given f analytic on the boundary C with z_0 any point inside C, then:

$$f(z_0) = \frac{1}{2\pi i}\oint_C \frac{f(z)}{z-z_0}\,dz\,, \tag{B.25}$$

where the contour integration is taken in the counterclockwise direction.

Corollary B.4.2.1.

$$f^{(n)}(z_0) = \frac{n!}{2\pi i}\oint_C \frac{f(z)}{(z-z_0)^{n+1}}\,dz\,. \tag{B.26}$$

In complex analysis, Liouville's theorem states that every *bounded entire function* must be constant. That is:

Theorem B.4.3 (Liouville's Theorem)**.**

$$f : \mathbb{C} \longrightarrow \mathbb{C} \text{ holomorphic } / \exists M \in \mathbb{R}^+ : |f(z)| < M, \forall z \in \mathbb{C} \Rightarrow f = c, c \in \mathbb{C} \,. \tag{B.27}$$

Equivalently, non-constant *holomorphic* functions on $\mathbb{C}$ have unbounded images.

Proof. Because every *holomorphic* function is analytic (Class C^∞), that is, it can be represented as a Taylor series, we can therefore write it as:

$$f(z) = \sum_{k=0}^{\infty} a_k \, z^k \,. \tag{B.28}$$

As f is *holomorphic* in the open region enclosed by the path and continuous on its closure. Because of B.4.2.1 we have:

$$a_k = \frac{f^{(k)}(0)}{k!} = \frac{1}{2\pi i} \oint_{C_r} \frac{f(z)}{z^{k+1} dz} \,, \tag{B.29}$$

where C_r is a circle of radius $r > 0$. Because f is bounded:

$$|a_k| \leqslant \frac{1}{2\pi} \oint_{C_r} \frac{|f(z)|}{|z|^{k+1}} |dz| \leqslant \frac{1}{2\pi} \oint_{C_r} \frac{M}{r^{k+1}} |dz| = \frac{M}{2\pi r^{k+1}} \oint_{C_r} |dz| \,, \tag{B.30}$$

$$= \frac{M}{2\pi r^{k+1}} 2\pi \,, \tag{B.31}$$

$$= \frac{M}{r^k} \,. \tag{B.32}$$

The latest derivation is also known as *Cauchy's inequality*.

As r is any positive real number, by taking the $r \to \infty$ then $a_k \to 0$ for all $k \neq 0$. Therefore from Equation B.28 we have that $f(z) = a_0$. □

Liouville's theorem implications were considered to be a big problem around 1990 as some researchers believed indifferentiability should lead to the impossibility to obtain and/or analyze the dynamics of the CVNNs [Hirose, 2013].

B.5 . Wirtinger Calculus

Wirtinger calculus, named after Wilhelm Wirtinger (1927) [Wirtinger, 1927], generalizes the notion of complex derivative, and the *holomorphic* functions become a special case only. Further reading on Wirtinger calculus can be found in [Kreutz-Delgado, 2009, Fischer, 2005].

Theorem B.5.1 (Wirtinger Calculus)**.** Given a complex function $f(z)$ of a complex variable $z = x + i\,y \in \mathbb{C}, x, y \in \mathbb{R}$.

The partial derivatives with respect to z and $\overline{z}$ respectively are defined as:

$$\begin{aligned} \frac{\partial f}{\partial z} &\triangleq \frac{1}{2}\left(\frac{\partial f}{\partial x} - i\,\frac{\partial f}{\partial y}\right), \\ \frac{\partial f}{\partial \overline{z}} &\triangleq \frac{1}{2}\left(\frac{\partial f}{\partial x} + i\,\frac{\partial f}{\partial y}\right). \end{aligned} \tag{B.33}$$

These derivatives are called $\mathbb{R}$-derivative and conjugate $\mathbb{R}$-derivative, respectively. As said before, the *holomorphic* case is only a special case where the function can be considered as $f(z, \overline{z}), \overline{z} = 0$. Wirtinger calculus enables to work with *non-holomorphic* functions, providing an alternative method for computing the gradient that also improves the stability of the training process.

Proof. Defining $z = x + i\,y$ one can also define $x(z, \overline{z}), y(z, \overline{z}) : \mathbb{C} \longrightarrow \mathbb{R}$ as follows:

$$\begin{aligned} x &= \frac{1}{2}\,(z + \overline{z})\,, \\ y &= \frac{1}{2\,i}\,(z - \overline{z})\,. \end{aligned} \tag{B.34}$$

Using (B.3.2):

$$\begin{aligned} \frac{\partial f}{\partial z} &= \frac{\partial f}{\partial x}\frac{\partial x}{\partial z} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial z}, \\ &= \frac{\partial f}{\partial x}\frac{1}{2} + \frac{\partial f}{\partial y}\left(\frac{-i}{2}\right), \\ &= \frac{1}{2}\left(\frac{\partial f}{\partial x} - i\,\frac{\partial f}{\partial y}\right). \end{aligned} \tag{B.35}$$

□

Corollary B.5.1.1.

$$\begin{aligned} \frac{\partial f}{\partial g} + \frac{\partial f}{\partial \overline{g}} &= \frac{1}{2}\left(\frac{\partial f}{\partial g_{\mathrm{Re}}} - i\,\frac{\partial f}{\partial g_{\mathrm{Im}}}\right) + \frac{1}{2}\left(\frac{\partial f}{\partial g_{\mathrm{Re}}} + i\,\frac{\partial f}{\partial g_{\mathrm{Im}}}\right), \\ &= \frac{\partial f}{\partial g_{\mathrm{Re}}} \\ i\left(\frac{\partial f}{\partial g} - \frac{\partial f}{\partial \overline{g}}\right) &= \frac{i}{2}\left(\frac{\partial f}{\partial g_{\mathrm{Re}}} - i\,\frac{\partial f}{\partial g_{\mathrm{Im}}}\right) - \frac{i}{2}\left(\frac{\partial f}{\partial g_{\mathrm{Re}}} + i\,\frac{\partial f}{\partial g_{\mathrm{Im}}}\right), \\ &= \frac{i}{2}\left(-2i\,\frac{\partial f}{\partial g_{\mathrm{Im}}}\right), \\ &= \frac{\partial f}{\partial g_{\mathrm{Im}}}, \end{aligned} \tag{B.36}$$

where g_{Re} and g_{Im} are the real and imaginary part of g respectively.

Theorem B.5.2. Given $f : \mathbb{C} \longrightarrow \mathbb{C}$ *holomorphic* with $f(x+iy) = u(x,y) + iv(x,y)$ where $u, v : \mathbb{R} \longrightarrow \mathbb{R}$ real-differentiable functions. Then

$$\frac{\partial f}{\partial \overline{z}} = 0\,.$$

Proof. Using Wirtinger calculus (Section B.5).

$$\frac{\partial f}{\partial \overline{z}} = \frac{1}{2}\left(\frac{\partial f}{\partial x} + i\,\frac{\partial f}{\partial y}\right)\,. \tag{B.37}$$

By definition, then:

$$\begin{aligned}\frac{\partial f}{\partial \overline{\cdot}} &= \frac{1}{2}\left(\frac{\partial f}{\partial x} + i\,\frac{\partial f}{\partial y}\right)\,,\\ &= \frac{1}{2}\left(\left(\frac{\partial u}{\partial x} + i\,\frac{\partial v}{\partial x}\right) + i\,\left(\frac{\partial u}{\partial y} + i\,\frac{\partial v}{\partial y}\right)\right)\,,\\ &= \frac{1}{2}\left(\left(\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right) + i\,\left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right)\right)\,.\end{aligned} \tag{B.38}$$

Because f is *holomorphic* then the Cauchy-Riemann equations (Theorem B.2.1) applies making Equation B.38 equal to zero. □

Even though so far we have always talked about general chain rule definitions. Here we will demonstrate a particularly interesting chain rule used when working with neural networks. For this optimization technique, the cost function to optimize is always real, even if its variables are not. Therefore, the following chain rule will have an application interest.

Theorem B.5.3 (complex chain rule with real output)**.** Given $f : \mathbb{C} \to \mathbb{R}$, $g : \mathbb{C} \to \mathbb{C}$ with $g(z) = r(z) + i\,s(z)$, $z = x + i\,y \in \mathbb{C}$:

$$\frac{\partial f}{\partial z} = \frac{\partial f}{\partial r}\frac{\partial r}{\partial z} + \frac{\partial f}{\partial s}\frac{\partial s}{\partial z}\,. \tag{B.39}$$

Proof. For this proof, we will assume we are already working with Wirtinger Calculus for the partial derivative definition.

$$\begin{aligned}\frac{\partial f}{\partial z} &= \frac{\partial f}{\partial g}\frac{\partial g}{\partial z} + \frac{\partial f}{\partial \overline{g}}\frac{\partial \overline{g}}{\partial z}\,,\\ &= \frac{1}{4}\left(\frac{\partial f}{\partial r} - i\,\frac{\partial f}{\partial s}\right)\left(\frac{\partial g}{\partial x} - i\,\frac{\partial g}{\partial y}\right) + \frac{1}{4}\left(\frac{\partial f}{\partial r} + i\,\frac{\partial f}{\partial s}\right)\overline{\left(\frac{\partial g}{\partial x} + i\,\frac{\partial g}{\partial y}\right)}\,,\\ &= \frac{1}{4}\left(\frac{\partial f}{\partial r} - i\,\frac{\partial f}{\partial s}\right)\left[\left(\frac{\partial r}{\partial x} + i\,\frac{\partial s}{\partial x}\right) - i\,\left(\frac{\partial r}{\partial y} + i\,\frac{\partial s}{\partial y}\right)\right] + \cdots\\ &\cdots + \frac{1}{4}\left(\frac{\partial f}{\partial r} + i\,\frac{\partial f}{\partial s}\right)\overline{\left[\left(\frac{\partial r}{\partial x} + i\,\frac{\partial s}{\partial x}\right) + i\,\left(\frac{\partial r}{\partial y} + i\,\frac{\partial s}{\partial y}\right)\right]}\,.\end{aligned} \tag{B.40}$$

□

B.6 . Neural Network-applied Chain Rule

B.6.1 . Real Valued Backpropagation

To learn how to optimize the weight of $\omega_{ij}^{(l)}$, it is necessary to find the partial derivative of the loss function for a given weight. We will use the chain rule as follows:

$$\frac{\partial \mathcal{L}}{\partial \omega_{ij}^{(l)}} = \sum_{n=1}^{N_L} \frac{\partial e_n}{\partial X_n^{(L)}} \frac{\partial X_n^{(L)}}{\partial V_n^{(L)}} \frac{\partial V_n^{(L)}}{\partial \omega_{ij}^{(l)}} \,. \tag{B.41}$$

Given these three terms inside the addition, we could find the value we are looking for. The first two partial derivatives are trivial as $\partial e_n(d_n, y_n)/\partial y_n$ exists and is no zero and $\partial X_n^{(L)}/\partial V_n^{(L)}$ is the derivate of σ.

With regard to $\partial V_n^{(L)}/\partial \omega_{ij}^{(l)}$, when the layer is the same for both values ($l = L$), the definition is trivial and the following result is obtained:

$$\begin{aligned} \frac{\partial V_i^{(l)}}{\partial \omega_{ij}^{(l)}} &= \frac{\partial \left(\sum_{\mathrm{j}} \omega_{\mathrm{ij}}^{(l)} X_{\mathrm{j}}^{(l-1)} \right)}{\partial \omega_{ij}^{(l)}} = \sum_{\mathrm{j}} \frac{\partial \left(\omega_{\mathrm{ij}}^{(l)} X_{\mathrm{j}}^{(l-1)} \right)}{\partial \omega_{ij}^{(l)}} \,, \\ &= \begin{cases} 0 & \mathrm{j} \neq j \\ X_j^{(l-1)} & \mathrm{j} = j \end{cases} \,, \\ &= X_j^{(l-1)} \,. \end{aligned} \tag{B.42}$$

Note the subtle difference between j and j. We will now define the cases where the weight and V_n are not from the same layer.

For given $h, l \in [0, L]$, with $h \leq l - 2$ we can define the derivative as follows:

$$\begin{aligned} \frac{\partial V_n^{(l)}}{\partial \omega_{jk}^{(h)}} &= \frac{\partial \left(\sum_i \omega_{ni}^{(l)} X_i^{(l-1)} \right)}{\partial \omega_{jk}^{(h)}} \,, \\ &= \sum_i^{N_{l-1}} \omega_{ni}^{(l)} \frac{\partial X_i^{(l-1)}}{\partial \omega_{jk}^{(h)}} \,, \\ &= \sum_i^{N_{l-1}} \omega_{ni}^{(l)} \frac{\partial X_i^{(l-1)}}{\partial V_i^{(l-1)}} \frac{\partial V_i^{(l-1)}}{\partial \omega_{jk}^{(h)}} \,. \end{aligned} \tag{B.43}$$

Using Equations B.42 and B.43, we have all cases except for $h = l - 1$ which

will be:

$$\begin{aligned}\frac{\partial V_n^{(l)}}{\partial \omega_{jk}^{(l-1)}} &= \frac{\partial \left(\sum_{\mathrm{j}} \omega_{n\mathrm{j}}^{(l)} X_{\mathrm{j}}^{(l-1)} \right)}{\partial \omega_{jk}^{(l-1)}} \,, \\ &= \sum_{\mathrm{j}}^{N_{l-1}} \omega_{n\mathrm{j}}^{(l)} \frac{\partial X_{\mathrm{j}}^{(l-1)}}{\partial \omega_{jk}^{(l-1)}} \,, \\ &= \omega_{nj}^{(l)} \frac{\partial X_j^{(l-1)}}{\partial V_j^{(l-1)}} \frac{\partial V_j^{(l-1)}}{\partial \omega_{jk}^{(l-1)}} \,.\end{aligned} \tag{B.44}$$

Using the result from (B.42) we can get a final result for (B.44). To sum up, the derivative can be written as follows:

$$\frac{\partial V_n^{(l)}}{\partial \omega_{jk}^{(h)}} = \begin{cases} X_j^{(l-1)} & h = l \,, \\ \omega_{nj}^{(l)} \dfrac{\partial X_j^{(l-1)}}{\partial V_j^{(l-1)}} \dfrac{\partial V_j^{(l-1)}}{\partial \omega_{jk}^{(l-1)}} & h = l-1 \,, \\ \displaystyle\sum_i^{N_{l-1}} \omega_{ni}^{(l)} \frac{\partial X_i^{(l-1)}}{\partial V_i^{(l-1)}} \frac{\partial V_i^{(l-1)}}{\partial \omega_{jk}^{(h)}} & h \leq l-2 \,. \end{cases} \tag{B.45}$$

Equation B.41 can then be solved applying (B.45) iteratively to reduce the exponent L to the desired value l. Note that $l \leq L$ and $L > 0$.

