Applicability of complex-valued convolutional neural networks to complex-valued data classification

Omair Khalid

School of Engineering and Physical Sciences
Heriot-Watt University

Supervised by:
Prof. Andrew Wallace

A Thesis Submitted for the Degree of
MSc Erasmus Mundus in Vision and Robotics (VIBOT)

· 2018 ·

Abstract

Complex numbers are aptly used to describe natural phenomena: wind (speed and direction), electromagnetic waves, MRI data etc. However, CNNs are dominated by the use of real numbers. If we aim to better exploit the camplex-natured data in order to improve the classsification, the CNN should be able to process the complex data. In this work, we explore the design choices and challenges faced while working with $\mathbb{C}$-CNNs. We propose the architectures of and compare the performance of $\mathbb{C}$-CNN and $\mathbb{R}$-CNNs on a synthetic data set (MNIST+P) and two real world radar-datasets. The results are inconclusive to establish the relevance of $\mathbb{C}$-CNNs.

The only way out is through.

Robert Frost

Contents

List of Figures

List of Tables

Acknowledgments

- God.
- Parents, the big sibling and the little sibling.
- My supervisor, Prof. Andrew Wallace, for his support. Marcel Sheeny de Moraes for his guidance throughout the journey.
- VIBOTians: Wajee, Albert, Dani, Thomas, Savinien, Anirudh, Darja, and the Vision Lab-mates without Clear Vision: Shubham, Alpha, Ziyang and Zaal.
- Saptarshi for his generosity.
- Christina Kuhl and Ahmed Riaz Khan for their endless supply of positivity.
- Rao, Ahdab, and Hammad for being there.
- Myself, for getting it done.

Chapter 1

Introduction

1.1 Motivation

Complex-valued Convolutional Neural Networks (ℂ-CNNs) are a class of Convolutional Neural Networks (CNNs) that incorporates complex number representations in their operation. This may mean that either data, internal parameters (weights), or both data and weights are complex in nature. We motivate our problem by first describing why CNNs are a suitable choice for image classification, followed by why complex numbers should be incorporated into CNNs for complex-natured data.

CNNs have emerged to be one of the most powerful tool in computer vision for the tasks including, but not limited to, image classification, and segmentation. CNNs make use of the compositional hierarchy of features present in natural images, i.e. low-level elements coming together to form higher-level representations, while enjoying the benefits of sparse connectivity, weight-sharing, and invariance to translations (further explained in Section 4.1) They gained tremendous popularity after they were used by Krizhevsky *et al*. (2012) [22] used them to win the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 by achieving record-breaking results. Recent years have seen major advancements in their performance which includes techniques like Batch Normalization [20], Drop-Out regularization, different architecture designs (AlexNet, ResNet) etc. All this makes CNNs the suitable choice for data classification problem.

Complex numbers can help us achieve a closer representation of natural phenomena as compared to only real-valued numbers. Complex numbers lend themselves to neuronal models in a neural network that exhibit the properties of a human brain (Reichert and Serre (2013) [28], represent wind's speed and direction [**?**], and electromagnetic waves [16], among others.

The idea of using complex-valued representations which closely mimic natural phenomena in a framework that can exploit such data (ℂ-CNN) is a promising one, as demonstrated in [**?**], [35].

According to literature, one of the advantages of ℂ-NNs can provide better guard against over-fitting phenomenon, leading to better generalization. This idea is also put forward by Hirose and Yoshida (2012) [16] who explain that the benefit of complex-valued neural networks (ℂ-NNs) compared to real-valued neural networks (ℝ-NNs) lies in the restricted flexibility in learning brought by the nature of multiplication, as it entails rotation and scaling, not just scaling as in real-valued case. This grants the network relief from "ineffective degrees of freedom" that can lead to poor generalization. Guberman (2016) [12] also reports better generalization for ℂ-CNNs as compared to ℝ-CNNs. Hence, if better generalization is desired, ℂ-CNNs might have the answer.

The recent success of ℂ-CNNs with Polarimetric Synthetic Aperture (SAR) data in various classification tasks ([35], [13], [32]), audio transcription, and speech spectrum prediction [31], against ℝ-CNNs presents us with a strong case of their potential advantages when the input data is complex-natured itself. Although the research of ℂ-NNs is not new, the recent advancements in CNNs and general deep learning architectures offer us an opportunity to further investigate how complex number representations can be useful in this new light.

1.2 Objectives

We aim to experiment and analyse if ℂ-CNNs can perfom better than ℝ-CNNs when the input is complex-natured data. The sub-tasks are:

1. Review literature concerning Complex-Valued Convolution Networks (ℂ-CNNs) to explore design possibilities.
2. Propose and implement comparable architectures ℂ-CNN and ℝ-CNN in the wake of recent deep learning advancements.
3. Experiment with an in-house complex-natured radar dataset to establish the benefit, or lack thereof, of ℂ-CNNs in terms of generalization capability and test accuracy as compared to ℝ-CNNs.

1.3 Document structure

This document is organized as follows:

- **Chapter 2**

- **Chapter 3**
- **Chapter 4**
- **Chapter 5**
- **Chapter 6**

Chapter 2

Literature Review

Complex-valued neural networks have been widely investigated since before the deep learning boom and have been found useful in the fields of adpative designing of patch antennas, neurophysiological analysis, and communications [16].

Increased incorporation of complex-valued units in recent works of Recurrent Neural Networks ([5], [33], [8]) and computer vision tasks([27], [6], [34]) has brought significant attention to the virtues of complex-valued representations.

One of the most important papers in this domain is [16], which gives a mathematical motivation for complex-valued convolutional neural networks, showing that they can be seen as nonlinear multiwavelet packets, thus making the mathematical analysis from the signal processing domain available for a rigorous formulation of the properties of complex-valued convolutional networks. In [12] a wavelet scattering network is proposed, which uses complex numbers

Complex numbers can be better utilized to represent natural phenomena. Utilizing complex-valued representations, Reichert and Serre (2013) [28] developed neuronal units of a neural network inspired from a model of neuron in a human brain. In the Spiking Neuron Model, a neuron's output is expressed in terms of its firing rate and phase (relative time of its activity). The authors equated the firing rate with the magnitude and the neuron's phase as the phase of the neuron's output. Their implementation captures the concept of Neuronal Synchrony, which says that a neuron's time of firing (phase) also plays a role in information processing. The firing of a neuron happens on the basis of phases of the inputs such that synchronous (similar in phase) inputs produce an output while the asynchronous phases inhibits the output. This way the network tries to synchronize the phases of those inputs whose activations result in correct outputs. Takahiro and Takashi (2010) developed an output prediction system of a wind power generation using a $\mathbb{C}$-CNN where they expressed the wind information (wind speed and direction) by complex numbers, and use them as input and demonstrated that $\mathbb{C}$-CNN

performed better as compared to $\mathbb{R}$-CNN, owing to the fact that complex nature of the data had a physical meaning behind it.

SARs are typically mounted on a moving platform, such as an aircraft or spacecraft, in order to capture 2D or 3D images of objects including landscapes. The capability of radars to operate in all-weather day-and-night conditions and make high and very high resolution images make them suitable for target classification, reconnaissance, surveillance, etc. Chen *et al.* propose A-ConvNet $\mathbb{R}$-CNN architecture which deviates from conventional $\mathbb{R}$-CNN in that it does away with the Fully Connected (FC) layers in order to decrease the number of parameters in an attempt to decrease over-fitting. They report state-of-the-art performance (99% average accuracy) on classification of ten classes in teh Moving and Stationary Target Acquisition and Recognition (MSTAR) dataset. This suggests the suitability of CNNs in classification of radar data.

The recent re-introduction of complex-valued neural networks in the classification of Synthetic Aperture Radar (SAR) data has been very promising in the wake of marked advancements in the theory of deep learning. The data is complex by its very nature, hence opens up the interesting opportunity to employ $\mathbb{C}$-CNNs to better make use of phase information in various problems.

Wilmanski *et al.* (2016) [32] explore the suitability of using $\mathbb{C}$-CNNs for Automatic Target Recognition for complex-valued SAR data. Although their $\mathbb{C}$-CNN model had only one complex-valued layer (complex weights) in their complex-valued variant architecture, it outperformed (99.21% accuracy) the state-of-the-art $\mathbb{R}$-CNN network (87.30% accuracy). In the dataset they used (GOTCHA [3]), they also pointed out how the phase surrounding an object had a distinctive structure, pointing to the potential importance of phase information in classification tasks.

Polarimetric SAR (PolSAR) uses microwaves with different polarisations to measure the distance to ground and the reflectance of a target [14]. Usually, PolSARs are installed on aeroplanes to It is of interest in a variety of applications to be able to correctly classify images according to their content. One of the traditional techniques for classification involving PolSAR data is the use of (real-valued) Multilayer Perceptrons (MLP). Inspired by the success of MLPs in computer vision, Hänsch and Hellwich (2009) [14] employed the $\mathbb{C}$-NNs to classify the complex-valued Polarimetric Synthetic Aperture Radar (PolSAR) data to perform a 3-class pixel-wise classification (forest, fields and urban areas). The authors test their architectures using different error functions for $\mathbb{C}$-NNs, and compare $\mathbb{C}$-NNs with their real-valued counterparts. The results conclude that $\mathbb{C}$-NNs outperformed $\mathbb{R}$-CNNs in that particular problem. However, the input data to both type of CNNs is not preprocessed the same way. The same authors go ahead to tackle the object-classification problem using complex-valued convolutional neural networks ($\mathbb{C}$-CNNs) and $\mathbb{R}$-CNNs [13]. They show that $\mathbb{C}$-CNNs, with only one

complex-convolutional layer, outperform the $\mathbb{C}$-NNs in the cases where the number of neurons in single convolutional layer exceeded 10. Zhang *et al.* (2017) [35] leverage magnitude as well as the phase of the PolSAR data to classify different terrains on the Flevoland (3 classes) and Oberpfaffenhofen datasets (15 classes). Compared to $\mathbb{R}$-CNN, $\mathbb{C}$-CNN performs better on both the datasets while having approximately same number of parameters.

