Applicability of complex-valued convolutional neural networks to radar image 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

The abstract is very abstract as of now.

The only way out is through.

Robert Frost

Contents

Acknowledgments **vi**

1 Introduction **1**

1.1 Motivation 1

1.2 Problem definition 2

1.3 Document structure 2

2 State of the art **3**

3 Background **5**

3.1 Radar technology? 5

3.2 Complex domain theory 5

3.2.1 Complex numbers and their representations 5

3.2.2 Complex functions 6

3.2.3 Holomorphism & Complex Differentiability 6

3.2.4 Generalized chain rule for real-valued complex loss functions 6

4 Comparison of $\mathbb{R}$-CNNs with $\mathbb{C}$-CNNs **7**

4.1 Convolutional neural networks 7

4.2 The Building Blocks 8

4.2.1 Convolution 8

4.2.2 Batch normalization . . . 10
4.2.3 Activation function . . . 12
4.2.4 Weight initialization . . . 14
4.2.5 Pooling layer . . . 16
4.2.6 Loss function . . . 17
4.2.7 Optimization through backpropagation . . . 18

5 Methodology **20**
5.1 Representation of complex numbers . . . 20
5.1.1 MNIST+P Dataset . . . 20
5.1.2 Radar Dataset . . . 21
5.1.3 Architecture – $\mathbb{R}$-CNNs & $\mathbb{C}$-CNNs . . . 21

6 Experimental Results **22**
6.1 Results . . . 22

7 Conclusion **23**

A The first appendix **24**

Bibliography **27**

List of Figures

4.1 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] 9

A.1 A schematic sketch of the convolution operation. An item in the output is the sum of point-wise multiplication of the kernel and input patch [1] 24

List of Tables

Acknowledgments

– God.

– Parents, the big sibling and the little sibling.

– My supervisor, Andrew Wallace, for his support. Marcel Sheeny de Moraes for his guidance throughout the journey.

– VIBOTians: Wajahat, Albert, Dani, Thomas, Savinien, Anirudh, Darja, and the Vision Lab-mates without Clear Vision: Shubham, Alpha, Ziyang and Zaal.

– Sap for his generosity.

– CK, ARK, the bois.

– Myself, for getting it done.

Chapter 1

Introduction

1.1 Motivation

Motivation in LeCUN

Why CVNN?

Page 38 akira 2012 In CVNNs, the flexibility in learning and self-organization is restricted rather than that in double-dimensional real-valued neural net- works. As we discussed in Section 3.2.1, the restriction is brought by the four fundamental rules of arithmetic in complex numbers (especially multiplication as it entails rotation and scaling, not just scaling as in real valued case). Such fundamental rules in processing often work well in solving real world problems. This feature is one of the most useful advantages in CVNNs. wave-related phenomena such as sonic wave, lightwave, and electromagnetic wave.4

pg 19 In short, in CVNNs, we can reduce ineffective degree of freedom in learning or self-organization to achieve better generalization characteristics.

if we know a priori that the objective quantities include phase and/or amplitude, we can reduce possibly harmful portion of the freedom by employing a complex-valued neural network, resulting in a more meaningful generalization characteristics

physical meaning of radar magnitude and phase? discernibility lies in them?

the mannerism of how the learning unfolds is restricted

biological motivation signal processing associative memory data patchhhhh historical uses success with PolSAR

1.2 Problem definition

complex data with complex weights? leveraging the promise of complex valued networks

1.3 Document structure

Chapter 2

State of the art

Complex-valued neural networks have been widely investigated and have been found useful in the fields of adpative designing of patch antennas, neurophysiological analysis, and communications [13]. Increased incorporation of complex-valued units in recent works of Recurrent Neural Networks ([3], [27], [5]) and computer vision tasks([23], [4], [28]) has brought significant attention to the virtues of complex-valued representations.

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, the reason of which is described in the motivation.

Polarimetric SAR (PolSAR) uses microwaves with different polarisations to measure the distance to ground and the reflectance of a target [11]. 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) [11] 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 $\mathbb{R}$-NNs 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 ($\mathbb{C}$-NNs) and $\mathbb{R}$-CNNs [10]. 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.

Wilmanski *et al.* (2016) [26] explore the suitability of using $\mathbb{C}$-CNNs for Automatic Target

Recognition for complex-valued SAR data. Although their 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 ℝ-CNN network (87.30% accuracy). In the dataset they used (GOTCHA [2]), 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.