Benvenuto and Piazza definition

In Reference [Benvenuto and Piazza, 1992], another recursive definition for the backpropagation algorithm is defined:

$$e_n^{(l)} = \begin{cases} e_n & l = L \,, \\ \displaystyle\sum_{q=1}^{N_{l+1}} \omega_{qn}^{(l+1)} \delta_q^{(l+1)} & l < L \,, \end{cases} \tag{B.46}$$

with $\delta_n^{(l)} = e_n^{(l)} \sigma'(V_n^{(l)})$. Then the derivation is defined recursively as:

$$\frac{\partial \mathcal{L}}{\partial \omega_{ij}^{(l)}} = \sum_n^{N_L} \delta_n^{(l)} X_m^{(l-1)} \,. \tag{B.47}$$

It can be proven that (B.41) and (B.47) are equivalent.

B.6.2 . Complex-Valued Backpropagation

The analysis in the complex case is analogous to that made in the real-valued backpropagation (Section B.6.1). The only change is that now $\sigma : \mathbb{C} \longrightarrow \mathbb{C}$,

$e_n : \mathbb{C} \longrightarrow \mathbb{R}$ and $\omega_{ij}^{(l)}, X_n^{(l)}, V_n^{(l)}, e_n^{(l)} \in \mathbb{C}$.

Now, the chain rule is changed using (B.17) so Equation (B.41) changes to:

$$\frac{\partial e}{\partial \omega} = \frac{\partial e}{\partial X}\frac{\partial X}{\partial V}\frac{\partial V}{\partial \omega} + \frac{\partial e}{\partial X}\frac{\partial X}{\partial \overline{V}}\frac{\partial \overline{V}}{\partial \omega} + \frac{\partial e}{\partial \overline{X}}\frac{\partial \overline{X}}{\partial V}\frac{\partial V}{\partial \omega} + \frac{\partial e}{\partial \overline{X}}\frac{\overline{X}}{\partial \overline{V}}\frac{\partial \overline{V}}{\partial \omega}\,. \tag{B.48}$$

Note that we have used the upper line to denote the conjugate for clarity. All subindexes have been removed for clarity but they stand the same as in Equation B.41.

As $e_n : \mathbb{C} \longrightarrow \mathbb{R}$ then using the conjugation rule (Definition B.1.3):

$$\begin{aligned}
\frac{\partial e}{\partial \overline{X}} &= \overline{\left(\frac{\partial e}{\partial X}\right)}\,,\\
\frac{\partial \overline{X}}{\partial \overline{V}} &= \overline{\left(\frac{\partial X}{\partial V}\right)}\,,\\
\frac{\partial X}{\partial \overline{V}} &= \overline{\left(\frac{\partial \overline{X}}{\partial V}\right)}\,,
\end{aligned} \tag{B.49}$$

so that not all the partial derivatives must be calculated.

Focusing our attention on the derivative $\partial V/\partial \omega$, a differentiation between layer difference of V and ω will be made in this approach. The simplest is the one where the layer from $V_n^{(l)}$ is the same as the weight. Regardless of the complex domain, $V_n^{(l)}$ is still equal to $\sum_i^{N_{l-1}} \omega_{ni}^{(l)} X_i^{(l-1)}$ for what the value of the derivative remains unchanged. For the wights (ω) of the previous layer, the derivative is as follows:

$$\begin{aligned}
\frac{\partial V_n^{(l)}}{\partial \omega_{jk}^{(l-1)}} &= \frac{\partial\left(\sum_i^{N_{l-1}} \omega_{n\mathrm{j}}^{(l)} X_{\mathrm{j}}^{(l-1)}\right)}{\partial \omega_{jk}^{(l-1)}}\,,\\
&= \omega_{nj}^{(l)} \frac{\partial X_j^{(l-1)}}{\partial \omega_{jk}^{(l-1)}}\,,\\
&= \omega_{nj}^{(l)} \left[\frac{\partial X_j^{(l-1)}}{\partial V_j^{(l-1)}}\frac{\partial V_j^{(l-1)}}{\partial \omega_{jk}^{(l-1)}} + \frac{\partial X_j^{(l-1)}}{\partial \overline{V}_j^{(l-1)}}\frac{\partial \overline{V}_j^{(l-1)}}{\partial \omega_{jk}^{(l-1)}}\right]\,.
\end{aligned} \tag{B.50}$$

Now by definition, $V_n^{(l)}$ is analytic because of being a polynomial series and therefore is holomorphic as well. Using Theorem B.5.2, $\partial \overline{V}_j^{(l-1)}/\partial \omega_{jk}^{(l-1)} = 0$. The second term could be removed. Therefore the equation is simplified to the following:

$$\frac{\partial V_n^{(l)}}{\partial \omega_{jk}^{(l-1)}} = \omega_{nj}^{(l)} \frac{\partial X_j^{(l-1)}}{\partial V_j^{(l-1)}}\frac{\partial V_j^{(l-1)}}{\partial \omega_{jk}^{(l-1)}} = \omega_{nj}^{(l)} \frac{\partial X_j^{(l-1)}}{\partial V_j^{(l-1)}} X_k^{(l-2)}\,. \tag{B.51}$$

For the rest of the cases where the layers are farther apart, the equation is as follows:

$$\begin{aligned}\frac{\partial V_n^{(l)}}{\partial \omega_{jk}^{(h)}} &= \frac{\partial \left(\sum_i^{N_{l-1}} \omega_{ni}^{(l)} X_i^{(l-1)} \right)}{\partial \omega_{jk}^{(l-1)}}, \\ &= \sum_i^{N_{l-1}} \omega_{nj}^{(l)} \left[\frac{\partial X_i^{(l-1)}}{\partial V_i^{(l-1)}} \frac{\partial V_i^{(l-1)}}{\partial \omega_{jk}^{(h)}} + \frac{\partial X_i^{(l-1)}}{\partial \overline{V}_i^{(l-1)}} \frac{\partial \overline{V}_i^{(l-1)}}{\partial \omega_{jk}^{(h)}} \right],\end{aligned} \tag{B.52}$$

where $h \leq l-2$. Therefore, based on Equations B.52 and B.51, the final equation remains as follows:

$$\frac{\partial V_n^{(l)}}{\partial \omega_{jk}^{(h)}} = \begin{cases} X_j^{(l-1)} & h = l\,, \\ \omega_{nj}^{(l)} \frac{\partial X_j^{(l-1)}}{\partial V_j^{(l-1)}} \frac{\partial V_j^{(l-1)}}{\partial \omega_{jk}^{(l-1)}} & h = l-1\,, \\ \sum_i^{N_{l-1}} \omega_{nj}^{(l)} \left[\frac{\partial X_i^{(l-1)}}{\partial V_i^{(l-1)}} \frac{\partial V_i^{(l-1)}}{\partial \omega_{jk}^{(h)}} + \frac{\partial X_i^{(l-1)}}{\partial \overline{V}_i^{(l-1)}} \frac{\partial \overline{V}_i^{(l-1)}}{\partial \omega_{jk}^{(h)}} \right] & h \leq l-2\,. \end{cases} \tag{B.53}$$

Using the property that $\partial \overline{V}/\partial \omega = \overline{(\partial V/\partial \overline{\omega})}$ and the distributed properties of the conjugate, the following equation can be derived:

$$\frac{\partial \overline{V}_n^{(l)}}{\partial \omega_{jk}^{(h)}} = \begin{cases} 0 & h = l\,, \\ \overline{\omega}_{nj}^{(l)} \frac{\partial \overline{X}_j^{(l-1)}}{\partial V_j^{(l-1)}} \frac{\partial V_j^{(l-1)}}{\partial \omega_{jk}^{(l-1)}} & h = l-1\,, \\ \sum_i^{N_{l-1}} \overline{\omega}_{nj}^{(l)} \left[\frac{\partial \overline{X}_i^{(l-1)}}{\partial V_i^{(l-1)}} \frac{\partial V_i^{(l-1)}}{\partial \omega_{jk}^{(h)}} + \frac{\partial \overline{X}_i^{(l-1)}}{\partial \overline{V}_i^{(l-1)}} \frac{\partial \overline{V}_i^{(l-1)}}{\partial \omega_{jk}^{(h)}} \right] & h \leq l-2\,. \end{cases} \tag{B.54}$$

Using Equations B.53 and B.54, we can calculate all possible values of $\partial V/\partial \omega$ and $\partial \overline{V}/\partial \omega$. Once defined the loss and the activation function, using Equation B.48, backpropagation can be made also in the complex plane.

Hänsch and Hellwich definition

Ronny Hänsch and Olaf Helllwich [Hänsch and Hellwich, 2009b] made a similar approach for the general equations of complex neural networks by using the complex chain rule. Using (B.48), they define X and V from the same layer as the weight instead of e.

By doing so, and using the fact that $\partial \overline{V}_j^{(l-1)}/\partial \omega_{jk}^{(l-1)} = 0$, two terms are deleted. In conjunction with the complex equivalent of Equation B.42, Equation

B.48 is simplified to:

$$\begin{aligned}\frac{\partial e_n}{\partial \omega_{ji}^{(l)}} &= \frac{\partial e_n}{\partial X_i^{(l)}} \frac{\partial X_i^{(l)}}{\partial V_i^{(l)}} \frac{\partial V_i^{(l)}}{\partial \omega_{ji}^{(l)}} + \frac{\partial e_n}{\partial \overline{X}_i^{(l)}} \frac{\partial \overline{X}_i^{(l)}}{\partial V_i^{(l)}} \frac{\partial V_i^{(l)}}{\partial \omega_{ji}^{(l)}}\,, \\ &= \frac{\partial e_n}{\partial X_i^{(l)}} \frac{\partial X_i^{(l)}}{\partial V_i^{(l)}} X_j^{(l-1)} + \frac{\partial e_n}{\partial \overline{X}_i^{(l)}} \frac{\partial \overline{X}_i^{(l)}}{\partial V_i^{(l)}} X_j^{(l-1)}\,.\end{aligned} \tag{B.55}$$

Now instead of making the analysis for $\partial V/\partial \omega$, the equivalent analysis will be made for $\partial e/\partial X$. The case with $l = L$ is the trivial one when its value depends on the chosen error function. For $l = L - 1$, the following equality applies:

$$\frac{\partial e_n}{\partial X_i^{(L-1)}} = \frac{\partial e_n}{\partial V_n^L} \frac{\partial V_n^L}{\partial X_i^{(L-1)}} + \frac{\partial e_n}{\partial \overline{V}_n^L} \frac{\partial \overline{V}_n^L}{\partial X_i^{(L-1)}}\,. \tag{B.56}$$

However, as $V_n^{(L)} = \sum_i \omega_{ni}^{(L)} X_i^{(L-1)}$, its derivatives are:

$$\begin{aligned}\frac{\partial V_n^{(l+1)}}{\partial X_i^{(l)}} &= \omega_{ni}^{(l+1)}\,, \\ \frac{\partial \overline{V}_n^{(l+1)}}{\partial X_i^{(l)}} &= 0\,.\end{aligned} \tag{B.57}$$

For that reason, the second term of Equation B.56 is deleted, and using the chain rule again we have:

$$\begin{aligned}\frac{\partial e_n}{\partial X_i^{(L-1)}} &= \frac{\partial e_n}{\partial V_n^{(L)}} \frac{\partial V_n^{(L)}}{\partial X_i^{(L-1)}}\,, \\ &= \frac{\partial e_n}{\partial X_n^{(L)}} \frac{\partial X_n^{(L)}}{\partial V_n^{(L)}} \frac{\partial V_n^{(L)}}{\partial X_i^{(L-1)}} + \frac{\partial e_n}{\partial \overline{X}_n^{L}} \frac{\partial \overline{X}_n^{(L)}}{\partial V_n^{(L)}} \frac{\partial V_n^{(L)}}{\partial X_i^{(L-1)}}\,, \\ &= \frac{\partial e_n}{\partial X_n^{(L)}} \frac{\partial X_n^{(L)}}{\partial V_n^{(L)}} \omega_{in}^{(L)} + \frac{\partial e_n}{\partial \overline{X}_n^{(L)}} \frac{\partial \overline{X}_n^{(L)}}{\partial V_n^{(L)}} \omega_{in}^{(L)}\,,\end{aligned} \tag{B.58}$$

where the partial derivatives depend on the definition of the loss and activation function.