Guberman (2016) [12] in his work report the problems of convergence in complex-valued deep networks, presents a new activation function ($zReLU$), and demonstrates that complex-valued networks can better avoid over-fitting in training.

Chiheb *et al.* (2018) [31] compare the performance of different architectures of the $\mathbb{C}$-CNNs and $\mathbb{R}$-CNNs on the tasks of image recognition, music transcription, and speech spectrum prediction. $\mathbb{C}$-CNNs were reported to perform comparably to $\mathbb{R}$-CNNs for the first task, and achieve state-of-the-art performance on the two tasks while beating $\mathbb{R}$-CNNs. The authors also contribute the extension of Batch Normalization (BN) and Weight Initialization to complex domain. They report a lot of failed experiments when they used the modReLU and zReLU complex activation functions in the real-valued image classification problem due to the appearance of NaN(Not-A-Number) values caused by their non-complex-differentiable nature.

The literature review brings to light the challenges of designing a $\mathbb{C}$-CNNs which include the choice of activation function and loss function while ensuring convergence,

Chapter 3

Background

This chapter presents the pre-requisite knowledge for understanding the challenges of the problem. The following subsections explain about the Radar technology and Complex Numbers theory.

3.1 Radar technology

Radar stands for Radio Detection and Range is a sensing technology that uses radio waves to detect range, angle, and velocity of objects. Radar technology was first developed to aid enemy target detection in World War II, and extended to other applications such as monitoring precipitation (meteorology), mapping componsition of Earth's crust (geology), monitoring speeds of road vehicles, among others. Nowadays, radars are also widely used in automotive industry in Advanced Driver Assistance Systems (ADAS) for automatic emergency braking system (AEBS), adaptive cruise control (ACC), automatic parking, among others.

3.1.1 Working principle of radar

Radars emit electromagnetic waves (continuous or pulsed) and detect the distance of a target object based on the amount of time it took for the reflected wave to be received. In this respect the equation governing this process is:

$$r = \frac{vt}{2} \tag{3.1}$$

where v is the constant for speed of light, t is the time taken for a round trip of the emitted signal to come back, and r is the range. Figure **??** illustrates this process.

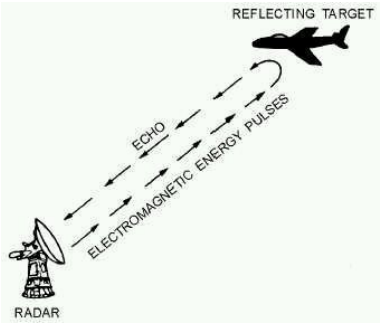

Figure 3.1: Representation of Radar working scheme

The direction from which maximum amplitude echo signal is received dictates target's location which is measured in angle. To measure range and location of moving objects, Doppler Effect is utilized.

The simplified model of radar gives us the maximum range R_{max} as,

$$\begin{aligned} R_{max} &= \left[\frac{P_t G A_e \sigma}{(4\pi)^2 Smin}\right], where \\ P_t &= \text{Transmitted power, in Watts,} \\ G &= \text{Antenna gain,} \\ A_e &= \text{Antenna effective aperture in } m^2, \\ \sigma &= \text{Radar cross section of the target, in } m^2, \\ S_{min} &= \text{Minimum detectable signal, in Watts.} \end{aligned} \tag{3.2}$$

All the above parameters, except σ, are to some extent under the control of the radar designer. As this model does not exactly capture the nature of the physical world, a number of other factors play a role in deteriorating the performance of the radar, namely, variance in performance of electronic components according to the environment, probabilistic nature of variables (S_{min}, σ), variance in environmental conditions, among others.

3.1.2 Frequency modulated continuous-wave radar

Frequency Modulated Continuous-Wave Radar (FMCW) transmits a continuous carrier signal, whose frequency modulated by periodic function such as a sinusoid or sawtooth wave to provide range data.

The internal schematics of one type of FMCW radar are presented in Figure 3.3, which will refer in this text as Type 1. Wave generator produces a Up, Down or Triangular Chirp,

as illustrated in Figure 3.2, that will be transmitted by the radar. Next, the transmitter components transmits this wave, which, after reflecting from an object, returns back to the receiver as a scaled and delayed version of the original signal. The receiver consists of two mixers: one to mix with the transmitted signal, and one to mix with the transmitted signal shifted 90 degrees. This type of mixing creates two channels for the received signal, denoted by I and Q. Both the channels are treated with a Low Pass Filter which removes the high-frequency noise, followed by an Analog-to-Digital Converter which makes it suitable for digital processing. The individual channels are added together to form one signal $I+jQ$. A windowing operation is performed to remove the side lobes before applying Fast Fourier Transform (FFT) on the signal. FFT is applied to this signal and the magnitude represents the range of the object, represented in the 2D plot in Figure 3.3.

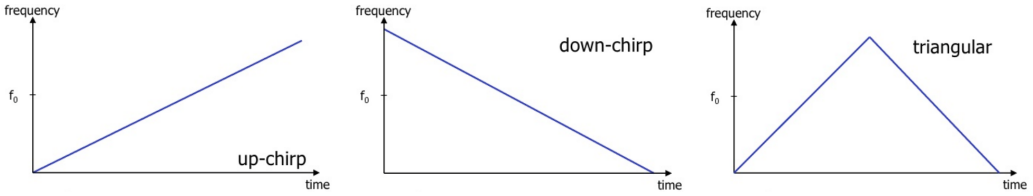

Figure 3.2: Types of chirp: Up, Down, Triangle

Another type of internal schematics found in a FMCW radar involve calculation of distance based on the difference between the instantaneous frequency of transmitted and received signals i.e. beat frequency. After receiving the signal, it is mixed with the original signal, and passed through a Low-Pass filter in order to retrieve the beat frequency. Beat frequency is given by $f_b = \left(\frac{2R}{c}\right)\left(\frac{B}{T_c}\right)$, where R is the range of the object, c is the speed of light and t_d is the round-trip delay of reflection. For a static object, the beat frequency is a single tone on the frequency spectrum, whereas multiple objects provide multiple tones. Moving objects are detected as the phase change across multiple chirps. In this text, we will refer to this type of radar as Type 2. FIGURE xyz shows the schematics of such type of a FMCW radar.

3.2 Complex numbers theory

Complex number field ($\mathbb{C}$) is an extension of the Real numbers field ($\mathbb{R}$); alternatively, $\mathbb{R} \subset \mathbb{C}$. In their Cartesian form, further described later in this chapter, they consist of a real part and an imaginary part ($z = a+ib$). If the imaginary part is 0, the complex number reduces to a real number. The term i, known as the Imaginary Quantiy, was proposed by Euler as a substitute for the quantity $\sqrt{-1}$.

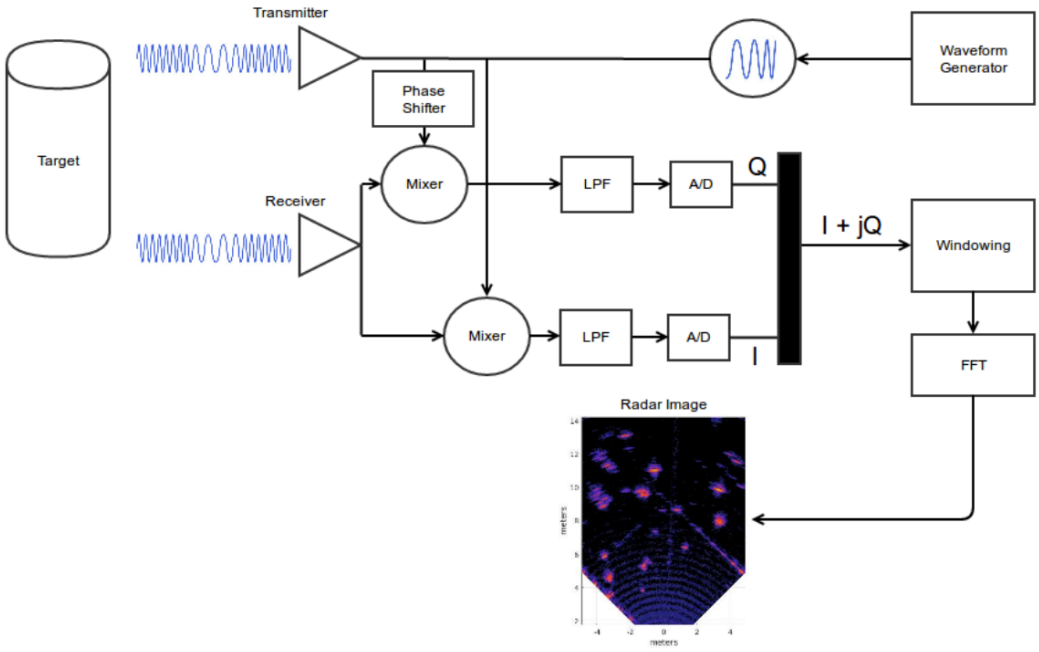

Figure 3.3: Schematics of a FMCW radar

Complex numbers are useful abstract quantities that can be used in calculations and result in physically meaningful solutions. They can be found in the fields of control theory, fluid dynamics, electromagnetism, signal analysis, and quantum mechanics, among others. One such example is the strength of an electromagnetic field. The field has both an electric and a magnetic component, so it takes a pair of real numbers (one for the intensity of the electric field, one for the intensity of the magnetic field) to describe the field strength. This pair of real numbers can be thought of as a complex number. In radar-sensing, the phase difference of the reflected wave can be interpreted as a measure of distance of the target object and the magnitude can be attributed to the surface properties of the object. where complex number representations are naturally befitting.