Zhang *et al*. (2017) [29] 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 ℝ-CNN, ℂ-CNN performs better on both the datasets while having approximately same number of parameters.

Chiheb *et al*. (2018) [25] compare the performance of different architectures of the ℂ-CNNs and ℝ-CNNs on the tasks of image recognition, music transcription, and speech spectrum prediction. ℂ-CNNs were reported to perform comparably to ℝ-CNNs for the first task, and achieve state-of-the-art performance on the two tasks while beating ℝ-CNNs. The authors also contribute the extension of Batch Normalization and Weight Initialization (BN) to complex domain.

Chapter 3

Background

3.1 Radar technology?

3.2 Complex domain theory

3.2.1 Complex numbers and their representations

Complex numbers are denoted as follows:

The field of complex numbers includes the field of real numbers as a subfield.

Complex numbers are useful abstract quantities that can be used in calculations and result in physically meaningful solutions.

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 of the complex plane, $|z|$, and the phase of the complex number, θ, measured positive counter-clockwise. The conversion formulae to and from Cartesian form and Polar form are given by:

$$x = |z|cos\theta \quad , \quad y = |z|sin\theta \tag{3.1}$$

$$|z| = \sqrt{x_2 + y_2} \quad , \quad \theta = \arctan(y/x) \tag{3.2}$$

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*.

ADD IMAGE

Cartesian form

Exponential form

3.2.2 Complex functions

3.2.3 Properties of complex number field

map c to c map c to r

3.2.4 Holomorphism & Complex Differentiability

Holomorphism guarantees that a complex-valued function is complex differentiable in the neighborhood of every point in its domain [25]. A complex function $f(z) : \mathbb{C} \mapsto \mathbb{C}$, $f(z) = u(z)+iv(z)$ ($u(z)$ 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:

$$\frac{\delta u}{\delta x} = \frac{\delta v}{\delta y} \quad , \quad \frac{\delta u}{\delta y} = -\frac{\delta v}{\delta x} \tag{3.3}$$

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

3.2.5 Generalized chain rule for real-valued complex loss functions

Consider that we have a real-valued loss 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:

$$\nabla_L(z) = \frac{\partial L}{\partial z} = \frac{\partial L}{\partial x} + i\frac{\partial L}{\partial y} = \Re(\nabla_L(z)) + i\Im(\nabla_L(z)) \tag{3.4}$$

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:

$$
\begin{aligned}
\nabla_L(t) &= \frac{\partial L}{\partial t} = \frac{\partial L}{\partial r} + i\frac{\partial L}{\partial s} \\
&= \frac{\partial L}{\partial x}\frac{\partial x}{\partial r} + \frac{\partial L}{\partial y}\frac{\partial y}{\partial r} + i\left(\frac{\partial L}{\partial x}\frac{\partial x}{\partial s} + \frac{\partial L}{\partial y}\frac{\partial y}{\partial s}\right) \\
&= \frac{\partial L}{\partial x}\left(\frac{\partial x}{\partial r} + i\frac{\partial x}{\partial s}\right) + \frac{\partial L}{\partial y}\left(\frac{\partial y}{\partial r} + i\frac{\partial y}{\partial s}\right) \\
&= \Re(\nabla_L(z))\left(\frac{\partial x}{\partial r} + i\frac{\partial x}{\partial s}\right) + \Im(\nabla_L(z))\left(\frac{\partial y}{\partial r} + i\frac{\partial y}{\partial s}\right)
\end{aligned}
\tag{3.5}
$$

Chapter 4

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

4.1 Convolutional neural networks

Convolutional Neural Networks (CNNs) (LeCun, 1989) [20] 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) [15] [16].

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, equivariant representations. In traditional neural netowrks, one learned parameter interacts with6 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 ouputs in the sense that very few parameters of the kernels 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 effeciency. Parameter sharing referes 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. [7]

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

4.2 The Building Blocks

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 e.g. sharpening, blurring, dilatting, and more. In case of features extraction, different features i.e. 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 [19]. 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 [19]. 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 [8]. The nonlinear acitvation function will be discussed in detail in Section 4.2.3.

NUMBER OF PARAMETERS/ MORE ACTIVATION FUNCTIONS IN POLSARHAENCSH

Real-Valued Convolution

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 [8].