For the general case of $l \leq L-2$ the derivation is similar to the previous one:

$$\begin{aligned}
\frac{\partial e_n}{\partial X_i^{(l)}} &= \sum_k \frac{\partial e_n}{\partial V_k^{l+1}} \frac{\partial V_k^{l+1}}{\partial X_i^{(l)}} + \frac{\partial e_n}{\partial \overline{V}_k^{l+1}} \frac{\partial \overline{V}_k^{l+1}}{\partial X_i^{(l)}}\,, \\
&= \sum_k \frac{\partial e_n}{\partial V_k^{l+1}} \frac{\partial V_k^{l+1}}{\partial X_i^{(l)}}\,, \\
&= \sum_k \frac{\partial e_n}{\partial V_k^{l+1}} \omega_{ik}^{(l+1)}\,, \\
&= \sum_k \frac{\partial e_n}{\partial X_k^{(l+1)}} \frac{\partial X_k^{(l+1)}}{\partial V_k^{(l+1)}} \omega_{ik}^{(l+1)} + \frac{\partial e_n}{\partial \overline{X}_k^{(l+1)}} \frac{\partial \overline{X}_k^{(l+1)}}{\partial V_k^{(l+1)}} \omega_{ik}^{(l+1)}\,.
\end{aligned} \tag{B.59}$$

Therefore, $\partial e / \partial X$ can be defined as:

$$\frac{\partial e_n}{\partial X_i^{(l)}} = \begin{cases} \frac{\partial e_n}{\partial X_n^{(L)}}\,, & l = L\,, \\ \frac{\partial e_n}{\partial X_n^{(L)}} \frac{\partial X_n^{(L)}}{\partial V_n^{(L)}} \omega_{in}^{(L)} + \frac{\partial e_n}{\partial \overline{X}_n^{(L)}} \frac{\partial \overline{X}_n^{(L)}}{\partial V_n^{(L)}} \omega_{in}^{(L)}\,, & l = L-1\,, \\ \displaystyle\sum_k \left(\frac{\partial e_n}{\partial X_k^{(l+1)}} \frac{\partial X_k^{(l+1)}}{\partial V_k^{(l+1)}} \omega_{ik}^{(l+1)} + \frac{\partial e_n}{\partial \overline{X}_k^{(l+1)}} \frac{\partial \overline{X}_k^{(l+1)}}{\partial V_k^{(l+1)}} \omega_{ik}^{(l+1)} \right)\,, & \\ & l \leq L-2\,. \end{cases} \tag{B.60}$$

As $e_n : \mathbb{C} \longrightarrow \mathbb{R}$, then applying (B.1.3):

$$\frac{\partial e_n}{\partial \overline{X}_i^{(l)}} = \overline{\left(\frac{\partial e_n}{\partial X_i^{(l)}} \right)}\,. \tag{B.61}$$

Using this latest equality with Equations B.55 and B.60, the backpropagation algorithms are fully defined.

C - Automatic Differentiation

This topic is also very well covered in Appendix D of [Géron, 2019]. The appendix also covers manual differentiation, symbolic differentiation, and numerical differentiation. In this Section, we will jump directly to forward-mode automatic differentiation (autodiff) and reverse-mode autodiff.

There many approaches to explain automatic differentiation (autodiff) [Hoffmann, 2016]. Most cases tend to assume the derivative at a given point exists which is a logical assumption having in mind the algorithm is computing the derivative itself. However, we have seen that for our case we can soften this definition and only ask for the *Wirtinger calculus* to exist. Furthermore, the requirement that the derivative at a point exists is a very strong condition that is to be avoided to compute complex number backpropagation. Even in the real domain, there are examples like Rectified Linear Unit (ReLU) that are widely used for deep neural networks that have no derivative at $x = 0$, and yet the backpropagation is applied without a problem.

- [Rall, 1986] explains autodiff it in a very clear and concise manner by defining the *dual number* (to be explained later in this Chapter) arithmetic directly but assumes that the derivative in the point exists.
- [Hall, 2003] is one of the few that actually talks about complex number autodiff but assumes that Cauchy-Riemann equations are valid.
- [Pearlmutter and Siskind, 2007] presents *Taylor series* as the main idea behind forward-mode autodiff. However, *Taylor series* assumes that the function is infinitely derivable at the desired point.
- [Rall, 1983] assumes the function is differentiable.

C.1 . Forward-mode automatic differentiation

In this Section, we will demonstrate the theory behind forward-mode autodiff in a general way, and we will not ask for f to be infinitely derivable at a point furthermore, we will not even require it to have a first derivative. We will later extend this definition to the complex domain. To the author's knowledge, this demonstration is not present in any other Reference, although the high quantity of bibliography on this subject may suggest otherwise. After defining forward-mode autodiff we will proceed to explain reverse-mode autodiff.

The definition of the derivative of function f is given by:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \,. \tag{C.1}$$

We will generalize this equation for only a one-sided limit. The choice of right- or left-sided limit will be indifferent to the demonstration:

$$Df(x) = \lim_{h \to 0^{\pm}} \frac{f(x+h) - f(x)}{h}\,, \tag{C.2}$$

where Df stands for this "soften" derivative definition and $\pm$ stands for either left $(-)$ or right $(+)$ sided limit.

In Equation C.2, we have "soften" the condition needed for the derivative. In cases where the derivative of f exists, we will not need to worry because (C.2) will converge to (C.1) hence being equivalent. However, in cases where the derivative does not exist, because the left side limit does not converge to the left side limit (like ReLU at $x = 0$), this definition will render a result.

Using the first order Taylor expansion, we have that

$$f(x+\epsilon) = f(x) + Df(x)\,\epsilon\,. \tag{C.3}$$

With, ϵ will be an infinitesimally small number such as $-0.00..001$ or $0.00..02$. Note that the last digit of ϵ can be anything; it can even be more than one digit as long as it is preceded by an "infinite" number of zeros.

Given a number $x = a + b\epsilon$, forward-mode automatic differentiation writes this number a tuple of two numbers in the form of $x = (a, b)$ called *dual numbers*. *Dual numbers* are represented in memory as a pair of floats. An arithmetic for this newly defined *dual numbers* are described in detail in [Rall, 1986]. We can think of dual numbers as a transformation $T_\epsilon[x] : \mathbb{R} \to \mathbb{R}^2 / T_\epsilon[a + b\,\epsilon] = (a, b)$ [Kedem, 1980]. The real number system maps isomorphically into this new space by the mapping $x \mapsto (x, 0), x \in \mathbb{R}$. To use the same notation as [Rall, 1986], we will call the *dual number* space $\mathbb{D}$. Operations in this space can be easily defined, for example:

$$\lambda(a, b) = (\lambda\, a, \lambda\, b)\,, \tag{C.4}$$

$$(a, b) + (c, d) = (a + b, c + d)\,, \tag{C.5}$$

$$(a, b)\,(c, d) = (a\,c, (a\,d + b\,c) + b\,d\,\epsilon) = (a\,c, a\,d + b\,c)\,. \tag{C.6}$$

See that in Equation C.6, we have approximated $(a\,c, (a\,d + b\,c) + b\,d\,\epsilon) = (a\,c, a\,d + b\,c)$. This is evident because the second term of the dual number is stored in memory as a float. Being $b\,d$ a product of two bounded scalar values and being ϵ, by definition, an infinitesimal number, the value yielded by the term $b\,d\,\epsilon$ will tend to zero and fall under the machine epsilon (machine precision). The existence of the additive inverse (or negative) and the identity element for addition can be easily defined. So is the case for multiplication. The multiplication and addition are commutative and associative. Basically, the space $\mathbb{D}$ is well defined [Rall, 1986].

The choice of ϵ will only affect how the transformation T_ϵ is applied but will not affect this demonstration nor the basic operations of the space $\mathbb{D}$.

If we rewrite Equation C.3 in *dual number* notation we have:

$$f\left((a,b)\right) = \left(f(a), b\,Df(a)[\epsilon]\right) \ . \tag{C.7}$$

Equation C.7 means that if we find a way to compute the function we want to derivative using dual number notation, the result will yield a dual number whose first number is the result of the function at point a and the second number will be the derivative (provided $b = 1$) at that same point. Note that the above equation has widened the "soft" derivative definition to $Df(x)[\epsilon]$ which means the derivative of f at point x on the ϵ direction, as the sign chosen of epsilon will define if we are computing either the left $(-)$ or right $(+)$ sided limit (C.2)).

The strength of forward-mode automatic differentiation is that this result is exact [Hall, 2003, Hoffmann, 2016]. This is, the only inaccuracies which occur are those which appear due to rounding errors in floating-point arithmetic or due to imprecise evaluations of elementary functions. If we had infinite float number precision and we could define the exact value of f on the *dual number* base, we will have the exact value of the derivative.

Another virtue of forward-mode autodiff is that f can be any coded function whose symbolic equation is unknown. Forward-mode autodiff can then compute any number of nested functions as long as the basic operations are well defined in *dual number* form. Then if f does many calls to these basic operations inside loops or any conditional call (making it difficult to derive the symbolic equation), it will still be possible for forward-mode autodiff to compute its derivative without any difficulty.

Examples

With the goal of helping the reader to better understand forward-mode autodiff, we will see the example used in [Géron, 2019], we define $f(x,y) = x^2y + y + 2$. For it we compute $f(x+\epsilon, y) = 42 + 24\,\epsilon$ following the logic of Figure C.1. Therefore, we conclude that $\dfrac{\partial f}{\partial x}(3,4) = 24$.

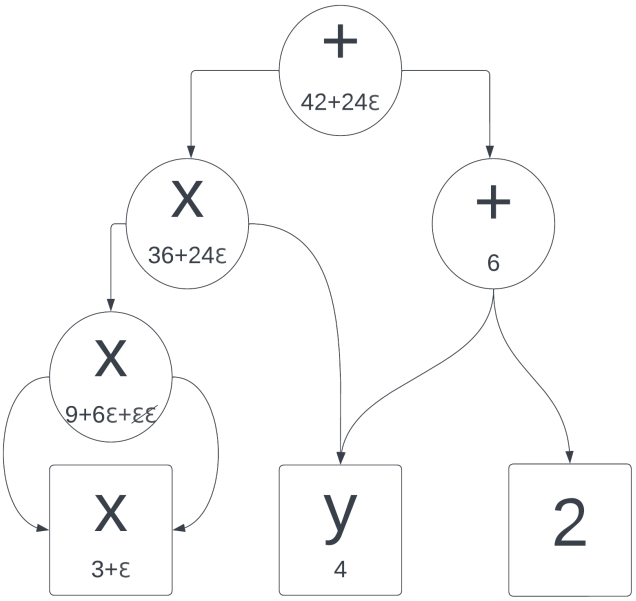


Figure C.1: Forward-mode autodiff block diagram example

Rectified Linear Unit (ReLU)

In another example, a very well-known function called Rectified Linear Unit (ReLU), is one of the most used activation functions in machine learning models. We can write ReLU as $f(x) = \max(0, x)$ for what it's derivative will be defined as:

$$f'(x) = \begin{cases} 1 & x > 0 \\ 0 & x < 0\,. \end{cases} \tag{C.8}$$

This is the example where:

$$\lim_{h \to 0^+} \frac{f(x+h) - f(x)}{h} = 1 \neq \lim_{h \to 0^-} \frac{f(x+h) - f(x)}{h} = 0\,. \tag{C.9}$$

Therefore, its derivative is not defined at $x = 0$. The theory of why this case doesn't pose any problem for reaching an optimal point when doing backpropagation is outside the scope of this text. However, we will see what the forward-mode automatic differentiation gives as a result. For it, we should first define, as usual, f in the *dual number* space:

$$f\left((a,b)\right) = \begin{cases} (a,b) & a > 0 \\ (0,b) & a = 0 \\ (0,0) & a < 0\,. \end{cases} \tag{C.10}$$

This definition is logical if we have chosen $\lim_{h^+ \to 0} = \epsilon$ and must be changed if the left-sided limit is chosen. We therefore compute the value of $f(0,1)$ to see the

value of the derivative at point $x = 0$:

$$\begin{aligned} &f(x+\epsilon) = \max(0, 0.00...01) = 0.00...01 = (0,1) = (f(0), f'(0))\,, \\ &f\left((0,1)\right) = (0,1)\,. \end{aligned} \tag{C.11}$$

We can see in Equation C.11 that the forward-mode autodiff algorithm will yield the result of $Df(0)[\epsilon] = 1$ which is an acceptable result for this case.

C.1.1 . Complex forward-mode automatic differentiation

With the generalization to the complex case, ϵ now becomes complex, in the real case, we arbitrarily chose either the left or right-sided limit. Now, limitless directions are possible (phase), affecting, of course, the result of the derivative. Table C.1 shows the derivative that is calculated when different values of ϵ are chosen.

Definition	One possibility for ϵ
$\lim_{x\to 0} \frac{f(z+x) - f(z)}{x}$	$0.00...01$
$\lim_{y\to 0} \frac{f(z+i\,y) - f(z)}{y}$	$0.00...01\,i$
$\lim_{x\to 0, y\to 0} \frac{f(z+x-i\,y) - f(z)}{x+y}$	$0.00...01\,(1-i)$
$\lim_{x\to 0, y\to 0} \frac{f(z+x+i\,y) - f(z)}{x+y}$	$0.00...01\,(1+i)$

Table C.1: Directional derivatives with respect to ϵ.