3.2.1 Representations of complex numbers

Cartesian form

In the Cartesian form, a complex number is represented by $z = x + iy$, where x is the imaginary part and y is the real part. They can be visualized in a rectangular plane, called the Complex Plane, where the x-axis represents the real part and the y-axis represents the imaginary part

(demonstrated in Figure 3.4).

Polar form

The Polar form of a complex number can be denoted by two quantities: the distance to a point (x,y) from the origin on the complex plane, $|z|$, and the phase of the complex number, θ, measured positive counter-clockwise from the x-axis. The conversion formulae to and from Cartesian form and Polar form are given by:

$$x = |z|cos\theta \quad , \quad y = |z|sin\theta \tag{3.3}$$

$$|z| = \sqrt{x_2 + y_2} \quad , \quad \theta = \arctan(y/x) \tag{3.4}$$

The magnitude or distance quantity $|z|$ is unique for each (x, y) in the complex domain, but the θ can have the following values: θ, $\theta \pm 2\pi$,...,, $\theta \pm 2n\pi$ where n = 1,2,3,... However, we often choose a valued between 0 and 2π and call it the *Principal Argument*.

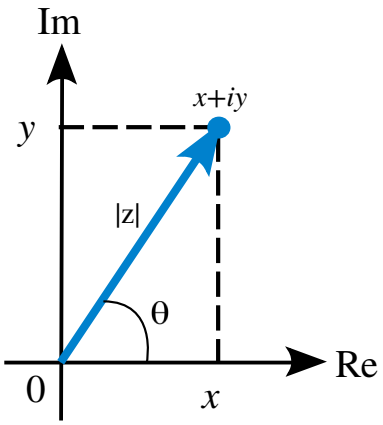

Figure 3.4: Illustration of complex number in Cartesian form ($z = x + iy$) and Polar form ($|z|, \theta$) on a Complex Plane

Exponential form

Utilizing Euler's formula given in equation 3.5, we can rewrite the Polar form of a complex number into the exponential form, defined in equation 3.6, where $r = |z|$.

$$e^{i\theta} = cos\theta + isin\theta \tag{3.5}$$

$$z = re^{i\theta} = rcos\theta + irsin\theta \tag{3.6}$$

Basic operations of complex numbers

Conjugate The complex conjugate of a complex number $z = x+iy$ is given by $x-iy$, denoted by z^* or $\bar{z}$. Conjugate of a complex number can be used to extract the real and imaginary parts of a complex number, as follows:

$$\begin{aligned} \Re(z) &= \frac{z+\bar{z}}{2} \\ \Im(z) &= \frac{z-\bar{z}}{2} \end{aligned} \tag{3.7}$$

Multiplication Let $z = a + ib$ and $t = c + id$ be two complex numbers. Utilizing the fact that $i^2 = (\sqrt{-1})^2 = -1$, the product of z and t would be as follows:

$$\begin{aligned} (a+ib)(c+id) &= ac + iad + ibc + (-1)bd \\ &= (ac - bd) + i(ad + bc) \end{aligned} \tag{3.8}$$

Division Complex division is involves rationalization of the denominator, meaning that, to get the answer, the conjugate of the denominator will be multiplied and divided to the fraction of the dividend and divisor. It is given by:

$$\begin{aligned} \frac{a+ib}{c+id} &= \frac{(a+ib)}{c+id}\frac{(c-id)}{c-id} \\ &= \frac{ac+bd}{(c^2+d^2)} + i\frac{bc-ad}{(c^2+d^2)} \end{aligned} \tag{3.9}$$

Addition and Substraction Let $z = a + ib$ and $t = c + id$ be two complex numbers. The addition and substraction of z and t would be as follows:

$$\begin{aligned} (a+ib) + (c+id) &= (a+c) + i(b+d) \\ (a+ib) - (c+id) &= (a-c) + i(b-d) \end{aligned} \tag{3.10}$$

As can be seen, the addition and substraction operations are performed component-wise.

3.2.2 Trichotomy law and comparison of two complex numbers

One way of comparing two complex numbers is by comparing the real components first, and if they are equal, compare the imaginary numbers. This is known as lexicographic ordering, a system used in dictionaries as well to define order. This can be summarized as follows:

$$(a + ib) < (c + id),\ if\ a < c\ or\ a = c\ and\ b < d \tag{3.11}$$

The law of trichotomy states that "for arbitrary real numbers a and b, exactly one of the relations $a < b$, $a = b$, $a > b$ holds" [4]. However, this relationship is not true for complex numbers and it becomes clear when dealing with multiplication of complex numbers (described below).

For an order to satisfy Trichomoty Law, $a < b$ and $0 < c$, then $ac < bc$ must be true. Now, consider that $a = 0 + i1$, and 0 is represented as $b = 0 + i0$. By lexicographic order, $0 < i$. We move forward to see that if we multiply both sides by i twice, we turn this inequality into $1 < -1$, which is a contradiction to our previous claim.

Another way to compare complex numbers is to compare their respective magnitude, instead of comparing the individual components lexicographically. But the shortcoming of this approach is that the distinction between negative and positive quantities is foregone i.e. if $z = -9 + i0$ and $t = 9 + i0$, $|z| = |t|$ even though the real part of both numbers have opposite signs.

3.2.3 Holomorphism & complex differentiability

Holomorphism guarantees that a complex-valued function is complex differentiable in the neighborhood of every point in its domain [31]. A complex function $f(z) = u(z) + iv(z)$, where $u(z)$ and $v(z)$ are real-valued functions, is complex differentiable if it satisfies the following two conditions:

1. It satisfies the Cauchy-Riemann (CR) equations, given by:

$$\frac{\delta u}{\delta x} = \frac{\delta v}{\delta y} \quad , \quad \frac{\delta u}{\delta y} = -\frac{\delta v}{\delta x} \tag{3.12}$$

2. $u(z)$ and $v(z)$ are individually differentiable at z as real functions

3.2.4 Generalized chain rule for real-valued complex function

Consider that we have a real-valued function $L(z) : \mathbb{C} \to \mathbb{R}$, where z is a complex variable $z = x + iy$ with $x, y \in \mathbb{R}$. The gradient of real-valued complex function is defined as follows:

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

If we have another complex variable $t = r + is$ where z could be expressed in terms of t, where $r, s \in \mathbb{R}$, the gradient with respect to t would be:

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

Chapter 4

Comparison of $\mathbb{R}$-CNNs with $\mathbb{C}$-CNNs

4.1 Convolutional neural networks

Convolutional Neural Networks (CNNs) (LeCun, 1989) [24] are specialized neural networks that are used for processing data that can be represented in the form of a grid or arrays e.g. 1D arrays of audio signals, three channels of 2D arrays of RGB images, or any other higher dimensional data. CNNs take their name due to the presence of a specialized kind of linear operation called convolution. CNNs are inspired by the visual cortex in the brain, which consists of alternating layers of simple and complex cells (Hubel & Wiesel, 1959, 1962) [18] [19].

In the context of image classification, CNNs make use of the inherent compositional hierarchies present in images such that higher-level features are obtained by composing lower-level ones. Additionally, there are three key ideas behind ConvNets that makes them attractive for learning: sparse interactions, shared weights, and equivariant representations [10]. In traditional neural networks, one learned parameter interacts with one input unit in order to produce an output. In the case of images where there are millions of pixels, this approach becomes extremely expensive. In this respect, CNNs offer sparse interactions between inputs and the outputs in the sense that kernels of very few parameters need to be learned to extract features from an input of, let's say, a million pixels. This reduces the memory requirements and improve statistical efficiency. Parameter sharing refers to the use of the same parameters for than one function in the model. When kernels are learned in a CNN, the same kernel is used for different spatial locations of the input rather than learning different parameters for different spatial locations. This further reduces the memory requirements. CNNs also grant equivariance to

translations, meaning that if the presence of a feature translates in an image, the output of a convolutional layer will also move by the same amount. This behavior is useful if we want to detect the same feature in different parts of the image. An example of a CNN, provided in LeCun (2015) paper [23], is given in Figure 4.1.

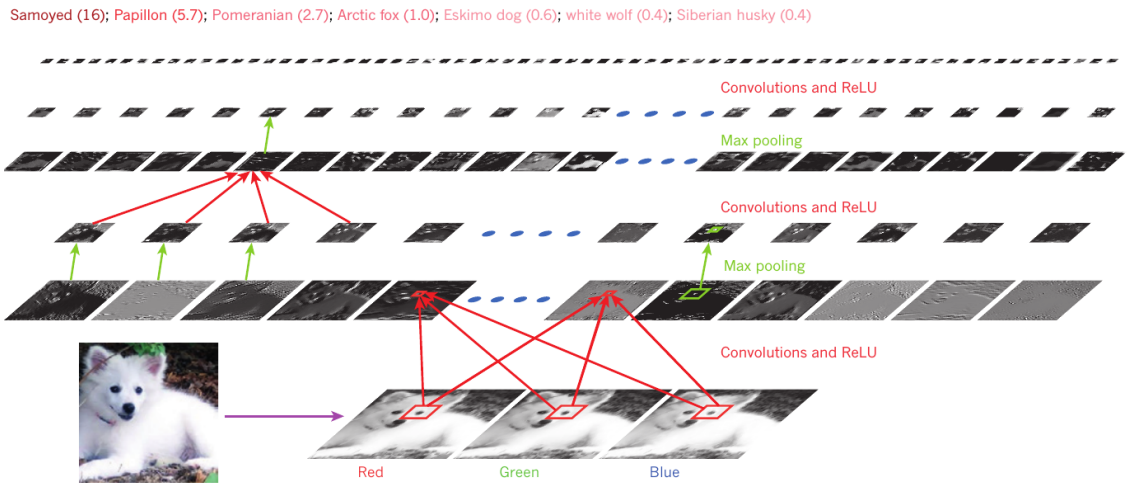

Figure 4.1: The outputs of each layer of a CNN can be seen as applied to a picture of Samoyed Dog. The RGB channels are fed into the network, and is met by two instances of Convolution, Rectified Linear Unit (RelU), and Max-pooling layer before reaching the output layer. Each image is a feature map corresponding to the output of the learned features at each image position. The output layer provides the probabilities of the image being a specific class. [1]

In the following section, the building blocks of a real-valued CNN (ℝ-CNN) and its complex counterpart, complex-valued CNN (ℂ-CNN), pertinent to our experiments, are discussed in detail.