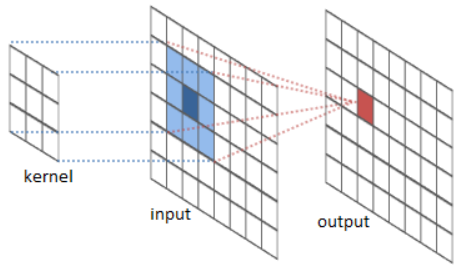

Figure 4.1: A schematic sketch of the convolution operation. An unit in the output is the sum of point-wise multiplication of the kernel and input patch [1]

Complex-valued convolution

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 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 [9].

Taking advantage of the fact that convolution is distributive, the result of $W * h$ results as follows:

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

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 [17], BN helps speed up the training of the network and enables the use of a wider variety of acceptable hyperparameters by reducing internal covariant 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 [17]. Normalization of input data is widely accepted method to make the nerual networks converge faster, as stated in [21]. In the case of BN, this idea of input normalization is applied on the input of each layer which 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 [8]. The exact procedure proposed is described lates 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 [8]. BN also weakens the coupling between functions of each layers which inturn expedites convergence. Furthermore, as BN involves subtraction of mean and scaling by standard deviation, where mena 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 $\mathbb{R}$-CNNs, the BN step is generally performed between the convolution step and the activation step, although the original paper [17] proposed it for after the activation step. Suppose that

a layer to normalize has a d dimensional input, i.e., 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 throught he help of one multiplicative and one additive paramter. 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 [25]. The first step, ofcourse, is normalization of the input distribution. Unlike real-valued BN, merely translating and scaling, like in [17], would result in skewed or elliptical variance with high eccentricity [25]. 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.

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

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

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 normalizatios 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 ditribution, 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:

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

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, thuse 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 [18], Rectified Linear Units (ReLU) [22] remains the most commonly used activation function in deep learning/CNN architectures for the community. ReLU is given by:

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

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

Soft-max

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 differentiablity, 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 provides sufficiency to be able to perform backpropagation, as demonstrated in [25], [10], [29]. 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 are beneficial in granting computational efficiency to our process because we would only need to compute half the number of partial derivatives [24].

Complex-valued activation functions can be divided into two types: Real-Imaginary type , and the Amplitude-Phase type [13]. The first type of activation functions deals with real and imaginary parts separately and independently, but identically. Hence, they can be thought to work similar to real-valued networks with double the number of dimensions. The latter type applies the nonlinearity on only the magnitude of the complex number, leaving the phase information unchanged [14] .

modReLU

Arjovsky *et al.* (2015) [3] proposed modReLU, an Amplitude-Phase type complex activation function, as a possible activation function for complex valued networks. It is formulated as:

$$\text{modReLU}(z) = \text{ReLU}(|z| + b)e^{i\theta_z} = \begin{cases} (|z| + b)\frac{z}{|z|}, & \text{for } |z| + b \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.13}$$

where b is a learnable parameter which dictates the offset for the deadzone around the origin 0. modReLU is an element-wise nonlinearity, where the element is muted(assigned a value of

zero in both real and imaginary channels) if it lies in the deadzone, or passed if it is beyond. Chiheb *et al.* [25] also make use of this non-linearity in their analysis of $\mathbb{R}$-CNNs and $\mathbb{C}$-CNNs.

$\mathbb{C}$ReLU

$\mathbb{C}$ReLU is Real-Imaginary type activation function introduced by Chiheb *et al.* (2018) [25], which satisfies the CR equations only in the first and the third quadrant. The formula is given as follows:

$$\mathbb{C}\text{ReLU}(z) = \text{ReLU}(\Re(z)) + i\text{ReLU}(\Im(z)) \tag{4.14}$$

zReLU

Proposed by Guberman (2106) [9], 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:

$$z\text{ReLU}(z) = \begin{cases} z, & \text{for } \theta_z \in [0, \pi/2] + b \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.15}$$

SPLIT COMPLEX SOFTMAX????

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 $\mathbb{R}$-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 [8] [6].

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 [8] [12].

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

Complex-valued weight initialization

Chiheb *et al.* (2018) introduced two methods of complex weight initialization, which incorporate the two real-valued $\mathbb{R}$-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 symmtrically 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. 2*t al.* (2015b) criteron to equate to define the vaue 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:

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

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 $\mathbb{R}$-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) [9] 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 Loss function

Loss functions take on two roles in a convolutional neural network - regularization, and provide 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 4.2.7). 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 differnetiable so that error gradients can be calculated.