The Table can be generalized to the following equation:

$$\nabla_\epsilon f = \lim_{h\to 0} \frac{f(z+h\,\epsilon) - f(z)}{h\,\epsilon}\,. \tag{C.12}$$

It is important to note that the module of ϵ is unimportant as long as it is small enough so that the analysis made in the last Section (Section C.1) stands.

For functions where the complex derivative exists, all possibilities will converge to the same value.

The first and second terms on Table C.1 are used in the computation of *Wirtinger Calculus*, and it is not necessarily equal to the third term of the Table.

C.2 . Reverse-mode automatic differentiation

Forward-mode automatic differentiation has many useful properties. First of all, as we have seen, the condition for the derivative to exist is not necessarily required for the method to yield a result that can be helpful when dealing with, in

our case, non-holomorphic functions. Also, it allows finding a derivative value for any coded function, even containing loops or conditionals, as long as the primitives are defined. It is also natural for the algorithm to deal with the chain rule.

However, by taking a look at the example of Figure C.1, if we now want to compute $\frac{\partial f}{\partial y}(3,4)$ we will need to compute $f(x, y+\epsilon)$, meaning that in order to know both $\frac{\partial f}{\partial x}(3,4)$ and $\frac{\partial f}{\partial y}(3,4)$ we will need to compute the algorithm twice. This can be exponentially costly for neural networks where there are sometimes even millions of trainable parameters from which we need to compute the partial derivative. Here is where reverse-mode automatic differentiation comes to the rescue enabling us to compute all partial derivatives at once.

Examples

In here we will show how reverse-mode autodiff can compute the previous both $\frac{\partial f}{\partial x}(3,4)$ and $\frac{\partial f}{\partial y}(3,4)$ of the previous example for forward-mode autodiff running the code once.

The first step is to compute $f(3,4)$, whose intermediate values are shown on the bottom right of each node. Each node was labeled as n_i for clarity with $i \in [1,7]$. The output of $f(3,4) = n_7 = 42$ as expected.

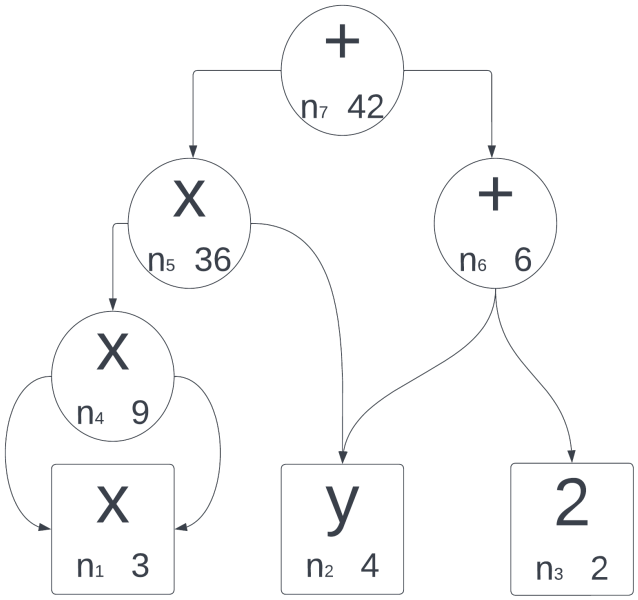


Figure C.2: Reverse-mode autodiff block diagram example

Now all the partial derivatives $\frac{\partial f}{\partial n_i}$ are computed starting with n_7. Since n_7

is the output node, $\frac{\partial f}{\partial n_7} = 1$. The chain rule is then used to compute the rest of the nodes by going down the graph. For example, to compute $\frac{\partial f}{\partial n_5}$ we use

$$\frac{\partial f}{\partial n_5} = \frac{\partial f}{\partial n_7}\frac{\partial n_7}{\partial n_5}\,, \tag{C.13}$$

and as we previously calculated $\frac{\partial f}{\partial n_7}$, we just need to compute the second term. This methodology will be repeated until all nodes' partial derivatives are computed. Here is the full list of the partial derivatives:

- $\frac{\partial f}{\partial n_7} = 1,$
- $\frac{\partial f}{\partial n_6} = \frac{\partial f}{\partial n_7}\frac{\partial n_7}{\partial n_6} = 1,$
- $\frac{\partial f}{\partial n_5} = \frac{\partial f}{\partial n_7}\frac{\partial n_7}{\partial n_5} = 1,$
- $\frac{\partial f}{\partial n_4} = \frac{\partial f}{\partial n_5}\frac{\partial n_5}{\partial n_4} = n_2 = 4,$
- $\frac{\partial f}{\partial y} = \frac{\partial f}{\partial n_2} = \frac{\partial f}{\partial n_5}\frac{\partial n_5}{\partial n_2} + \frac{\partial f}{\partial n_6}\frac{\partial n_6}{\partial n_2} = n_4 + 1 = 10$
- $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial n_1} = \frac{\partial f}{\partial n_5}\frac{\partial n_5}{\partial n_1} + \frac{\partial f}{\partial n_5}\frac{\partial n_5}{\partial n_1} = n_1 \cdot n_4 + n_1 \cdot n_4 = 3 \cdot 4 + 3 \cdot 4 = 24.$

This code has the advantage that all partial derivatives are computed at once which allows computing the values for a very high number of trainable parameters with less computational cost.

C.2.1 . Complex reverse-mode automatic differentiation

Complex Example

In the following complex example, we will compute the reverse automatic differentiation on a complex multiplication operation $f = |(a + i\,b)(c + i\,d)| = a\,c - b\,d + a\,d + b\,c$, we know by definition that the derivative, using *Wirtinger Calculus*, is:

$$\frac{\partial f}{\partial (a + i\,b)} = \frac{\partial f}{\partial a} + i\,\frac{\partial f}{\partial b} = (c + d) + i\,(c - d)\,, \tag{C.14}$$

$$\frac{\partial f}{\partial (c + i\,d)} = \frac{\partial f}{\partial c} + i\,\frac{\partial f}{\partial d} = (a + b) + i\,(a - b)\,. \tag{C.15}$$

The following Figure C.3 shows the block diagram of f.

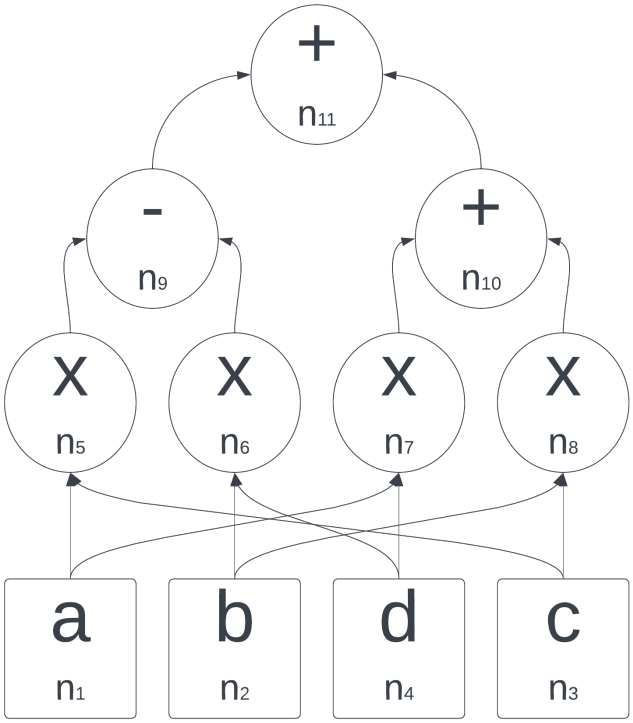


Figure C.3: Complex reverse-mode autodiff block diagram example

Using the block diagram of Figure C.3 we can compute the nodes as follows:

- $n_1 = a$,
- $n_2 = b$,
- $n_3 = c$,
- $n_4 = d$,
- $n_5 = n_1 \cdot n_3 = a \cdot c$,
- $n_6 = n_2 \cdot n_4 = b \cdot d$,
- $n_7 = n_1 \cdot n_4 = a \cdot d$,
- $n_8 = n_2 \cdot n_3 = b \cdot c$,
- $n_9 = n_5 - n_6 = a\,c - b\,d$,
- $n_{10} = n_7 + n_8 = a\,d + b\,c$,
- $f = n_{11} = n_{10} + n_9 = a\,c - b\,d + a\,d + b\,c$.

We now perform the reverse-mode backpropagation starting from the last node (n_{11}) and go back using the previously computed values to extract the partial derivative of $f = n_{11}$ with respect to every node.

- $\dfrac{\partial f}{\partial n_{11}} = \dfrac{\partial n_{11}}{\partial n_{11}} = 1,$
- $\dfrac{\partial n_{11}}{\partial n_{10}} = 1,$
- $\dfrac{\partial n_{11}}{\partial n_{9}} = 1,$
- $\dfrac{\partial n_{11}}{\partial n_{8}} = \dfrac{\partial n_{11}}{\partial n_{10}}\dfrac{\partial n_{10}}{\partial n_{8}} = 1,$
- $\dfrac{\partial n_{11}}{\partial n_{7}} = \dfrac{\partial n_{11}}{\partial n_{10}}\dfrac{\partial n_{10}}{\partial n_{7}} = 1,$
- $\dfrac{\partial n_{11}}{\partial n_{6}} = \dfrac{\partial n_{11}}{\partial n_{9}}\dfrac{\partial n_{9}}{\partial n_{6}} = -1,$
- $\dfrac{\partial n_{11}}{\partial n_{5}} = \dfrac{\partial n_{11}}{\partial n_{9}}\dfrac{\partial n_{9}}{\partial n_{5}} = 1,$
- $\dfrac{\partial f}{\partial d} = \dfrac{\partial n_{11}}{\partial n_{4}} = \dfrac{\partial n_{11}}{\partial n_{7}}\dfrac{\partial n_{7}}{\partial n_{4}} + \dfrac{\partial n_{11}}{\partial n_{6}}\dfrac{\partial n_{6}}{\partial n_{4}} = n_1 - n_2 = a - b,$
- $\dfrac{\partial f}{\partial c} = \dfrac{\partial n_{11}}{\partial n_{3}} = \dfrac{\partial n_{11}}{\partial n_{8}}\dfrac{\partial n_{8}}{\partial n_{3}} + \dfrac{\partial n_{11}}{\partial n_{5}}\dfrac{\partial n_{5}}{\partial n_{3}} = n_2 + n_1 = b + a,$
- $\dfrac{\partial f}{\partial b} = \dfrac{\partial n_{11}}{\partial n_{2}} = \dfrac{\partial n_{11}}{\partial n_{6}}\dfrac{\partial n_{6}}{\partial n_{2}} + \dfrac{\partial n_{11}}{\partial n_{8}}\dfrac{\partial n_{8}}{\partial n_{2}} = n_3 - n_4 = c - d,$
- $\dfrac{\partial f}{\partial a} = \dfrac{\partial n_{11}}{\partial n_{1}} = \dfrac{\partial n_{11}}{\partial n_{5}}\dfrac{\partial n_{5}}{\partial n_{1}} + \dfrac{\partial n_{7}}{\partial n_{1}}\dfrac{\partial n_{7}}{\partial n_{1}} = n_3 + n_4 = c + d.$

Now, if we consider $\partial f/\partial(a + i\,b) = \partial f/\partial a + i\,\partial f/\partial b$ and replace the values we obtained in the previous list, we get the same result as in Equation C.14.

Bibliography

[Aizenberg et al., 1973] Aizenberg, N. N., Ivas'kiv, Y. L., Pospelov, D. A., and Khudyakov, G. F. (1973). Multivalued threshold functions. *Cybernetics*, 9(1):61–77.

[Ajzenberg and Tošić, 1972] Ajzenberg, N. N. and Tošić, Z. (1972). A generalization of the threshold functions. *Publikacije Elektrotehničkog Fakulteta. Serija Matematika i Fizika*, pages 97–99.

[Akramifard et al., 2012] Akramifard, H., Firouzmand, M., and Moghadam, R. A. (2012). Extracting, recognizing, and counting white blood cells from microscopic images by using complex-valued Neural Networks. *Journal of Medical Signals and Sensors*, 2(3):169.

[Al-Nuaimi et al., 2012] Al-Nuaimi, A. Y. H., Amin, M. F., and Murase, K. (2012). Enhancing MP3 encoding by utilizing a predictive complex-valued Neural Network. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE.

[Amilia et al., 2015] Amilia, S., Sulistiyo, M. D., and Dayawati, R. N. (2015). Face image-based gender recognition using complex-valued Neural Network. In *2015 3rd International Conference on Information and Communication Technology (ICoICT)*, pages 201–206. IEEE.

[Amin et al., 2011] Amin, M. F., Amin, M. I., Al-Nuaimi, A. Y. H., and Murase, K. (2011). Wirtinger Calculus Based Gradient Descent and Levenberg-Marquardt Learning Algorithms in Complex-Valued Neural Networks. In *Neural Information Processing*, pages 550–559, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Antoniou et al., 2017] Antoniou, A., Storkey, A., and Edwards, H. (2017). Data augmentation generative adversarial networks.

[Argenti and Alparone, 2002] Argenti, F. and Alparone, L. (2002). Speckle removal from SAR images in the undecimated wavelet domain. *IEEE Transactions on Geoscience and Remote Sensing*, 40(11):2363–2374.