4.2 The building blocks of ℝ-CNNs and ℂ-CNNs

4.2.1 Convolution

In the field of Computer Vision, convolution operation is utilized for applying a 2D filter to an image for the purpose of photo enhancement or feature extraction. In photo enhancement, the filters used can have different effects such as sharpening, blurring, dilating, and more, whereas in feature extraction, different features such as edges, corners, gradients etc. can be extracted. In short, the application of filters through convolution operation enables us to convert images in a form which are easier to understand and can further help in myriads of computer vision applications such as face detection, object detection, SLAM etc.

In a CNN, the convolutional layer performs the role of feature extraction in a way that each succeeding layer represents a more complex concept that helps get closer to classification [23]. For examples, the first layers would learn low-level features (e.g. edges) form patterns and the latter layers would learn higher level features (e.g. shapes, textures) which would inturn be composed of these the earlier ones.

Units in a convolutional layer are organized in feature maps, within which each unit is connected to local patches in the feature maps of the previous layer through a set of weights called a filter bank [23]. The local patches in the feature maps of the previous layer are called the receptive field for the respective unit in the feature map under discussion. Different feature maps in a layer use different filter banks but each unit in the same feature map undergo the same filter bank. The new feature map can be obtained by first convolving the input with a learned kernel and then applying an element-wise nonlinear activation function on the convolved results [11]. The nonlinear acitvation function will be discussed in detail in Section 4.2.3.

Convolution in ℝ-CNNs

Convolution entails point-wise multiplication of the filter with the input image and summing all the multiplicands to form the unit in the feature map corresponding to that on which the filter's central unit is positioned in the output feature map (after the application of non-linear activation function). The filter slides and performs this operation all over the space of the input by centering upon all or some of the units of the input, according to the design choice of number of strides. The complete feature maps are obtained by using several different kernels.

Mathematically, the feature value at location (i, j) in the k-th feature map of l-th layer, $z^l_{i,j,k}$, is calculated by:

$$z^l_{i,j,k} = {w^l_k}^T * x^l_{i,j} + b^l_k \tag{4.1}$$

where w^l_k and b^l_k are the weight vector and bias term of the k-th filter of the l-th layer respectively, and $x^l_{i,j}$ is the input patch centered at location (i, j) of the l-th layer. Note that the kernel w^l_k that generates the feature map $z^l_{:,:,k}$ is shared. Such a weight sharing mechanism has several advantages such as it can reduce the model complexity and make the network easier to train [11].

Convolution in ℂ-CNNs

Like complex domain can be thought of as an extension of real domain, complex-valued convolution can also be formulated through such thinking. In complex-valued convolution, the filter

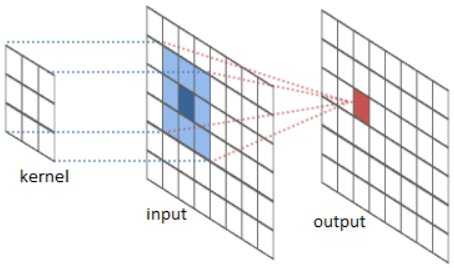

Figure 4.2: A schematic sketch of the convolution operation. An unit in the output is the sum of point-wise multiplication of the kernel and input patch [1]

and the input both take on the complex form:

$$\mathrm{W} = \mathrm{A} + i\mathrm{B} \tag{4.2}$$

$$\mathrm{h} = \mathrm{x} + i\mathrm{y} \tag{4.3}$$

where A and B are real-valued matrices which belong to one complex kernel, implemented as seperate layers. x and y are real-valued input vecetors representing the real and imaginary part of the complex-valued patch of the input layer. Note that the real and imaginary parts of both the kernel and the input are present as separate layers, and the result of the convolution is one complex-valued layer represented as two real-valued layers [12].

Taking advantage of the fact that convolution is distributive, the result of $\mathrm{W} * \mathrm{h}$ is given in equation 4.4. A schematic of the implementation of convolution in our experiments is depicted in 4.3.

$$\mathrm{W} * \mathrm{h} = (\mathrm{A} * \mathrm{x} - \mathrm{B} * \mathrm{y}) + i(\mathrm{B} * \mathrm{x} + \mathrm{A} * \mathrm{Y}) \tag{4.4}$$

4.2.2 Batch normalization

One of the most widespread methods used to improve the performance of Deep Neural networks, including CNNs, is called Batch Normalization (BN). Introduced by Ioffe and Szegedy in 2015 [20], BN helps speed up the training of the network and enables the use of a wider variety of acceptable hyper-parameters by reducing internal covariate shift. Internal Covariate Shift is defined as the change in the distribution of network activations, the input to the next layer, due to the change in network parameters during training [20]. Normalization of input data is widely accepted method to make the nerual networks converge faster, as stated in [25]. In the case of BN, this idea of input normalization is applied on the input of each layer which

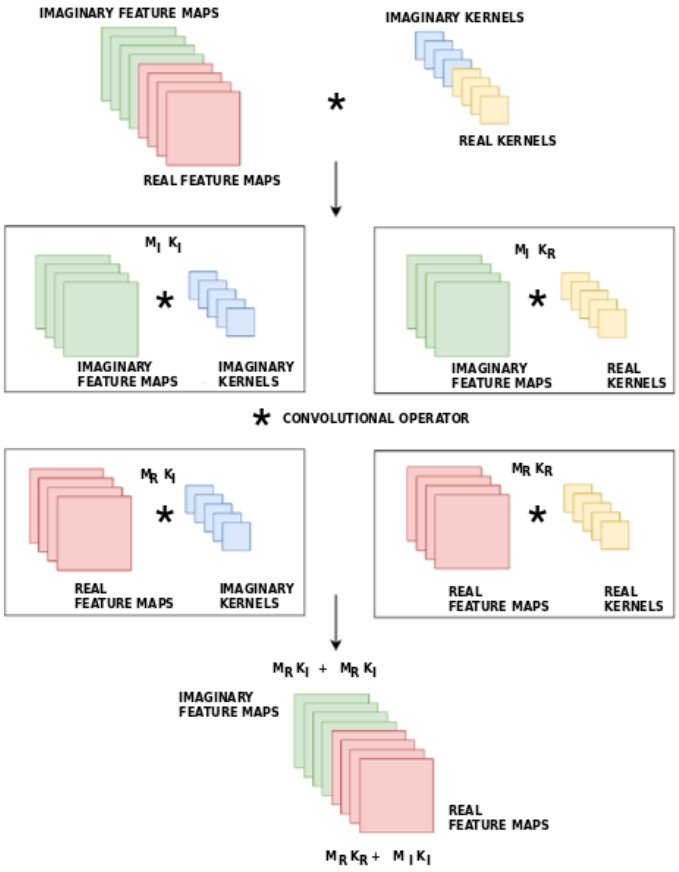

Figure 4.3: A schematic sketch of the complex-convolution operation. M and K represent the feature map and kernel, respectively. [31]

happens to be the output of the previous layer. It fights the internal covariate shift problem by a normalization step that fixes the means and variances of layer inputs where the estimations of mean and variance are computed after each mini-batch rather than the entire training set [11]. The exact procedure proposed is described later in 4.2.2.

Batch normalization has many advantages compared with global data normalization that is done before training. Apart from reducing internal covariant shift, BN also reduces the dependence of gradients on the scale of the parameters or of their initialization strategy. This benefits the flow of gradients and enables the use of higher learning rate without the risk of divergence [11]. BN also weakens the coupling between functions of each layers which in turn expedites convergence. Furthermore, as BN involves subtraction of mean and scaling by standard

deviation, where mean and standard deviation are calculated on mini-batches, multiplicative and additive noise is added to the each iteration. As a result, a slight regularization effect is induced in the network which helps it not to rely too much on the output of a single neuron/node in any layer of the network. One should note that the bigger the size of the mini-batch, the weaker is the regularization effect.

Real-valued batch normalization

In ℝ-CNNs, the BN step is generally performed between the convolution step and the activation step, although the original paper [20] proposed it for after the activation step. Suppose that a layer to normalize has a d dimensional input, i.e., $\mathrm{x} = [x_1, x_2, ..., x_d]^{\mathrm{T}}$. The first step is to normalize the k-th dimension as follows:

$$\hat{x}_k = \frac{x_k - \mu}{\sqrt{\sigma^2 - \epsilon}} \tag{4.5}$$

where μ and σ^2 are the mean and variance of mini-batch respectively, and ϵ is a constant value. Next, we let the network change the distribution of the network as is required by the model through the help of one multiplicative and one additive parameter. The normalized input $\hat{x}_k$ is further transformed into:

$$y_k = \mathrm{BN}_{\gamma,\beta}(x_k) = \gamma\hat{x}_k + \beta \tag{4.6}$$

where γ and β are learned parameters. It might not be beneficial for the layer input to have zero mean and variance, although the network has the choice to keep them such through the two parameters. γ and β help scale and shift the mean of the input distribution to better make advantage of the nonlinearity of the activation function.