Loss functions in $\mathbb{R}$-CNNs

In the classification problem, $\mathbb{C}$-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), defined as:

$$L_{y'}(y) = -\sum_{i=1} y_i^{'} \log(y_i) \tag{4.29}$$

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 $\mathbb{C}$-CNNs

In the literature, we find that the output of the final layer of $\mathbb{C}$-CNNs can be complex-valued or real-valued in nature. Since complex domain is not an ordered field, the output of the loss function has to real-valued, making it a real-valued complex function. When the output of the $\mathbb{C}$-CNN is complex-valued, the loss function maps from complex domain to real domain. Examples of such loss functions 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 [11] [29] [10]. In the other case when the loss function maps from complex domain to real domain, we have two options: either project the data from complex domain to real domain and use a real valued loss function [9], or use a real-valued complex loss function whose input is complex but the output is real (image recognition problem in [25]).

4.2.7 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 [19]. 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 $\mathbb{R}$-CNNs. However, as discussed in section 4.2.3, such strict requirement can be relaxed for the case of $\mathbb{C}$-CNNs, as substantiated by the literature.

Chapter 5

Methodology

WHY BRO WHY DID YOU CHOOSE AVERAGE POOLING? AHAAA! We don't wanna have to deal with a activation function whose values are only positive!

briefly mention the challenges of building a complex valued neural networks.

5.1 Representation 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.

IMAGE REQUIRED

5.1.1 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 wach image is of dimensions 28x28.

It is 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 is 50, and the white part is 200. The phase for each

image class is picked from uniform distribution whose range is determined class-wise: The range for class 1 representing the digit 0 is [0,π/10], that for class 2 representing the digit 1 is [π/10,2π/10],...., that for class 10 representing digit 9 is [9π/10,π].

Preprocessing of MNIST+P Dataset

Now that we know the magnitude and phase information for each image, we convert them to the cartesian form having a real channel and an imaginary channel. Now, the shape of each image becomes 28x28x2.

5.1.2 Radar Dataset

The magnitude and phase maps are first reshaped to a same size: 32x32.

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.

5.1.3 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) [25] for the problem of real-valued image classification. It contains three residual blocks containing 2 convolutional layers (3x3 filter size), 2 ReLU activation functions, and two BN layers in the order shown in Figure **??**. 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 that concatenates the output of the last residual block with the output of a 1x1 convolution applied on it with the same number of filters used throughout the stage and subsample by a factor of 2. While the $\mathbb{C}$-CNNs perform complex convolutions, complex BN, complex average-pooling, the $\mathbb{R}$-CNNs employs the real-valued counter parts. The weight initialization of $\mathbb{R}$-CNNs are done following the Glorot and Bengio criterion (2010) [6], while that of $\mathbb{C}$-CNNs follow the complex-valued counterpart of the same, as described in section 4.2.4. The last layer of both the network types contain a FC layer employing the Softmax function as the activation function, and the loss function is categorical-cross entropy.

Chapter 6

Experimental Results

6.1 Results

Chapter 7

Conclusion

Appendix A

The first appendix

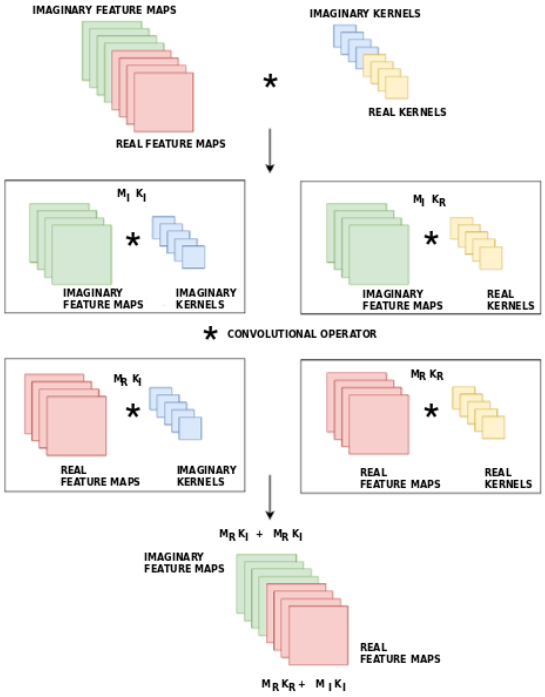

Figure A.1: A schematic sketch of the convolution operation. An item in the output is the sum of point-wise multiplication of the kernel and input patch [1]

Bibliography

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

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

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

[4] 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.

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

[6] 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.

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

[8] 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.

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

[10] 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.

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

[12] 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.

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

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

[15] 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.

[16] 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.

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

[18] 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.

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

[20] 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.

[21] 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.

[22] 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.

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

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

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.