[Arjovsky et al., 2016] Arjovsky, M., Shah, A., and Bengio, Y. (2016). Unitary evolution recurrent Neural Networks. In *International Conference on Machine Learning*, pages 1120–1128. PMLR.

[Bacaër, 2011] Bacaër, N. (2011). *Verhulst and the logistic equation (1838)*, pages 35–39. Springer London, London.

[Bain, 1873] Bain, A. (1873). *Mind and body: The theories of their relation*, volume 4. D. Appleton.

[Bain, 1883] Bain, A. (1883). Notes and discussion: Mind and Body. *Mind*, 8(31):402–412.

[Baqué et al., 2019] Baqué, R., Dreuillet, P., and Oriot, H. (2019). SETHI: Review of 10 years of development and experimentation of the remote sensing platform. In *2019 International Radar Conference (RADAR)*, pages 1–5.

[Barbaresco and Chevalier, 2008] Barbaresco, F. and Chevalier, P. (2008). Non-circularity exploitation in signal processing overview and application to radar. In *2008 IET Waveform Diversity Digital Radar Conference - Day 1: Waveform Diversity Design*, pages 1–6.

[Bargsten and Schlaefer, 2020] Bargsten, L. and Schlaefer, A. (2020). SpeckleGAN: a generative adversarial network with an adaptive speckle layer to augment limited training data for ultrasound image processing. *International Journal of Computer Assisted Radiology and Surgery*, 15(9):1427–1436.

[Barrachina, 2019] Barrachina, J. A. (2019). Complex-Valued Neural Networks (CVNN). https://github.com/NEGU93/cvnn.

[Barrachina, 2020] Barrachina, J. A. (2020). A comparison between complex and real-valued fully connected neural networks on non-circular complex data. XXII Giambiagi Winter School: Artificial intelligence and deep learning in physics.

[Barrachina, 2021] Barrachina, J. A. (2021). NEGU93/cvnn. https://doi.org/10.5281/zenodo.4452131.

[Barrachina et al., 2022a] Barrachina, J. A., Ren, C., Morisseau, Vieillard, G., C., and Ovarlez, J.-P. (2022a). Complex-Valued Neural Networks for Polarimetric SAR segmentation using Pauli representation. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*.

[Barrachina et al., 2022b] Barrachina, J. A., Ren, C., Morisseau, Vieillard, G., C., and Ovarlez, J.-P. (2022b). Complex-Valued Neural Networks for Polarimetric SAR segmentation using Pauli representation. 5th SONDRA Workshop.

[Barrachina et al., 2022c] Barrachina, J. A., Ren, C., Morisseau, Vieillard, G., C., and Ovarlez, J.-P. (2022c). Merits of Complex-Valued Neural Networks for PolSAR image segmentation. In *GRETSI XXVIIIème Colloque Francophone de Traitement du Signal et des Images*.

[Barrachina et al., 2021a] Barrachina, J. A., Ren, C., Morisseau, C., Vieillard, G., and Ovarlez, J.-P. (2021a). Complex-Valued vs. Real-Valued Neural Networks for Classification Perspectives: An Example on Non-Circular Data. In *ICASSP*

2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 2990–2994.

[Barrachina et al., 2021b] Barrachina, J. A., Ren, C., Morisseau, C., Vieillard, G., and Ovarlez, J.-P. (2021b). Complex-Valued vs. Real-Valued Neural Networks for Classification Perspectives: An Example on Non-Circular Data.

[Barrachina et al., 2022d] Barrachina, J. A., Ren, C., Morisseau, C., Vieillard, G., and Ovarlez, J.-P. (2022d). Comparison Between Equivalent Architectures of Complex-Valued and Real-Valued Neural Networks - Application on Polarimetric SAR Image Segmentation. *Journal of Signal Processing Systems*, pages 1–10.

[Barrachina et al., 2021c] Barrachina, J. A., Ren, C., Vieillard, G., Morisseau, C., and Ovarlez, J.-P. (2021c). About the Equivalence Between Complex-Valued and Real-Valued Fully Connected Neural Networks - Application to PolInSAR Images. In *2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6.

[Barrachina et al., 2022e] Barrachina, J. A., Ren, C., Vieillard, G., Morisseau, C., and Ovarlez, J.-P. (2022e). Real- and Complex-Valued Neural Networks for SAR image segmentation through different polarimetric representations. In *2022 IEEE International Conference on Image Processing (ICIP)*.

[Bassey et al., 2021] Bassey, J., Qian, L., and Li, X. (2021). A survey of complex-valued Neural Networks. *arXiv preprint arXiv:2101.12249*.

[Ben Hamida et al., 2018] Ben Hamida, A., Benoit, A., Lambert, P., and Ben Amar, C. (2018). 3-D Deep Learning Approach for Remote Sensing Image Classification. *IEEE Transactions on Geoscience and Remote Sensing*, 56(8):4420–4434.

[Benvenuto and Piazza, 1992] Benvenuto, N. and Piazza, F. (1992). On the complex backpropagation algorithm. *IEEE Transactions on Signal Processing*, 40(4):967–969.

[bkkm16, 2019] bkkm16 (2019). What we should use align corners to False?

[Boeddeker et al., 2017] Boeddeker, C., Hanebrink, P., Drude, L., Heymann, J., and Haeb-Umbach, R. (2017). On the computation of complex-valued gradients with application to statistically optimum beamforming. *arXiv preprint arXiv:1701.00392*.

[Bratasanu et al., 2011] Bratasanu, D., Nedelcu, I., and Datcu, M. (2011). Bridging the semantic gap for satellite image annotation and automatic mapping applications. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 4(1):193–204.

[Bridle, 1989] Bridle, J. (1989). Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In Touretzky, D., editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann.

[Burnham, 1888] Burnham, W. H. (1888). Memory, Historically and Experimentally Considered. I. An Historical Sketch of the Older Conceptions of Memory. *The American Journal of Psychology*, 2(1):39–90.

[Cao et al., 2019] Cao, Y., Wu, Y., Zhang, P., Liang, W., and Li, M. (2019). Pixel-wise PolSAR image classification via a novel complex-valued deep fully convolutional network. *Remote Sensing*, 11(22):2653.

[Carson, 1926] Carson, J. R. (1926). Electric circuit theory and the operational calculus. *The Bell System Technical Journal*, 5(2):336–384.

[Castro, 2020] Castro, F. (2020). We're (data) hungry: the importance of big data.

[Çevik et al., 2018] Çevik, H. H., Acar, Y. E., and Çunkaş, M. (2018). Day ahead wind power forecasting using complex-valued Neural Network. In *2018 International Conference on Smart Energy Systems and Technologies (SEST)*, pages 1–6. IEEE.

[Cha and Kassam, 1995] Cha, I. and Kassam, S. A. (1995). Channel equalization using adaptive complex radial basis function networks. *IEEE Journal on Selected Areas in communications*, 13(1):122–131.

[Chambers, 2018] Chambers, J. M. (2018). *Graphical methods for data analysis*. CRC Press.

[Chen et al., 2016] Chen, S., Wang, H., Xu, F., and Jin, Y.-Q. (2016). Target classification using the deep convolutional networks for SAR images. *IEEE Transactions on Geoscience and Remote Sensing*, 54(8):4806–4817.

[Chierchia et al., 2017] Chierchia, G., Cozzolino, D., Poggi, G., and Verdoliva, L. (2017). SAR image despeckling through Convolutional Neural Networks. In *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 5438–5441.

[Chistyakov et al., 2011] Chistyakov, Y. S., Kholodova, E. V., Minin, A. S., Zimmermann, H.-G., and Knoll, A. (2011). Modeling of electric power transformer using complex-valued Neural Networks. *Energy Procedia*, 12:638–647.

[Choi et al., 2019] Choi, D., Shallue, C. J., Nado, Z., Lee, J., Maddison, C. J., and Dahl, G. E. (2019). On empirical comparisons of optimizers for deep learning.

[Clevert et al., 2015] Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2015). Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs).

[Commission d'enrichissement de la langue française, 2019] Commission d'enrichissement de la langue française (2019). Bulletin officiel n°6 du 7 février 2019. Bulletin Officiel ISSN: 2110-6061, Ministère de l'Enseignement supérieur, de la Recherche et de l'Innovation.

[Copeland, 2016] Copeland, M. (2016). What's the Difference Between Artificial Intelligence, Machine Learning and Deep Learning? Nvidia's Blog. https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/.

[Dabov et al., 2007] Dabov, K., Foi, A., Katkovnik, V., and Egiazarian, K. (2007). Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering. *IEEE Transactions on Image Processing*, 16(8):2080–2095.

[Dalsasso et al., 2022a] Dalsasso, E., Denis, L., Muzeau, M., and Tupin, F. (2022a). Self-supervised training strategies for SAR image despeckling with Deep Neural Networks. In *14th European Conference on Synthetic Aperture Radar (EUSAR)*, Leipzig, Germany.

[Dalsasso et al., 2021] Dalsasso, E., Denis, L., and Tupin, F. (2021). Sar2sar: A semi-supervised despeckling algorithm for SAR images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14:4321–4329.

[Dalsasso et al., 2022b] Dalsasso, E., Denis, L., and Tupin, F. (2022b). As if by magic: Self-supervised training of deep despeckling networks with MERLIN. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–13.

[Datcu et al., 2007] Datcu, M., Schwarz, G., Soccorsi, M., and Chaabouni, H. (2007). Phase information contained in meter-scale SAR images. In Notarnicola, C. and Posa, F., editors, *SAR Image Analysis, Modeling, and Techniques IX*, volume 6746, page 67460H. International Society for Optics and Photonics, SPIE.

[De et al., 2017] De, S., Bruzzone, L., Bhattacharya, A., Bovolo, F., and Chaudhuri, S. (2017). A novel technique based on deep learning and a synthetic target database for classification of urban areas in PolSAR data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11(1):154–170.

[Deledalle et al., 2017] Deledalle, C.-A., Denis, L., Tabti, S., and Tupin, F. (2017). Mulog, or how to apply gaussian denoisers to multi-channel SAR speckle reduction? *IEEE Transactions on Image Processing*, 26(9):4389–4403.

[Deledalle et al., 2009] Deledalle, C.-A., Denis, L., and Tupin, F. (2009). Iterative Weighted Maximum Likelihood Denoising With Probabilistic Patch-Based Weights. *IEEE Transactions on Image Processing*, 18(12):2661–2672.

[Deledalle et al., 2011] Deledalle, C.-A., Denis, L., and Tupin, F. (2011). NL-InSAR: Nonlocal Interferogram Estimation. *IEEE Transactions on Geoscience and Remote Sensing*, 49(4):1441–1452.

[Deledalle et al., 2015] Deledalle, C.-A., Denis, L., Tupin, F., Reigber, A., and Jäger, M. (2015). NL-SAR: A Unified Nonlocal Framework for Resolution-Preserving (Pol)(In)SAR Denoising. *IEEE Transactions on Geoscience and Remote Sensing*, 53(4):2021–2038.

[Deledalle et al., 2010] Deledalle, C.-A., Tupin, F., and Denis, L. (2010). Polarimetric SAR estimation based on non-local means. In *2010 IEEE International Geoscience and Remote Sensing Symposium*, pages 2515–2518.

[Desai, 2020] Desai, C. (2020). Comparative analysis of optimizers in deep neural networks. *International Journal of Innovative Science and Research Technology*, 5(10):959–962.

[Ding and Hirose, 2014] Ding, T. and Hirose, A. (2014). Fading channel prediction based on combination of complex-valued Neural Networks and chirp z-transform. *IEEE Transactions on Neural Networks and Learning Systems*, 25(9):1686–1695.

[Dong et al., 2020] Dong, H., Zou, B., Zhang, L., and Zhang, S. (2020). Automatic Design of CNNs via Differentiable Neural Architecture Search for PolSAR Image Classification. *IEEE Transactions on Geoscience and Remote Sensing*, 58(9):6362–6375.

[Dong and Huang, 2019] Dong, T. and Huang, T. (2019). Neural cryptography based on complex-valued Neural Network. *IEEE Transactions on Neural Networks and learning systems*, 31(11):4999–5004.

[Dozat, 2016] Dozat, T. (2016). Incorporating Nesterov momentum into Adam. In *ICLR International Conference on Learning Representations*.

[Dramsch, 2019] Dramsch, J. S. e. a. (2019). Complex-Valued Neural Networks in Keras with Tensorflow.

[Dreuillet et al., 2006] Dreuillet, P., Cantalloube, H., Colin, E., Dubois-Fernandezx, P., Dupuis, X., Fromage, P., Garestier, F., Heuze, D., Oriot, H., Peron, J., Peyret, J., Bonin, G., du Plessis, O., Nouvel, J., and Vaizan, B. (2006). The ONERA RAMSES SAR: latest significant results and future developments. In *2006 IEEE Conference on Radar*, pages 7 pp.–.

[Dubois-Fernandez et al., 2002] Dubois-Fernandez, P., du Plessis, O., le Coz, D., Dupas, J., Vaizan, B., Dupuis, X., Cantalloube, H., Coulombeix, C., Titin-Schnaider, C., Dreuillet, P., Boutry, J., Canny, J., Kaisersmertz, L., Peyret, J., Martineau, P., Chanteclerc, M., Pastore, L., and Bruyant, J. (2002). The ONERA RAMSES SAR system. In *IEEE International Geoscience and Remote Sensing Symposium*, volume 3, pages 1723–1725 vol.3.

[Duchi et al., 2011] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).