Complex-valued batch normalization

The complex-valued BN is an extension of its real-valued counterpart and it was first introduced in [31]. The first step, of course, is normalization of the input distribution. Unlike real-valued BN, merely translating and scaling, like in [20], would result in skewed or elliptical variance with high eccentricity [31]. To cater for this problem, the authors the normalization step of complex BN is treated as whitening of 2D vectors. The following equations describe the process.

$$\tilde{x} = (V^{-\frac{1}{2}})(x - \mathbb{E}[x]) \tag{4.7}$$

$$V = \begin{pmatrix} V_{rr} & V_{ri} \\ V_{ir} & V_{ii} \end{pmatrix} = \begin{pmatrix} \text{Cov}(\Re\{x\}, \Re\{x\}) & \text{Cov}(\Re\{x\}, \Im\{x\}) \\ \text{Cov}(\Im\{x\}, \Re\{x\}) & \text{Cov}(\Im\{x\}, \Im\{x\}) \end{pmatrix} \tag{4.8}$$

As described by the equations above, the mean-centered input $(x - \mathbb{E}[x])$ is multiplied by the inverse of the square-root of input variance V in the normalization step.Whereas in real-valued BN, the first step entailed converting the input distribution into a standard normal distribution, the complex-valued BN requires it to be converted to standard complex normal distribution, which is characterized by having location parameter (also called mean) $\mu = 0$, covariance matrix $\Gamma = 1$, and the relation matrix (also called pseudo-covariance) $C = 0$. The formulae of these parameters are given by:

$$\begin{aligned} \mu &= \mathbb{E}[\tilde{x}] \\ \Gamma &= \mathbb{E}[(\tilde{x} - \mu)(\tilde{x} - \mu)^*] = V_{rr} + V_{ii} + i(V_{ir} - V_{ri}) \\ C &= \mathbb{E}[(\tilde{x} - \mu)(\tilde{x} - \mu)] = V_{rr} - V_{ii} + i(V_{ir} + V_{ri}) \end{aligned} \tag{4.9}$$

The next step in complex-valued BN is to scale and shift the input distrubtion to a desired mean and variance, with the help of γ and β, respectively. γ is now a 2x2 positive-semidefinite matrix with three tunable parameters and it is given by:

$$\gamma = \begin{pmatrix} \gamma_{rr} & \gamma_{ri} \\ \gamma_{ri} & \gamma_{ii} \end{pmatrix} \tag{4.10}$$

β is a complex parameter with real and imaginary learnable components. The final step of complex-valued BN follows the same equation as of real-valued BN, given by:

$$y_k = \text{BN}_{\gamma,\beta}(x_k) = \gamma \hat{x}_k + \beta \tag{4.11}$$

4.2.3 Activation function

In the absence of non-linear activation functions, the output of a neural network would essentially be a linear combination of the input, however deep the network is. Non-linear activation functions are introduced in order to add non-linearity to the network so that functions of a higher degree than 1 can be learned, thus giving them the title of universal function approximators. In a CNN, activation function is applied element-wise to the output of the convolution (or BN, when it is used) and they act as to decide whether or not this output should be taken into account by its connection, and if so, what should be the nature of it. For R-CNNs, some common types of activation functions include Sigmoid, Hyperbolic Tangent function (tanh), and the most popular, Recitified Linear Unit (ReLU).

Real-Valued Activation Functions

ReLU

Popularized by [22], Rectified Linear Units (ReLU) [26] remains the most commonly used activation function in deep learning/CNN architectures for the community. ReLU is given by:

$$\text{ReLU}(x) = \begin{cases} x, & \text{for } x \geq 0 \\ 0, & \text{for } x < 0 \end{cases} \tag{4.12}$$

Other variants of ReLU include Leaky ReLU (LReLU), Parametric ReLU (PReLU), Randomized ReLU (RReLU), and Exponential Linear Unit (ELU), as shown in Figure 4.2.3.

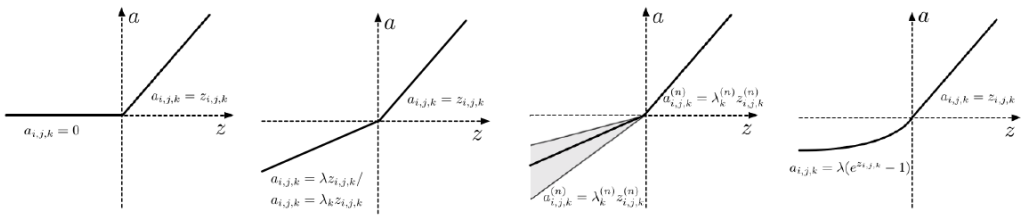

Figure 4.4: Variants of ReLU activation function

Complex-Valued Activation Functions

Real-valued activation functions in ℝ-CNNs have to strictly comply with the requirement of differentiability for the chain rule, the backbone of the Gradient Descent optimization algorithm, to work. Since ℂ-CNNs also make use of the same optimization algorithm, it is only natural to think that the complex activation functions must also be differentiable at any point in their domain i.e. they should be Holomorphic. However, Louiville theorem presents us with a realization: the only complex functions that can be Holomorphic are constant functions. This begs us to question if the condition of differentiability can be relaxed while still being able to optimize the network. We find a way to circumvent this problem by ignoring the requirement of complex differentiability, finding solace in the fact that as long as the complex function is differentiable with respect to its real part and its imaginary part, it should provide sufficiency in its ability to perform back-propagation, as demonstrated in [31], [13], [35]. The chain-rule for complex function differentiation is given in Section 3.2.4. However, we must note that since Holomorphic functions obey CR equations, they can be beneficial in granting computational efficiency to our process as we would only need to compute half the number of partial derivatives [29].

Complex-valued activation functions can be divided into two types: Split-complex type and Transcendental type. Split-complex type functions deals with real and imaginary parts (Real-Imaginary subtype) or magnitude and phase (Magnitude-phase subtype) independently but identically. Hence, they can be thought to work similar to real-valued networks with double the number of dimensions. The Split-complex activation functions are said to be appropriate when we can assume that our data contains cartesian symmetry when using Real-imaginary type, rotational symmtery when using Magnitude-phase type. Split-complex functions are Holomorphic but unbounded.

Transcendental type is the one which are bounded almost everywhere, with well-defined first order derivatives. They do have singularities but it is suggested by Kima and Adali (2003) [21] that with proper weight initialization and regularization schemes, this type of activation functions will not hamper the process of backpropagation.

ℂ**ReLU**

ℂReLU is Real-Imaginary type activation function introduced by Chiheb *et al.* (2018) [31], which satisfies the CR equations only in the first and the third quadrant. The formula is given as follows:

$$\mathbb{C}\text{ReLU}(z) = \text{ReLU}(\Re(z)) + i\text{ReLU}(\Im(z)) \tag{4.13}$$

z**ReLU**

Proposed by Guberman (2106) [12], zReLU does not strictly fall in any of the two categories described above. It follows the CR equations on all points on the complex domain except for the points along the real and imaginary axes of the complex plane. It is defined as follows:

$$z\text{ReLU}(z) = \begin{cases} z, & \text{for } \theta_z \in [0, \pi/2] + b \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.14}$$

tanh(z)

A Transcendental type, $\tanh(z)$ is a complex- hyperbolic tangent function suggested by Kim and Adali 2003 [21]. This function gives us a periodic singularity at every $(1/2+n)\pi i$, $n \in \mathbb{N}$, as shown in Fig. **??**. However, even in the presence of singularities, this activation function can be used to train the network with some success, although the possibility of hitting the singularity remains.

$$\tanh(z) = \tanh(x+iy) = = \begin{cases} \Re(\tanh(z)) = \frac{\sinh(2x)}{\cosh(2x)+\cosh(2y)} \\ \Im(\tanh(z)) = \frac{sinh(2x)}{\cosh(2x)+\cosh(2y)} \end{cases} \tag{4.15}$$

4.2.4 Weight initialization

Weight initialization can help in speeding up the convergence of the gradient descent algorithm, and combats the problem of vanishing and exploding gradients, especially in the absence of BN.

Real-valued weight initialization

In the context of ℝ-CNNs, two weight initialization techniques are popular among the community, which are explained below. In these two techniques, the first step concerns with choosing choosing a random distribution function whose defining parameters can be adjusted according to their benefits.

Glorot and Bengio (2010) criterion This type of initialization demands that the variance of the weight distibution should be as follows:

$$\text{Var}(W) = 2/(n_{in} + n_{out}) \tag{4.16}$$

The constraint ensures that the variances of the input, output and the gradients are the same. They go ahead to pick the weights from a Gaussian distribution with zero mean and variance as given in equation4.16. This initialization has been proved to be useful owing to the fact that it can detect the scale of initialization based on the number of inputs and outputs, keeping the signal in a reasonable limit through many layers [11] [9].

He *et al.* (2015b) criterion He *et al.* (2105b) proposed an initialization criterion specifically accounts for ReLU non-linearity function, which grants the ability to train extremely deep networks [11] [15].

$$\text{Var}(W) = 2/n_{in} \tag{4.17}$$

Complex-valued weight initialization

Chiheb *et al.* (2018) introduced two methods of complex weight initialization, which incorporate the two real-valued ℝ-CNN weight initialization techniques given above.

