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

List of Figures

List of Tables

Acknowledgments

- God.
- Parents, the big sibling and the little sibling.
- My supervisor for his support. Marcel Sheeny de Moraes for his guidance throughout the journey.
- VIBOTians, especially the Vision Lab-mates without Clear Vision: Shubham, Alpha, Ziyang and Zaal.
- CK, ARK, the bois.
- Myself, for getting it done.

Chapter 1

Introduction

1.1 Motivation

Motivation in LeCUN ubiquity of complex numbers around us

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

history

rnn

Guberman

SAR based

The incorporation of complex-valued blocks in the classification of Synthetic Aperture Radar (SAR) data has been very promising. The data is of complex by its very nature, hence opening up the interesting opportunity to employ $\mathbb{C}$-CNNs in various problems.

PolSAR uses microwaves with different polarisations to measure the distance to ground and the reflectance of a target [8]. One of the traditional techniques for classification involving PolSAR data is the use of Multilayer Perceptrons (MLP). Inspired by the success of MLPs in computer vision, Hänsch and Hellwich (2009) [8] employed the $\mathbb{C}$-CNNs 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 there architectures using different error functions for $\mathbb{C}$-CNNs, and compare $\mathbb{C}$-CNNs with their real-valued counterparts. The results conclude that $\mathbb{C}$-CNNs 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 further tackle the object-classification problem using complex-valued neural networks ($\mathbb{C}$-NNs) and $\mathbb{C}$-CNNs [7]. 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) [18] explore the suitability of using $\mathbb{C}$-CNNs for Automatic Target Recognition for complex-valued SAR data. Although their $\mathbb{C}$-CNN 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 [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) [19] 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.

Chiheb *et al*. (2018) [17] 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 and Weight Initialization (BN) to complex domain.

Chapter 3

Background

3.1 Radar Technology?

3.2 Complex Domain Theory

3.2.1 Complex Numbers

representation hello

3.2.2 Complex Functions

Representation types DO I DESCRIBE THEIR IMPLEMENTATION HERE?

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 [17]. 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:

$$\frac{\delta u}{\delta x} = \frac{\delta v}{\delta y} \quad , \quad \frac{\delta u}{\delta y} = -\frac{\delta v}{\delta x} \tag{3.1}$$

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

While CR equations are a necessary condition for holomorphism, the presence of the second condition makes CR equations become a sufficient condition for holomorphism.

Chapter 4

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

4.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) (LeCun, 1989) [14] 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) [9] [10].

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 with the 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. [4]

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 methodology, 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 [13]. 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 [13]. 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 [5]. 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 [5].

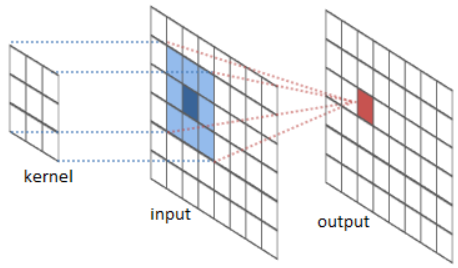

Figure 4.1: A schematic sketch of the convolution operation. An unit in the output is the sum of point-wise multiplication of the kernel and input patch [1]

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 [6].

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

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

This convolution strategy has been used in [6], [17] , Learning Rep, Polsar and ...

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 [11], 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 [11]. Normalization of input data is widely accepted method to make the nerual networks converge faster, as stated in [15]. 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 [5]. 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 [5]. 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 [11] 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 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 [17]. The first step, ofcourse, is normalization of the input distribution. Unlike real-valued BN, merely translating and scaling, like in [11], would result in skewed or elliptical variance with high eccentricity [17]. 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 [12], Rectified Linear Units (ReLU) [16] 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.

Complex-Valued Activation Functions

modReLU

Arjovsky *et al.* (2015) [3] proposed modReLU 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.* [17] also make use of this non-linearity in their analysis of ℝ-CNNs and ℂ-CNNs.

zReLU

ℂReLU

4.2.4 Weight Initialization

Real-Valued Activation Functions

Complex-Valued Activation Functions

4.2.5 Output Layer and Loss Function

Real-Valued CNNs

Complex-Valued CNNs

4.2.6 Optimization through Backpropagation

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!

5.1 Experimental setups

Explain the architecture, experimental details, the datasets used. Datasets: Toy+ results.Tables and shit. representation of complex numbers

5.1.1 MNIST+P Dataset

5.1.2 Radar Dataset

Chapter 6

Conclusion

Chapter 7

Experimental Results

Appendix A

The first appendix

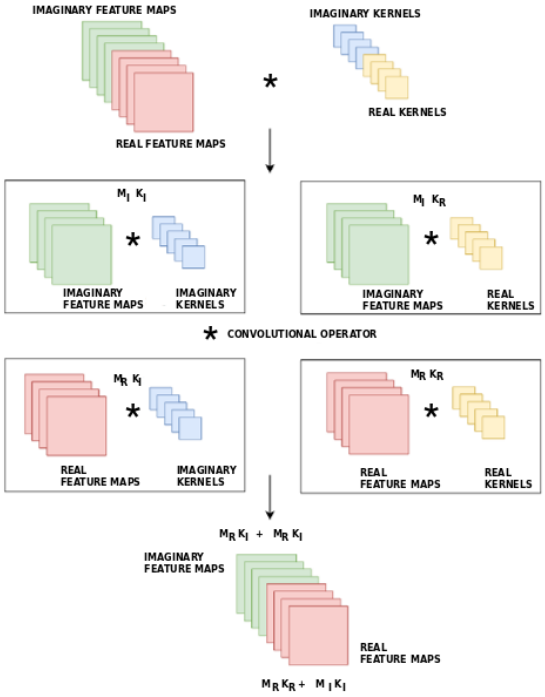

Figure A.1: A schematic sketch of the convolution operation. An item in the output is the sum of point-wise multiplication of the kernel and input patch [1]

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

[5] 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.

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

[7] 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.

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

[9] 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.

[10] 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.

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

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.