[Dumoulin and Visin, 2016] Dumoulin, V. and Visin, F. (2016). A guide to convolution arithmetic for deep learning.

[Duvaut, 1994] Duvaut, P. (1994). *Traitement du signal: concepts et applications*. Hermes.

[El-Darymli et al., 2015] El-Darymli, K., Mcguire, P., Gill, E. W., Power, D., and Moloney, C. (2015). Characterization and statistical modeling of phase in single-channel synthetic aperture radar imagery. *IEEE Transactions on Aerospace and Electronic Systems*, 51(3):2071–2092.

[El-Darymli et al., 2013] El-Darymli, K., McGuire, P., Power, D., and Moloney, C. (2013). Rethinking the phase in single-channel sar imagery. In *2013 14th International Radar Symposium (IRS)*, volume 1, pages 429–436.

[El-Darymli et al., 2014] El-Darymli, K., Moloney, C., Gill, E., McGuire, P., and Power, D. (2014). On circularity/noncircularity in single-channel synthetic aperture radar imagery. In *2014 Oceans - St. John's*, pages 1–4.

[Fischer, 2005] Fischer, R. F. H. (2005). *Appendix A: Wirtinger Calculus*, pages 405–413. John Wiley & Sons, Ltd.

[Fix et al., 2021] Fix, J., Ren, C., Costa Lopes, A., Morice, G., Kobayashi, S., Leterte, T., and Hinostroza Sáenz, I. D. (2021). Deep learning for aircraft classification from vhf radar signatures. *IET Radar, Sonar & Navigation*, 15(7):697–707.

[Fix et al., 2022] Fix, J., Vialle, S., Hellequin, R., Mercier, C., Mercier, P., and Tavernier, J. (2022). Feedback from a data center for education at CentraleSupélec engineering school. In *EduPar-22: 12th NSF/TCPP Workshop on Parallel and Distributed Computing Education*.

[Formont et al., 2010] Formont, P., Pascal, F., Vasile, G., Ovarlez, J.-P., and Ferro-Famil, L. (2010). Statistical classification for heterogeneous polarimetric SAR images. *IEEE Journal of selected topics in Signal Processing*, 5(3):567–576.

[Frost et al., 1982] Frost, V. S., Stiles, J. A., Shanmugan, K. S., and Holtzman, J. C. (1982). A model for radar images and its application to adaptive digital filtering of multiplicative noise. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-4(2):157–166.

[Gabor, 1946] Gabor, D. (1946). Theory of communications. *J. Inst. Elec. Eng*, 93:429–457.

[Georgiou and Koutsougeras, 1992] Georgiou, G. M. and Koutsougeras, C. (1992). Complex domain backpropagation. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 39(5):330–334.

[German Aerospace Center (DLR), 1988a] German Aerospace Center (DLR) (1988a). E-SAR The Experimental airborne SAR System of DLR.

[German Aerospace Center (DLR), 1988b] German Aerospace Center (DLR) (1988b). E-SAR – The Airborne SAR System of DLR.

[Géron, 2019] Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media.

[Gleich and Sipos, 2018] Gleich, D. and Sipos, D. (2018). Complex valued convolutional Neural Network for TerraSAR-X patch categorization. In *EUSAR 2018; 12th European Conference on Synthetic Aperture Radar*, pages 1–4. VDE.

[Glorot and Bengio, 2010] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

[Gong et al., 2017] Gong, W., Liang, J., and Li, D. (2017). Design of high-capacity auto-associative memories based on the analysis of complex-valued Neural Networks. In *2017 International Workshop on Complex Systems and Networks (IWCSN)*, pages 161–168. IEEE.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[Goodman, 1976] Goodman, J. W. (1976). Some fundamental properties of speckle$*$. *J. Opt. Soc. Am.*, 66(11):1145–1150.

[Gu and Ding, 2018] Gu, S. and Ding, L. (2018). A complex-valued VGG network-based deep learning algorithm for image recognition. In *2018 Ninth International Conference on Intelligent Control and Information Processing (ICICIP)*, pages 340–343. IEEE.

[Guberman, 2016] Guberman, N. (2016). On Complex Valued Convolutional Neural Networks. *CoRR*, abs/1602.09046.

[Guo et al., 2015] Guo, Y., Wang, S., Gao, C., Shi, D., Zhang, D., and Hou, B. (2015). Wishart RBM based DBN for polarimetric synthetic radar data classification. In *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 1841–1844. IEEE.

[Gérard et al., 2021] Gérard, J., Tomasik, J., Morisseau, C., Rimmel, A., and Vieillard, G. (2021). Micro-doppler signal representation for drone classification by deep learning. In *2020 28th European Signal Processing Conference (EUSIPCO)*, pages 1561–1565.

[Hafiz et al., 2015] Hafiz, A. R., Al-Nuaimi, A. Y., Amin, M., Murase, K., et al. (2015). Classification of skeletal wireframe representation of hand gesture using complex-valued Neural Network. *Neural Processing Letters*, 42(3):649–664.

[Hahn, 1996] Hahn, S. (1996). Hilbert Transforms in Signal Processing. *Artech House, Norwood, USA*.

[Hall, 2003] Hall, B. D. (2003). Calculating measurement uncertainty for complex-valued quantities. *Measurement Science and Technology*, 14(3):368.

[Hänsch, 2010] Hänsch, R. (2010). Complex-Valued Multi-Layer Perceptrons - An Application to Polarimetric SAR Data. *Photogrammetric Engineering & Remote Sensing*, 76(9):1081–1088.

[Hänsch and Hellwich, 2009a] Hänsch, R. and Hellwich, O. (2009a). Classification of polarimetric SAR data by complex-valued neural networks. In *Proc. ISPRS Hannover Workshop, High-Resolution Earth Imag. Geospatial Inf.*, volume 37.

[Hänsch and Hellwich, 2009b] Hänsch, R. and Hellwich, O. (2009b). Classification of polarimetric SAR data by complex-valued Neural Networks. In *ISPRS Workshop High-Resolution Earth Imaging for Geospatial Information*, volume 38, pages 4–7.

[Hänsch and Hellwich, 2010] Hänsch, R. and Hellwich, O. (2010). Complex-valued convolutional neural networks for object detection in PolSAR data. In *8th European Conference on Synthetic Aperture Radar*, pages 1–4. VDE.

[Hardy, 1902] Hardy, G. H. (1902). The Theory of Cauchy's Principal Values. (Third Paper: Differentiation and Integration of Principal Values.). *Proceedings of the London Mathematical Society*, 1(1):81–107.

[Hardy, 1909] Hardy, G. H. (1909). The Theory of Cauchy's Principal Values.(Fourth Paper: The Integration of Principal Values—Continued—with Applications to the Inversion of Definite Integrals). *Proceedings of the London Mathematical Society*, 2(1):181–208.

[Hardy, 1924a] Hardy, G. H. (1924a). Notes on some points in the integral calculus (LIX). On Hilbert transforms. *Messenger of Math.*, 54:81–88.

[Hardy, 1924b] Hardy, G. H. (1924b). Notes on some points in the integral calculus (LVIII). On Hilbert transforms. *Messenger of Math.*, 54:20–27.

[Hayakawa et al., 2018] Hayakawa, D., Masuko, T., and Fujimura, H. (2018). Applying complex-valued Neural Networks to acoustic modeling for speech recognition. In *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1725–1731. IEEE.

[Haykin, 2005] Haykin, S. S. (2005). *Adaptive filter theory*. Pearson Education India.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on Imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

[Heaton, 2008] Heaton, J. (2008). *Introduction to neural networks with Java*. Heaton Research, Inc.

[Hebb, 1949] Hebb, D. (1949). *The Organization of Behavior: A Neuropsychological Theory*. A Wiley book in clinical psychology. Wiley.

[Hilbert, 1912] Hilbert, D. (1912). *Fundamentals of a general theory of linear integral equations*, volume 3. BG Teubner.

[Hirose, 1992] Hirose, A. (1992). Continuous complex-valued back-propagation learning. *Electronics Letters*, 20(28):1854–1855.

[Hirose, 2003] Hirose, A. (2003). *Complex-valued Neural Networks: theories and applications*, volume 5. World Scientific.

[Hirose, 2009] Hirose, A. (2009). Complex-valued neural networks: The merits and their origins. In *2009 International Joint Conference on Neural Networks*, pages 1237–1244. IEEE.

[Hirose, 2010] Hirose, A. (2010). Recent progress in applications of complex-valued Neural Networks. In *International Conference on Artificial Intelligence and Soft Computing*, pages 42–46. Springer.

[Hirose, 2011] Hirose, A. (2011). Nature of complex number and complex-valued neural networks. *Frontiers of Electrical and Electronic Engineering in China*, 6(1):171–180.

[Hirose, 2012] Hirose, A. (2012). *Complex-valued Neural Networks*, volume 400. Springer Science & Business Media.

[Hirose, 2013] Hirose, A. (2013). *Complex-valued Neural Networks: Advances and applications*, volume 18. John Wiley & Sons.

[Hirose et al., 2013] Hirose, A., Amin, F., and Murase, K. (2013). Learning Algorithms in Complex-Valued Neural Networks using Wirtinger Calculus. In *Complex-Valued Neural Networks: Advances and Applications*, pages 75–102. IEEE.

[Hirose and Yoshida, 2012] Hirose, A. and Yoshida, S. (2012). Generalization characteristics of complex-valued feedforward neural networks in relation to signal coherence. *IEEE Transactions on Neural Networks and learning systems*, 23(4):541–551.

[Hoffmann, 2016] Hoffmann, P. H. W. (2016). A Hitchhiker's guide to automatic differentiation. *Numerical Algorithms*, 72(3):775–811.

[Hornik et al., 1989] Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer Feedforward Networks are Universal Approximators. *Neural Networks*, 2(5):359–366.

[Hou et al., 2016] Hou, B., Kou, H., and Jiao, L. (2016). Classification of polarimetric SAR images using multilayer autoencoders and superpixels. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 9(7):3072–3081.

[Hu et al., 2008] Hu, J., Li, Z., Hu, Z., Yao, D., and Yu, J. (2008). Spam detection with complex-valued neural network using behavior-based characteristics. In *2008 Second International Conference on Genetic and Evolutionary Computing*, pages 166–169.

[Huang et al., 2021] Huang, Z., Dumitru, C. O., Pan, Z., Lei, B., and Datcu, M. (2021). Classification of Large-Scale High-Resolution SAR Images With Deep Transfer Learning. *IEEE Geoscience and Remote Sensing Letters*, 18(1):107–111.

[Hunger, 2007] Hunger, R. (2007). An introduction to complex differentials and complex differentiability. Technical report, Munich University of Technology, Inst. for Circuit Theory and Signal Processing.

[Inisan et al., 2022] Inisan, M., Morisseau, C., TERREAUX, E., Vieillard, G., and Jourdan, A. (2022). Rapport de stage: Étiquetage par IA de basses d'apprentissage pour applications radar. Technical report, ONERA / CY Tech.

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *CoRR*, abs/1502.03167.

[Jankowski et al., 1996] Jankowski, S., Lozowski, A., and Zurada, J. M. (1996). Complex-valued multistate neural associative memory. *IEEE Transactions on Neural Networks*, 7(6):1491–1496.

[Jet Propulsion Laboratory (JPL), 2008] Jet Propulsion Laboratory (JPL) (2008). The AIRSAR Mission.

[Jianping et al., 2002] Jianping, D., Sundararajan, N., and Saratchandran, P. (2002). Communication channel equalization using complex-valued minimal radial basis function Neural Networks. *IEEE Transactions on Neural Networks*, 13(3):687–696.

[Jiao and Liu, 2016] Jiao, L. and Liu, F. (2016). Wishart deep stacking network for fast PolSAR image classification. *IEEE Transactions on Image Processing*, 25(7):3273–3286.

[Kang et al., 2020] Kang, D.-Y., Duong, P., and Park, J.-C. (2020). Application of deep learning in dentistry and implantology. *The Korean Academy of Oral and Maxillofacial Implantology*, 24:148–181.

[Kataoka et al., 1998] Kataoka, M., Kinouchi, M., and Hagiwara, M. (1998). Music information retrieval system using complex-valued Recurrent Neural Networks. In *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218)*, volume 5, pages 4290–4295. IEEE.

[Kawaguchi, 2000] Kawaguchi, K. (2000). *A multithreaded software model for backpropagation neural network applications*. The University of Texas at El Paso.

[Kaynak, 2021] Kaynak, O. (2021). The golden age of Artificial Intelligence.

[Kedem, 1980] Kedem, G. (1980). Automatic differentiation of computer programs. *ACM Transactions on Mathematical Software (TOMS)*, 6(2):150–165.

[Kim and Adali, 2001a] Kim, T. and Adali, T. (2001a). Approximation by fully complex MLP using elementary transcendental activation functions. In *Neural Networks for Signal Processing XI: Proceedings of the 2001 IEEE Signal Processing Society Workshop (IEEE Cat. No.01TH8584)*, pages 203–212. IEEE.

[Kim and Adali, 2001b] Kim, T. and Adali, T. (2001b). Complex backpropagation Neural Network using elementary transcendental activation functions. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221)*, volume 2, pages 1281–1284. IEEE.

[Kim and Adali, 2002] Kim, T. and Adali, T. (2002). Fully complex multi-layer perceptron network for nonlinear signal processing. *Journal of VLSI signal processing systems for signal, image and video technology*, 32(1):29–43.

[King, 2009] King, F. (2009). Hilbert Transforms: Encyclopedia of Mathematics and its Applications 124.