They start with dealing with the problem of defining the variance of complex weights. The standard definition of variance of complex variables goes as follows:

$$\text{Var}(W) = \mathbb{E}[WW^*] - (\mathbb{E}[W])^2 = \mathbb{E}[|W|^2] - (\mathbb{E}[W])^2 \tag{4.18}$$

When W is a symmetrically distributed around 0, equation 4.18 reduces to $\text{Var}(W) = \mathbb{E}[|W|^2]$. In order to calculate this variance, we adopt a route that involves Rayleigh distribution. We know that the variance of the magnitude of complex weights, $\text{Var}(|W|)$, follows a Rayleigh distribution, given below:

$$\text{Var}(|W|) = \mathbb{E}[|W||W|^*] - (\mathbb{E}[|W|])^2 = \mathbb{E}[|W|^2] - (\mathbb{E}[|W|])^2 \tag{4.19}$$

Bu substitution, we reach the following formulation for the variance of the weights:

$$\text{Var}(|W|) = \text{Var}(W) - (\text{E}[|W|])^2, and \text{Var}(W) = \text{Var}(|W|) + (\text{E}[|W|])^2 \tag{4.20}$$

The expectation and variance of magnitude of complex weights can be computer using the definitions given by the Rayleigh distribution, which can be characterized by just one parameter, mode (σ). By substitution, we can finally define the variance of the complex weights.

$$\text{E}(|W|) = \sigma\sqrt{\frac{\pi}{2}}, \text{Var}(|W|) = \frac{4-\pi}{2}\sigma^2 \tag{4.21}$$

$$\text{Var}(W) = 2\sigma^2 \tag{4.22}$$

Now that the variance is defined, we can choose either Glorot and Bengio criterion or the He. *2t al.* (2015b) criterion to equate to define the value of σ. In the former criterion, we get $\sigma = 1/\sqrt{n_{in} + n_{out}}$, and in the latter, $\sigma = 1/\sqrt{n_{in}}$. Having the appropriate σ, we can pick the magnitude of our weights from teh Rayleigh distribution. However, as the variance of W depends only on the magnitude of the weights and not their phase, we have the liberty of choosing phase. In their implementation, phase is chosen from a uniform distribution of -π to π.

Finally, we can use equation 4.23 to form the real and imaginary parts of the weights.

$$W = |W|e^{i\theta} = \Re\{W\} + i\Im\{W\} \tag{4.23}$$

4.2.5 Pooling layer

The sequence of layers in a typical CNN is as given below:

$$Convolutional\ Layer \longrightarrow BN \longrightarrow Activation\ Function \longrightarrow Pooling\ Layer \tag{4.24}$$

Pooling layer operates patch-wise on the input to extract a statistical summary of the patch that replaces the pixel corresponding to the central pixel of the patch in the output. This results in merging semantically similar features into one. One of the benefit lies in the reduced size of the inputs and weights for the next layers which reduces their computational burden. Another benefit is slight translational invariance of feature, since even if a feature occur at a slightly different areas of the feature map, the pooling layer would result in the same output as if it hadn't.

Pooling in ℝ-CNNs

Max pooling Max pooling is a patch-wise operation whose output is the maximum of all the values in the patch. Its formula is given by:

$$y = \max_{i,j \in patch} x_{i,j} \tag{4.25}$$

where x is a real number, and i, j are the indices of the patch.

Average pooling Average pooling is a patch-wise operation whose output is the maximum of all the values in the patch. Its formula is given by:

$$y = \left(\sum_{i,j \in patch} x_{i,j} \right) / sizeOfPatch \tag{4.26}$$

where x is a real number, and i, j are the indices of the patch.

Pooling in ℂ-CNNs

Max-by-mag pooling Maximum operator is not defined for a Complex Field since it is not an ordered field. Hence, we cannot compare two complex numbers wholly. Guberman (2016) [12] proposed the max-by-magnitude pooling, where the basis of comparison is the magnitude. In ℂ-CNNs, we are to decide what the statistical summary of the output should be based on. Knowing that choosing either the real part or the imaginary part would be wrong as it will mean that one is more important than the other, magnitude is chosen to provide the comparison. However, the output is itself a complex number courtesy of the $argmax$ operator, which allows the remainder of the network to remain in complex form. The formulation is given by:

$$y = \underset{i,j \in patch}{argmax} \ |z_{i,j}| \tag{4.27}$$

where z is a complex number, and i, j are the indices of the patch.

Average pooling Averaging operator is trivially generalized to the complex domain. As th

$$y = \left(\sum_{i,j \in patch} z_{i,j} \right) / sizeOfPatch \tag{4.28}$$

where z is a complex number, and i, j are the indices of the patch.

4.2.6 Fully-connected Layer

Fully-connected (FC) layer converts the CNN into a traditional neural network in that the 2D grid structure of the feature maps of the layers are vectorized and, unlike a CNN, each input is connected to each output through a weight. A single FC layer can exist at the end of the network, or a number of FC layers can exist until the output. If and when the FC layer becomes the output layer, it serves to provide the class-wise probability in a classification problem. If and when an FC layer exists as a hidden layer, an affine transformation is performed on the input: weight multiplication and bias addition, followed by application of the activation function.

FC layer in ℝ-CNNs

In ℝ-CNNs, the FC layer at the output usually employ the Softmax function which squashes the input between 0 and 1 so that it gives non-negative class-wise probability distribution over the classes. It is given by:

$$p_j^{(i)} = \frac{e^{z_j^i}}{\sum_{l=1\ldots K} e^{z_l^{(i)}}} \tag{4.29}$$

where $z_j^{(i)}$ is the activation of the FC layer.

FC layer in ℂ-CNNs

In ℂ-CNNs, the FC layer which act as hidden layers, same as in ℝ-CNNs, undergo an affine transformation (weight multiplication and bias addition) followed by an acitvation function. As complex numbers cannot be ordered, meaning that one cannot tell if a complex number is greater than the other, we are compelled to convert the complex numbers into real numbers. For this purpose, we find three ways of implementation in the literature:

1. Choosing to project the complex numbers to real domain by magnitude projection and feeding them to a real-valued FC layer, as proposed by Guberman (2016) [12].

2. Treating the real and imaginary components as two independent real-valued channels feeding into two output FC layers, as in Zhang *et al.* (2017) [35].

3. Ignoring the distinction between real and imaginary components by feeding them directly to a real-valued output FC layer, as in Chiheb *et al.* (2018) [31]

4.2.7 Loss function

Loss functions take on two roles in a convolutional neural network - regularization, and providing a measure of comparison between the prediction and the ground truth label. Regularization refers to the idea that the loss function can help enforce a specific quality of the learning in a neural network i.e. L2 regularization prefers that the result obtained is not a consequence of a little number of peaky weight vectors, but a large number of diffused weight vectors. In a classification problem solved with the help of neural networks, we intend to minimize the error, i.e. the difference between the prediction and the ground truth label, using an optimization technique, most common of which is Stochastic Gradient Descent (discussed in Section 4.2.8). Different loss functions will give different errors for the same prediction, and thus have a considerable effect on the performance of the model. One of the strict requirements is that loss functions are differentiable so that error gradients can be calculated.

Loss functions in ℝ-CNNs

In the classification problem, ℂ-CNNs employ a variety of loss functions including Mean Squared Error (MSE), Hinge Loss, Cross Entropy etc. One of the most common loss functions used for multi-class classification, Categorical Cross Entropy (negative), is defined as:

$$L_{y'}(y) = -\sum_{i=1} y_i^{'} \log(y_i) \tag{4.30}$$

where y_i is the predicted probability for class i and $y_i^{'}$ is the ground truth label of the class. Cross Entropy, true to the nature of a loss function, shows a trend of increase as the difference between the predicted and the ground truth class label increases.

Loss functions in ℂ-CNNs

The loss function is preceded by an FC layer which gives us probability scores of the output. Examples of loss functions used in ℂ-CNNs when the real and imaginary channels are compared with the respective real and imanginary components of the complex-valued label include Complex Quadratic Error Function, Complex FOurth Power Error Function, Complex Cauchy Error Function, Complex Log-Cosh Error Function, Adapted Least-Squares Error, among others, as found in [14] [35] [13]. In the other cases when the loss function compares two real

numbers (prediction and label), any ℝ-CNN based loss function can be used e.g. Categorical Cross Entropy.

4.2.8 Optimization through backpropagation

In the supervised learning problem involving CNNs, the output layer of the system produces a vector of scores, each denote the probability of each hypothesis class i, where $i \in [1, P]$. Ideally, the k^{th} entry of the vector, $O(k)$, should larger than the others, corresponding to and correctly identifying the input data class. However, the system needs to go through a training or optimization, in which the parameters or weights of the system are tuned such that they give us correct predictions of the result.

The most common procedure to train CNNs is known as Stochastic Gradient Descent(SGD). This learning algorithm involves computing a gradient vector, with the help of the chain-rule for derivatives, that, for each weight, indicates by what amount the error would increase or decrease if the weight were increased by a tiny amount [23]. SGD updates the parameters θ of the objective $L(\theta)$ as $\theta_{t+1} = \theta_t - \eta_t \nabla_\theta E[L(\theta_t)]$, where $E[L(\theta_t)$ is the average of $L(\theta)$ over a sub-set of the input datset called minibatch, and η is the learning rate. Hence, differentiability of the loss function and the activation functions is a strict condition in the case of ℝ-CNNs. However, as discussed in section 4.2.3, such strict requirement can be relaxed for the case of ℂ-CNNs, as substantiated by the literature.

Chapter 5

Methodology

This section details the methodology adopted to investigate the applicability of $\mathbb{C}$-CNNs to complex-natured data classification.