[Kingma and Ba, 2014] Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Kinouchi and Hagiwara, 1996] Kinouchi, M. and Hagiwara, M. (1996). Memorization of melodies by complex-valued recurrent network. In *Proceedings of International Conference on Neural Networks (ICNN'96)*, volume 2, pages 1324–1328. IEEE.

[Klambauer et al., 2017] Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. (2017). Self-normalizing neural networks. *Advances in neural information processing systems*, 30.

[Ko et al., 2022] Ko, M., Panchal, U. K., Andrade-Loarca, H., and Mendez-Vazquez, A. (2022). CoShNet: A Hybird Complex Valued Neural Network using Shearlets.

[Konishi et al., 2021] Konishi, B., Hirose, A., and Natsuaki, R. (2021). Complex-Valued Reservoir Computing for Interferometric SAR Applications with Low Computational Cost and High Resolution. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14:7981–7993.

[Kramer et al., 2002] Kramer, H. J. et al. (2002). *Observation of the Earth and its Environment: Survey of Missions and Sensors*, volume 1982. Springer.

[Kramers, 1927] Kramers, H. A. (1927). La diffusion de la lumière par les atomes. In *Atti Cong. Intern. Fisica (Transactions of Volta Centenary Congress) Como*, volume 2, pages 545–557.

[Kreutz-Delgado, 2009] Kreutz-Delgado, K. (2009). The Complex Gradient Operator and the CR-Calculus. *arXiv e-prints*, page arXiv:0906.4835.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[Krizhevsky et al., 2017] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.

[Kronig, 1926] Kronig, R. d. L. (1926). On the theory of dispersion of x-rays. *Journal of the Optical Society of America*, 12(6):547–557.

[Krull et al., 2019] Krull, A., Buchholz, T.-O., and Jug, F. (2019). Noise2Void - Learning Denoising From Single Noisy Images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Kuan et al., 1987] Kuan, D., Sawchuk, A., Strand, T., and Chavel, P. (1987). Adaptive restoration of images with speckle. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(3):373–383.

[Kulkarni and Joshi, 2015] Kulkarni, P. and Joshi, P. (2015). *Artificial intelligence: building intelligent systems*. PHI Learning Pvt. Ltd.

[Kuroe et al., 2003] Kuroe, Y., Yoshid, M., and Mori, T. (2003). On activation functions for complex-valued Neural Networks: existence of energy functions. In *Artificial Neural Networks and Neural Information Processing, ICANN/ICONIP 2003*, pages 985–992. Springer.

[Laine et al., 2019] Laine, S., Karras, T., Lehtinen, J., and Aila, T. (2019). High-Quality Self-Supervised Deep Image Denoising. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

[Lasswell, 2005] Lasswell, S. W. (2005). History of SAR at Lockheed Martin (previously Goodyear Aerospace). In Trebits, R. N. and Kurtz, J. L., editors, *Radar Sensor Technology IX*, volume 5788, pages 1 – 12. International Society for Optics and Photonics, SPIE.

[Lattari et al., 2019] Lattari, F., Gonzalez Leon, B., Asaro, F., Rucci, A., Prati, C., and Matteucci, M. (2019). Deep Learning for SAR Image Despeckling. *Remote Sensing*, 11(13).

[Le Cun et al., 1999] Le Cun, Y., Haffner, P., Bottou, L., and Bengio, Y. (1999). Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer.

[Le Cun et al., 1995] Le Cun, Y., Jackel, L. D., Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Muller, U. A., Sackinger, E., Simard, P., et al. (1995). Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*, 261(276):2.

[Lee et al., 2022] Lee, C., Hasegawa, H., and Gao, S. (2022). Complex-Valued Neural Networks: A Comprehensive Survey. *IEEE/CAA Journal of Automatica Sinica*, 9(8):1406–1426.

[Lee, 1981] Lee, J.-S. (1981). Speckle analysis and smoothing of synthetic aperture radar images. *Computer Graphics and Image Processing*, 17(1):24–32.

[Lee et al., 2006] Lee, J.-S., Grunes, M., Schuler, D., Pottier, E., and Ferro-Famil, L. (2006). Scattering-model-based speckle filtering of polarimetric SAR data. *IEEE Transactions on Geoscience and Remote Sensing*, 44(1):176–187.

[Lee and Pottier, 2009] Lee, J.-S. and Pottier, E. (2009). *Polarimetric radar imaging: from basics to applications*. CRC press, 1st edition edition.

[Lehtinen et al., 2018] Lehtinen, J., Munkberg, J., Hasselgren, J., Laine, S., Karras, T., Aittala, M., and Aila, T. (2018). Noise2noise: Learning image restoration without clean data.

[Leung and Haykin, 1991] Leung, H. and Haykin, S. (1991). The complex backpropagation algorithm. *IEEE Transactions on Signal Processing*, 39(9):2101–2104.

[Li and Adalı, 2008] Li, H. and Adalı, T. (2008). Complex-valued adaptive signal processing using nonlinear functions. *EURASIP Journal on Advances in Signal Processing*, 2008(1):765615.

[Li et al., 2020] Li, X., Sun, Q., Li, L., Liu, X., Liu, H., Jiao, L., and Liu, F. (2020). SSCV-GANs: Semi-supervised complex-valued GANs for PolSAR image classification. *IEEE Access*, 8:146560–146576.

[Li et al., 2018a] Li, Y., Chen, Y., Liu, G., and Jiao, L. (2018a). A novel deep fully convolutional network for PolSAR image classification. *Remote Sensing*, 10(12):1984.

[Li et al., 2018b] Li, Y., Chen, Y., Liu, G., and Jiao, L. (2018b). A Novel Deep Fully Convolutional Network for PolSAR Image Classification. *Remote Sensing*, 10(12).

[Lienou et al., 2010] Lienou, M., Maitre, H., and Datcu, M. (2010). Semantic annotation of satellite images using latent Dirichlet allocation. *IEEE Geoscience and Remote Sensing Letters*, 7(1):28–32.

[Liu et al., 2017] Liu, S., Xu, M., Wang, J., Lu, F., Zhang, W., Tian, H., and Chang, G.-K. (2017). A multilevel artificial Neural Network nonlinear equalizer for millimeter-wave mobile fronthaul systems. *Journal of Lightwave Technology*, 35(20):4406–4417.

[Liu et al., 2019] Liu, X., Jiao, L., and Liu, F. (2019). PolSF: PolSAR image dataset on San Francisco. *arXiv preprint arXiv:1912.07259*.

[Liu et al., 2014] Liu, Y., Huang, H., and Huang, T. (2014). Gain parameters based complex-valued backpropagation algorithm for learning and recognizing hand gestures. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 2162–2166. IEEE.

[Long et al., 2015] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.

[Lopes et al., 1993] Lopes, A., Nezry, E., Touzi, R., and Laur, H. (1993). Structure detection and statistical adaptive speckle filtering in SAR images. *International Journal of Remote Sensing*, 14(9):1735–1758.

[Maas et al., 2013] Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve Neural Network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*.

[Mandic et al., 2009] Mandic, D., Javidi, S., Goh, S., Kuh, A., and Aihara, K. (2009). Complex-valued prediction of wind profile using augmented complex statistics. *Renewable Energy*, 34(1):196–201.

[Marmanis et al., 2016a] Marmanis, D., Datcu, M., Esch, T., and Stilla, U. (2016a). Deep Learning Earth observation classification using Imagenet pretrained networks. *IEEE Geoscience and Remote Sensing Letters*, 13(1):105–109.

[Marmanis et al., 2018] Marmanis, D., Schindler, K., Wegner, J., Galliani, S., Datcu, M., and Stilla, U. (2018). Classification with an edge: Improving semantic image segmentation with boundary detection. *ISPRS Journal of Photogrammetry and Remote Sensing*, 135:158–172.

[Marmanis et al., 2016b] Marmanis, D., Wegner, J. D., Galliani, S., Schindler, K., Datcu, M., and Stilla, U. (2016b). Semantic segmentation of aerial images with an ensemble of CNSS. In *ISPRS Congress*, volume III-3, pages 473–480. Copernicus Publications.

[Marseet and Sahin, 2017] Marseet, A. and Sahin, F. (2017). Application of complex-valued convolutional Neural Network for next generation wireless networks. In *2017 IEEE Western New York Image and Signal Processing Workshop (WNYISPW)*, pages 1–5. IEEE.

[Mascarenhas, 1997] Mascarenhas, N. D. (1997). An overview of speckle noise filtering in SAR images. In *Image Processing Techniques, First Latino-American Seminar on Radar Remote Sensing*, volume 407, pages 71–79.

[Matlacz and Sarwas, 2018] Matlacz, M. and Sarwas, G. (2018). Crowd counting using complex convolutional Neural Network. In *2018 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, pages 88–92. IEEE.

[Matthès et al., 2021] Matthès, M. W., Bromberg, Y., de Rosny, J., and Popoff, S. M. (2021). Learning and avoiding disorder in multimode fibers. *Phys. Rev. X*, 11:021060.

[McCulloch and Pitts, 1943] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

[McGill et al., 1978] McGill, R., Tukey, J. W., and Larsen, W. A. (1978). Variations of box plots. *The American Statistician*, 32(1):12–16.

[McMahan et al., 2013] McMahan, H. B., Holt, G., Sculley, D., Young, M., Ebner, D., Grady, J., Nie, L., Phillips, T., Davydov, E., Golovin, D., et al. (2013). Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230.

[Milner and Milner, 1996] Milner, P. M. and Milner, B. (1996). Donald Olding Hebb. 22 July 1904-20 August 1985. *Biographical Memoirs of Fellows of the Royal Society*, 42:193–204.

[Minin et al., 2012] Minin, A., Chistyakov, Y., Kholodova, E., Zimmermann, H.-G., and Knoll, A. (2012). Complex valued open Recurrent Neural Network for power transformer modeling. *International Journal of Applied Mathematics and Informatics*, 6:41–48.

[Minsky and Papert, 1969] Minsky, M. and Papert, S. (1969). *Perceptrons; an Introduction to Computational Geometry*. MIT Press.

[Minsky, 1991] Minsky, M. L. (1991). Logical versus analogical or symbolic versus connectionist or neat versus scruffy. *AI Magazine*, 12(2):34.

[Miyauchi and Seki, 1992] Miyauchi, M. and Seki, M. (1992). Interpretation of optical flow through Neural Network learning. In *[Proceedings] Singapore ICCS/ISITA92*, pages 1247–1251. IEEE.

[Miyauchi et al., 1993] Miyauchi, M., Seki, M., Watanabe, A., and Miyauchi, A. (1993). Interpretation of optical flow through complex Neural Network. In *International Workshop on Artificial Neural Networks*, pages 645–650. Springer.

[Molini et al., 2022] Molini, A. B., Valsesia, D., Fracastoro, G., and Magli, E. (2022). Speckle2Void: Deep Self-Supervised SAR Despeckling with Blind-Spot

Convolutional Neural Networks. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–17.

[Mönning and Manandhar, 2018] Mönning, N. and Manandhar, S. (2018). Evaluation of Complex-Valued Neural Networks on Real-Valued Classification Tasks. *arXiv preprint arXiv:1811.12351*.

[Montúfar et al., 2014] Montúfar, G., Pascanu, R., Cho, K., and Bengio, Y. (2014). On the number of linear regions of deep neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2924–2932, Cambridge, MA, USA. MIT Press.

[Moreno-Díaz and Moreno-Díaz, 2007] Moreno-Díaz, R. and Moreno-Díaz, A. (2007). On the legacy of W.S. McCulloch. *Bio Systems*, 88 3:185–90.

[Morgan and Bourlard, 1990] Morgan, N. and Bourlard, H. (1990). Advances in neural information processing systems 2. chapter generalization and parameter estimation in feedforward nets: some experiments.

[Nair and Hinton, 2010] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Icml*.

[Nishikawa et al., 2006] Nishikawa, I., Iritani, T., and Sakakibara, K. (2006). Improvements of the traffic signal control by complex-valued Hopfield networks. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 459–464. IEEE.

[Nishikawa et al., 2005] Nishikawa, I., Sakakibara, K., Iritani, T., and Kuroe, Y. (2005). 2 types of complex-valued Hopfield networks and the application to a traffic signal control. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 782–787. IEEE.

[Nitta, 2004] Nitta, T. (2004). Orthogonality of decision boundaries in complex-valued Neural Networks. *Neural computation*, 16(1):73–97.

[Nwankpa et al., 2020] Nwankpa, C., Ijomah, W., Gachagan, A., and Marshall, S. (2020). Activation functions: comparison of trends in practice and research for deep learning. In *INCCST 2020: 2nd International Conference on Computational Sciences and Technology*, pages 124 – 133.

[Olanrewaju et al., 2011] Olanrewaju, R., Khalifa, O., Abdulla, A., and Khedher, A. M. (2011). Detection of alterations in watermarked medical images using Fast Fourier Transform and Complex-Valued Neural Network. In *2011 4th IEEE International Conference on Mechatronics (ICOM)*, pages 1–6.

[Ollila, 2008] Ollila, E. (2008). On the circularity of a complex random variable. *IEEE Signal Processing Letters*, 15:841–844.

[Oyama and Hirose, 2018] Oyama, K. and Hirose, A. (2018). Adaptive phase-singular-unit restoration with entire-spectrum-processing complex-valued Neural Networks in interferometric SAR. *Electronics Letters*, 54(1):43–45.