5.1 Approach

In our experimentation, we aim to answer the following question:

Can $\mathbb{C}$-CNNs *outperform* $\mathbb{R}$-CNNs *in a complex-valued data classification problem, given that phase of the complex-natured input data contains, as well, discriminatory information?*

The aims of our experimentation are:

- To establish the importance of various complex-valued activation functions in $\mathbb{C}$-CNNs
- To determine if $\mathbb{C}$-CNNs can better utilize phase information to produce better results than $\mathbb{R}$-CNNs

We employ two types of datasets: synthetically created toy data set which we call MNIST+P (Section 5.2.1), and real-world Radar datasets (Section 5.2.2), to feed into comparable $\mathbb{R}$-CNN and $\mathbb{C}$-CNN architectures in a bid to experimentally determine the winner in the complex-valued data classification problem. MNIST+P dataset is created to test if additional information of phase can help increase classification accuracy of $\mathbb{C}$-CNNs more than that of $\mathbb{R}$-CNNs, under the assumption that the complex-nature data would be better exploited by $\mathbb{C}$-CNNs. In this case, we are sure that phase does contain meaningful information which can be leveraged to increase accuracy. The real-world radar datasets are used for experimental verification of the

deductions made from the results of MNIST+P dataset experiments, and to test applicability of ℂ-CNNs an FMCW radar data.

The architecture of ℝ-CNN and ℂ-CNN employed in this task is discussed in Section 5.3. All the experiments are conducted using Keras [7] with Theano [30] backend.

5.2 Datasets

5.2.1 MNIST+P dataset

The MNIST database of handwritten digits (10 classes) has a training set of 60,000 examples, and a test set of 10,000 examples, where each image is of dimensions 28x28. It is a commonly used benchmark for supervised learning algorithm performance. In the MNIST+P dataset created for our experimentation, we consider that an image of MNIST dataset represents the magnitude of a complex number. The magnitude in each image is scaled such that the black part of the image corresponds to the value of 50, and the white part to 200. The phase for each image class (0,1,2,3,4,5,6,7,8,9) is picked from uniform distribution whose range is determined class-wise: The range for class 1 representing the digit 0 is $[0,\pi/10]$, that for class 2 representing the digit 1 is $[\pi/10,2\pi/10]$,...., that for class 10 representing digit 9 is $[9\pi/10,\pi]$.

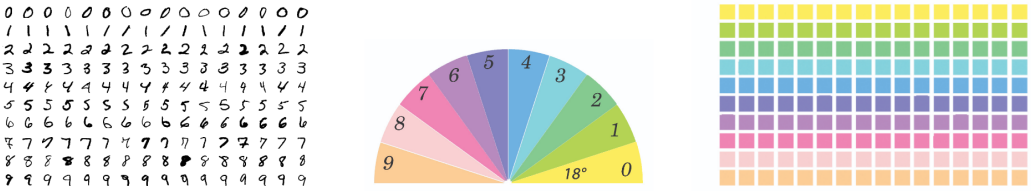

Figure 5.1: LEFT: Examples of digits from MNIST Dataset; CENTER: Angle bin sector for each class; RIGHT: Phase channels corresponding to the classes as shown in the left most image [2]

5.2.2 Radar datasets

Radar-150 and Radar-300 are two complex-valued datasets whose source is two FMCW radars with bandwidth of 6GHz and central frequencies of 150GHz and 300GHz, respectively. Both the datasets contain scans of six static objects (bike , cone, dog, mannequin, sign, trolley), performed in a control indoor environment by rotating the radar around a single object at different angles. Total number of images in both the datasets are 813. In each class, the scans are taken from two different distances away from the object i.e. 3.8m and 6.3m. Table ?? provides the data distribution among the classes. Figure BLABLABLA shows some examples

class	3.8m	6.3m	Total
bike	90	41	131
cone	25	25	50
dog	46	46	92
mannequin	90	90	180
sign	90	90	180
trolley	90	90	180

Table 5.1: Distribution of data over classes in Radar datasets

of images present in the datasets. Radar-300 is present in Cartesian representation and Polar representation while Radar-150 is present in only Polar distribution.

Preprocessing of Radar Dataset

We convert the Polar form information (magnitude and phase) to Cartesian form having a real channel and an imaginary channel. Now, the shape of each image becomes 32x32x2.

5.3 Architecture – $\mathbb{R}$-CNNs & $\mathbb{C}$-CNNs

The architecture employed for our experiments is a simplified variant of the one used by Chiheb *et al.* (2018) [31] for the problem of real-valued image classification. Our network contains three stages, each containing one residual block with 2 convolutional layers (3x3 filter size), 2 activation functions, and 2 BN layers in the order shown in Figure A.1. The skip connection of the first block differs from that of the second and the third block. The skip connection of the first block simply adds the input of the block to the output of the other thread of the block. However, the last two blocks perform a projection operation at their output: they convolve the input of the block with a 1x1 convolution operation with the same number of filters used throughout the stage, and concatenate it with the output of the other thread (with 2 conv, 2 BN, 2 activation layers) in a way that the distinction between real and imaginary parts remains. At the end of each stage, the number of filters are doubled. The architecture is illustrated in Figure 5.2.

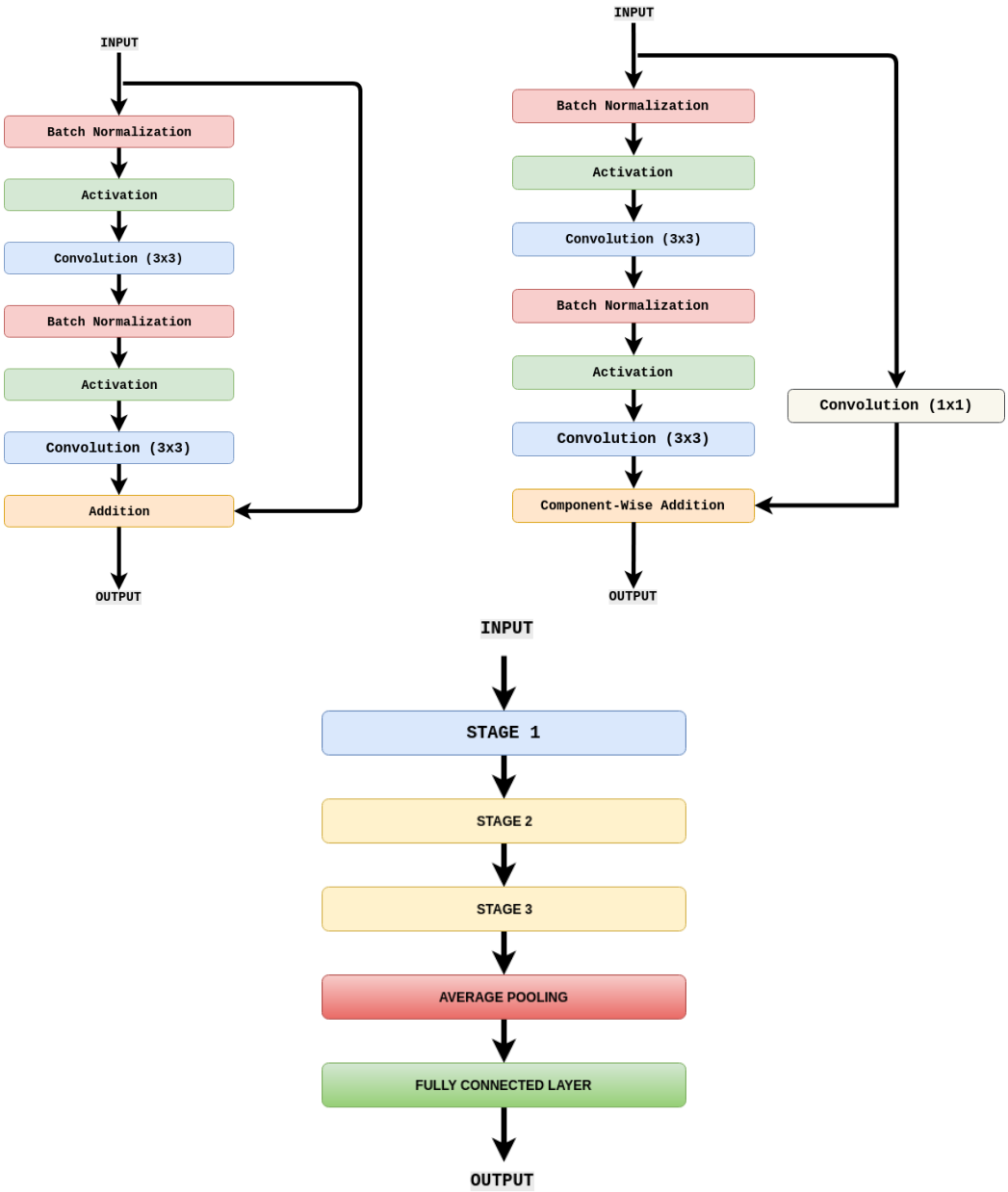

Figure 5.2: Architecture of the first residual block in stage 1 (left), and that in stages 2 and 3 (right); General architecture of both ℝ-CNN and ℂ-CNN (bottom)

Chapter 6

Experimentation

6.1 Practical implementation of complex numbers

Consider a complex number $z = a + ib$ with the real component a and the imaginary component b. In $\mathbb{R}$-CNNs, the filter bank or the feature maps of a layer exist in 3 dimensions, where two of them denote the size of the filter or feature map in 2D, and the third dimension indicates how many there are. If we have N feature maps (where N is divisible by 2), the first $N/2$ feature maps would be dedicated to the real components (a) and the last $N/2$ feature maps would be dedicated to the imaginary components (b). In this manner, the imaginary feature map corresponding to the real feature map on index one of the 3rd dimension would like on the $\frac{N}{2} + 1$ position.

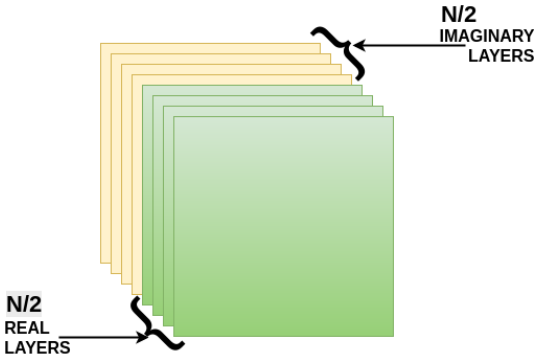

Figure 6.1: Arrangement of N feature maps in a $\mathbb{C}$-CNN layer

How to write about polar and cartesian form?