[Parikh et al., 2020] Parikh, H., Patel, S., and Patel, V. (2020). Classification of SAR and PolSAR images using deep learning: A review. *International Journal of Image and Data Fusion*, 11(1):1–32.

[Parrilli et al., 2012] Parrilli, S., Poderico, M., Angelino, C. V., and Verdoliva, L. (2012). A Nonlocal SAR Image Denoising Algorithm Based on LLMMSE Wavelet Shrinkage. *IEEE Transactions on Geoscience and Remote Sensing*, 50(2):606–616.

[Patel and Patel, 2020] Patel, R. and Patel, S. (2020). A comprehensive study of applying convolutional neural network for computer vision. *International Journal of Advanced Science and Technology*, 6:2161–2174.

[Pearlmutter and Siskind, 2007] Pearlmutter, B. A. and Siskind, J. M. (2007). Lazy multivariate higher-order forward-mode ad. *ACM SIGPLAN Notices*, 42(1):155–160.

[Peker et al., 2015] Peker, M., Sen, B., and Delen, D. (2015). A novel method for automated diagnosis of epilepsy using complex-valued classifiers. *IEEE Journal of Biomedical and Health Informatics*, 20(1):108–118.

[Picinbono, 1996] Picinbono, B. (1996). Second-order complex random vectors and Normal distributions. *IEEE Transactions on Signal Processing*, 44(10):2637–2640.

[Popa, 2017] Popa, C.-A. (2017). Complex-valued convolutional Neural Networks for real-valued image classification. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 816–822. IEEE.

[Popa, 2018] Popa, C.-A. (2018). Deep hybrid real-complex-valued convolutional Neural Networks for image classification. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE.

[Pottier and Ferro-Famil, 2012] Pottier, E. and Ferro-Famil, L. (2012). PolSARPro V5.0: An ESA educational toolbox used for self-education in the field of POLSAR and POL-INSAR data analysis. In *2012 IEEE International Geoscience and Remote Sensing Symposium*, pages 7377–7380.

[Qin et al., 2021] Qin, X., Zou, H., Yu, W., and Wang, P. (2021). Superpixel-Oriented Classification of PolSAR Images Using Complex-Valued Convolutional

Neural Network Driven by Hybrid Data. *IEEE Transactions on Geoscience and Remote Sensing*, 59(12):10094–10111.

[Rall, 1983] Rall, L. B. (1983). Differentiation and generation of taylor coefficients in Pascal-SC. In *A New Approach to Scientific Computation*, pages 291–309. Elsevier.

[Rall, 1986] Rall, L. B. (1986). The arithmetic of differentiation. *Mathematics Magazine*, 59(5):275–282.

[Ramachandran et al., 2018] Ramachandran, P., Zoph, B., and Le, Q. V. (2018). Searching for activation functions. *ArXiv*, abs/1710.05941.

[Rasti et al., 2022] Rasti, B., Chang, Y., Dalsasso, E., Denis, L., and Ghamisi, P. (2022). Image restoration for remote sensing: Overview and toolbox. *IEEE Geoscience and Remote Sensing Magazine*, 10(2):201–230.

[Rayleigh, 1880] Rayleigh, L. (1880). XII. On the resultant of a large number of vibrations of the same pitch and of arbitrary phase. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 10(60):73–78.

[Real Academia Española, 2022] Real Academia Española (2022). Diccionario de la lengua española.

[Reddi et al., 2018] Reddi, S. J., Kale, S., and Kumar, S. (2018). On the convergence of adam and beyond. *ArXiv*, abs/1904.09237.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional Networks for Biomedical Image Segmentation. In Navab, N., Hornegger, J., Wells, W. M., and Frangi, A. F., editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham. Springer International Publishing.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

[Scardapane et al., 2018] Scardapane, S., Van Vaerenbergh, S., Hussain, A., and Uncini, A. (2018). Complex-valued Neural Networks with nonparametric activation functions. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 4(2):140–150.

[Schreier and Scharf, 2010] Schreier, P. J. and Scharf, L. L. (2010). *Statistical Signal Processing of Complex-Valued Data*. Cambridge University Press.

[Selmo et al., 2018] Selmo, C., Barrachina, J. A., and Sutton, N. (2018). rn-itba-2018 overfitting regularization tensorflow. https://github.com/rn-itba-2018/5-Overfiting-Regularization-Tensorflow.

[Sepasi et al., 2017] Sepasi, S., Reihani, E., Howlader, A. M., Roose, L. R., and Matsuura, M. M. (2017). Very short-term load forecasting of a distribution system with high PV penetration. *Renewable energy*, 106:142–148.

[Shang et al., 2019] Shang, R., Wang, G., A. Okoth, M., and Jiao, L. (2019). Complex-Valued Convolutional Autoencoder and Spatial Pixel-Squares Refinement for Polarimetric SAR Image Classification. *Remote Sensing*, 11(5).

[Sherwin et al., 1962] Sherwin, C. W., Ruina, J. P., and Rawcliffe, R. D. (1962). Some Early Developments in Synthetic Aperture Radar Systems. *IRE Transactions on Military Electronics*, MIL-6(2):111–115.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The journal of machine learning research*, 15:1929–1958.

[Stinchcombe and White, 1989] Stinchcombe, M. and White, H. (1989). Universal approximation using feedforward networks with non-sigmoid hidden layer activation functions. In *IJCNN International Joint Conference on Neural Networks*.

[Sun et al., 2019] Sun, Q., Li, X., Li, L., Liu, X., Liu, F., and Jiao, L. (2019). Semi-supervised complex-valued GAN for polarimetric SAR image classification. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS 2019)*, pages 3245–3248. IEEE.

[Szegedy et al., 2015] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

[Taillade, 2020] Taillade, T. (2020). *A new strategy for change detection in SAR time-series : application to target detection*. PhD thesis, Paris-Saclay University. PhD Thesis Electronique et Optoélectronique, Nano- et Microtechnologies université Paris-Saclay 2020.

[Thomas et al., 2017] Thomas, A. J., Petridis, M., Walters, S. D., Gheytassi, S. M., and Morgan, R. E. (2017). Two Hidden Layers are Usually Better than One. In *Engineering Applications of Neural Networks*, Communications in Computer and Information Science, pages 279–290, Cham. Springer International Publishing.

[Tieleman and Hinton, 2012] Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop, coursera: Neural networks for machine learning. *University of Toronto, Technical Report*, 6.

[Torralba and Efros, 2011] Torralba, A. and Efros, A. A. (2011). Unbiased look at dataset bias. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, page 1521–1528, USA. IEEE Computer Society.

[Trabelsi et al., 2017] Trabelsi, C., Bilaniuk, O., Zhang, Y., Serdyuk, D., Subramanian, S., Santos, J. F., Mehri, S., Rostamzadeh, N., Bengio, Y., and Pal, C. J. (2017). Deep complex networks. *arXiv preprint arXiv:1705.09792*.

[Tsuzuki et al., 2013] Tsuzuki, H., Kugler, M., Kuroyanagi, S., and Iwata, A. (2013). An approach for sound source localization by complex-valued Neural Network. *IEICE Transactions on Information and Systems*, 96(10):2257–2265.

[Turing, 1950] Turing, A. (1950). Machinery and intelligence. *Mind: A Quarterly Review of Psychology and Philosophy*, 59:236.

[Vasile and Totir, 2012] Vasile, G. and Totir, F. C. (2012). Circularity of complex stochastic models in PolSAR and multi-pass InSAR images. In *2012 IEEE International Geoscience and Remote Sensing Symposium*, pages 3720–3723.

[Vieillard, 2018] Vieillard, G. (2018). Reconnaissance de signaux radar par apprentissage profond. Technical Report RA 2/26398 DEMR, ONERA - Département Électromagnétisme et Radar (DEMR).

[Virtue et al., 2017] Virtue, P., Stella, X. Y., and Lustig, M. (2017). Better than real: Complex-valued neural nets for MRI fingerprinting. In *2017 IEEE international conference on image processing (ICIP)*, pages 3953–3957. IEEE.

[Walessa and Datcu, 2000] Walessa, M. and Datcu, M. (2000). Model-based despeckling and information extraction from SAR images. *IEEE Transactions on Geoscience and Remote Sensing*, 38(5):2258–2269.

[Wang and Raj, 2017] Wang, H. and Raj, B. (2017). On the Origin of Deep Learning. *arXiv e-prints*, page arXiv:1702.07800.

[Wang et al., 2017] Wang, P., Zhang, H., and Patel, V. M. (2017). SAR Image Despeckling Using a Convolutional Neural Network. *IEEE Signal Processing Letters*, 24(12):1763–1767.

[Wiley, 1954] Wiley, C. (U.S. Patent 3196436, Aug. 1954). Pulsed doppler radar methods and apparatus.

[Wilkes and Wade, 1997] Wilkes, A. L. and Wade, N. J. (1997). Bain on neural networks. *Brain and Cognition*, 33(3):295–305.

[Williamson et al., 1989] Williamson, D. F., Parker, R. A., and Kendrick, J. S. (1989). The Box Plot: A simple visual method to interpret data. *Annals of internal medicine*, 110(11):916–921.

[Wilmanski et al., 2016] Wilmanski, M., Kreucher, C., and Hero, A. (2016). Complex input Convolutional Neural Networks for wide angle SAR ATR. In *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 1037–1041.

[Wirtinger, 1927] Wirtinger, W. (1927). Zur formalen Theorie der Funktionen von mehr komplexen Veränderlichen. *Mathematische Annalen*, 97(1):357–375.

[Wu et al., 2016] Wu, W., Li, X., Guo, H., Ferro-Famil, L., and Zhang, L. (2016). Noncircularity Parameters and Their Potential Applications in UHR MMW SAR Data Sets. *IEEE Geoscience and Remote Sensing Letters*, 13(10):1547–1551.

[Xia et al., 2018] Xia, G.-S., Bai, X., Ding, J., Zhu, Z., Belongie, S., Luo, J., Datcu, M., Pelillo, M., and Zhang, L. (2018). DOTA: A Large-Scale Dataset for Object Detection in Aerial Images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Xie et al., 2002] Xie, H., Pierce, L., and Ulaby, F. (2002). SAR speckle reduction using wavelet denoising and Markov random field modeling. *IEEE Transactions on Geoscience and Remote Sensing*, 40(10):2196–2212.

[Xie et al., 2020] Xie, W., Ma, G., Zhao, F., Liu, H., and Zhang, L. (2020). PolSAR image classification via a novel semi-supervised recurrent complex-valued convolution neural network. *Neurocomputing*, 388:255–268.

[Yamashita et al., 2018] Yamashita, R., Nishio, M., Do, R. K. G., and Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, 9(4):611–629.

[Yu and Koltun, 2015] Yu, F. and Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions.

[Zafar et al., 2018] Zafar, I., Tzanidou, G., Burton, R., Patel, N., and Araujo, L. (2018). *Hands-on convolutional neural networks with TensorFlow: Solve computer vision problems with modeling in TensorFlow and Python*. Packt Publishing Ltd, Birmingham, UK.

[Zeiler et al., 2013] Zeiler, M., Ranzato, M., Monga, R., Mao, M., Yang, K., Le, Q., Nguyen, P., Senior, A., Vanhoucke, V., Dean, J., and Hinton, G. (2013). On rectified linear units for speech processing. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3517–3521.

[Zeiler, 2012] Zeiler, M. D. (2012). ADADELTA: An Adaptive Learning Rate Method.

[Zeiler et al., 2010] Zeiler, M. D., Krishnan, D., Taylor, G. W., and Fergus, R. (2010). Deconvolutional networks. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2528–2535. IEEE.

[Zhang, 2015] Zhang, C. (2015). Mocha.jl: Deep Learning for Julia. https://developer.nvidia.com/blog/mocha-jl-deep-learning-julia/.

[Zhang and Mandic, 2016] Zhang, H. and Mandic, D. P. (2016). Is a complex-valued stepsize advantageous in complex-valued gradient learning algorithms? *IEEE Transactions on Neural Networks and Learning Systems*, 27(12):2730–2735.

[Zhang and Wu, 2017] Zhang, J. and Wu, Y. (2017). A new method for automatic sleep stage classification. *IEEE Transactions on Biomedical Circuits and Systems*, 11(5):1097–1110.

[Zhang et al., 2018] Zhang, Q., Yuan, Q., Li, J., Yang, Z., and Ma, X. (2018). Learning a Dilated Residual Network for SAR Image Despeckling. *Remote Sensing*, 10(2).

[Zhang et al., 2017] Zhang, Z., Wang, H., Xu, F., and Jin, Y.-Q. (2017). Complex-valued convolutional neural network and its application in polarimetric SAR image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(12):7177–7188.

[Zhao et al., 2019a] Zhao, J., Datcu, M., Zhang, Z., Xiong, H., and Yu, W. (2019a). Contrastive-Regulated CNN in the Complex Domain: A Method to Learn Physical Scattering Signatures from Flexible PolSAR Images. *IEEE Transactions on Geoscience and Remote Sensing*, 57(12):10116–10135.

[Zhao et al., 2019b] Zhao, J., Datcu, M., Zhang, Z., Xiong, H., and Yu, W. (2019b). Learning physical scattering patterns from PolSAR images by using complex-valued CNN. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS 2019)*, pages 10019–10022. IEEE.

[Zhou et al., 2016] Zhou, Y., Wang, H., Xu, F., and Jin, Y.-Q. (2016). Polarimetric SAR image classification using deep convolutional neural networks. *IEEE Geoscience and Remote Sensing Letters*, 13(12):1935–1939.