6.2 Experimentation details

For experimentation with MNIST+P dataset, we trained using Stochastic Gradient Descent for 64 batch size, 50 epochs, batch 64, and with learning rate 0.001. For the radar datasets, these hyperparameters were the same except for epochs which were 200. The number of starting filters in each stage are 12, 24, and 48, respectively.

Table 6.1: Test accuracy (%) of activation functions in $\mathbb{C}$-CNNs (zReLU(z), $\mathbb{C}$ReLU(z)), $tanh(z)$), $\mathbb{R}$-CNNs (ReLU) on MNIST+P dataset

-	zReLU(z)	$\mathbb{C}$ReLU(z)	tanh(z)	ReLU
Test Accuracy (%)	98.31	99.07	88.42	**99.59**

Peculiar problem with class with digit 1

Table 6.2: Test accuracy (%) of activation functions in $\mathbb{C}$-CNNs (zReLU(z), $\mathbb{C}$ReLU(z)), $tanh(z)$) and $\mathbb{R}$-CNNs (ReLU) on Cartesian and Polar representation of Radar-150 and Radar-300 datasets

-	zReLU(z)	$\mathbb{C}$ReLU(z)	tanh(z)	ReLU
Radar-300-Cart (%)	52.76*	79.09	NaN	96.63
Radar-150-Polar (%)	70.55	90.18	NaN	97.54
Radar-300-Polar (%)	85.88	89.36	NaN	92.87

* 300-cart-zrelu missclassifiying two classes(2,3). no convergence

6.3 Discussion

We DID experience convergence difficulties. Generalization what? Perform at par

CONVERGENCE PROBLEMS of tanhz, very data dependent! CReLU vs ReLU: inly dufference is th convolution - did it owrk? not really...

6.3.1 Generalizibility testing experiment

6.3.2 effect of complex weight initialization without batch normalization

6.3.3 Other architecture

CReLU and ReLU at least

data augmentation problem class problem.

Chapter 7

Conclusion & future work

Introduce LReLU! Tested out with different loss functions Proved that complex numbers can result in better representation of complex domain.

Future work

Appendix A

Appendix

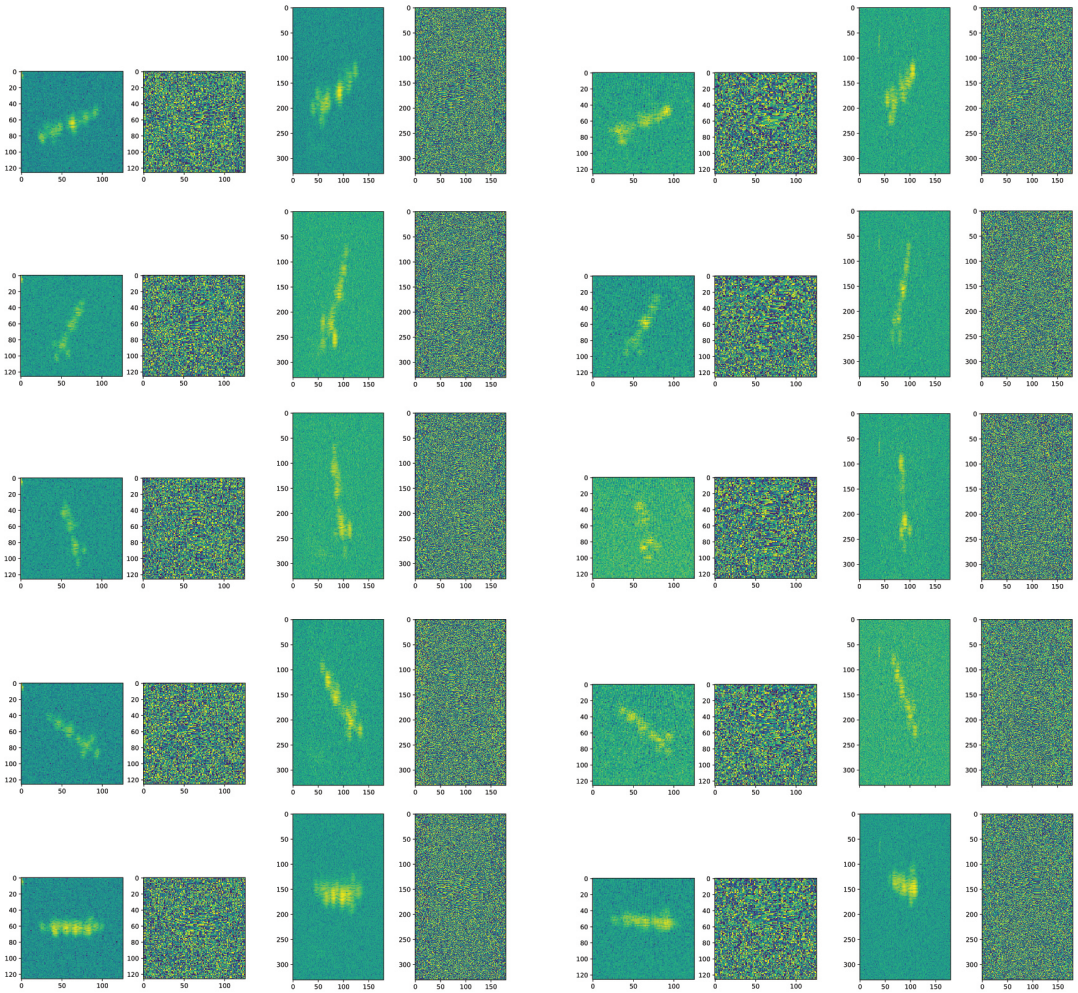

Figure A.1: Examples of images in MNIST Dataset [2]

Bibliography

[1] Bringing parallelism to the web with river trail.

[2] By josef steppan - own work, cc by-sa 4.0, https://commons.wikimedia.org/w/index.php?curid=64810040.

[3] Gotcha wide-angle synthetic aperture radar dataset.

[4] T. M. Apostol. *One-Variable Calculus, with an Introduction to Linear Algebra*, volume 1. Blaisdell Publishing Company, Waltham, MA, 1967.

[5] Martín Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. *CoRR*, abs/1511.06464, 2015.

[6] Joan Bruna, Soumith Chintala, Yann LeCun, Serkan Piantino, Arthur Szlam, and Mark Tygert. A theoretical argument for complex-valued convolutional networks. *CoRR abs/1503.03438*, 2015.

[7] François Chollet et al. Keras. 2015.

[8] Ivo Danihelka, Greg Wayne, Benigno Uria, Nal Kalchbrenner, and Alex Graves. Associative long short-term memory. *arXiv preprint arXiv:1602.03032*, 2016.

[9] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[10] Ian Goodfellow, Yoshua Bengio, and Courville Aaron. *Deep Learning*, pages 324–330. The MIT Press, 2016.

[11] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, and Gang Wang. Recent advances in convolutional neural networks. *CoRR*, abs/1512.07108, 2015.

[12] Nitzan Guberman. On complex valued convolutional neural networks. *CoRR*, abs/1602.09046, 2016.

[13] Ronny Haensch and Olaf Hellwich. Complex-valued convolutional neural networks for object detection in polsar data. In *Synthetic Aperture Radar (EUSAR), 2010 8th European Conference on*, pages 1–4. VDE, 2010.

[14] Ronny Hänsch and Olaf Hellwich. Classification of polarimetric sar data by complex valued neural networks. In *Proceedings of ISPRS Workshop, Hannover, Germany*, 2009.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[16] Akira Hirose. *Complex-valued neural networks*, volume 400. Springer Science & Business Media, 2012.

[17] Akira Hirose. Complex-valued neural networks. volume 400, page 45. Springer Science & Business Media, 2012.

[18] David H Hubel and Torsten N Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3):574–591, 1959.

[19] David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154, 1962.

[20] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[21] Taehwan Kim and Tülay Adalı. Approximation by fully complex multilayer perceptrons. *Neural computation*, 15(7):1641–1666, 2003.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[23] Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[24] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[25] Yann LeCun, Leon Bottou, Genevieve B. Orr, and Klaus Robert Müller. *Efficient BackProp in Neural Networks: Tricks of the Trade*, pages 9–50. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[26] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[27] Edouard Oyallon and Stéphane Mallat. Deep roto-translation scattering for object classification. In *CVPR*, volume 3, page 6, 2015.

[28] David P Reichert and Thomas Serre. Neuronal synchrony in complex-valued deep networks. *arXiv preprint arXiv:1312.6115*, 2013.

[29] Andy M Sarroff, Victor Shepardson, and Michael A Casey. Learning representations using complex-valued nets. *arXiv preprint arXiv:1511.06351*, 2015.

[30] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[31] Chiheb Trabelsi, Olexa Bilaniuk, Ying Zhang, Dmitriy Serdyuk, Sandeep Subramanian, Joao Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio, and Christopher J Pal. Deep complex networks. In *International Conference on Learning Representations*, 2018.

[32] Michael Wilmanski, Chris Kreucher, and Alfred Hero. Complex input convolutional neural networks for wide angle sar atr. In *Signal and Information Processing (GlobalSIP), 2016 IEEE Global Conference on*, pages 1037–1041. IEEE, 2016.

[33] Scott Wisdom, Thomas Powers, John Hershey, Jonathan Le Roux, and Les Atlas. Full-capacity unitary recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 4880–4888, 2016.

[34] Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. Harmonic networks: Deep translation and rotation equivariance. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2017.

[35] Zhimian Zhang, Haipeng Wang, Feng Xu, and Ya-Qiu Jin. Complex-valued convolutional neural network and its application in polarimetric sar image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(12):7177–7188, 2017.