COMPLEX-VALUED VS. REAL-VALUED NEURAL NETWORKS FOR CLASSIFICATION PERSPECTIVES: AN EXAMPLE ON NON-CIRCULAR DATA

J. A. Barrachina*† *C. Ren** *C. Morisseau*† *G. Vieillard*† *J.-P. Ovarlez* *†

* SONDRA, CentraleSupélec, Université Paris-Saclay, 91192 Gif-sur-Yvette, France
† DEMR, ONERA, Université Paris-Saclay, 91120 Palaiseau, France

ABSTRACT

This paper shows the interest of Complex-Valued Neural Network (CVNN) on classification tasks for non-circular complex-valued datasets. Motivated by radar and SAR applications, we propose a statistical analysis of fully connected feed-forward neural networks performance in the cases where real and imaginary parts of the data are correlated through the non-circular property. In this context, comparisons between CVNNs and its real-valued equivalent models are conducted showing that CVNNs provide better performance for multiple types of non-circularity. Notably, CVNNs statistically perform less overfitting, higher accuracy and provide shorter confidence interval than its equivalent Real-Valued Neural Network (RVNN)s.

***Index Terms*—** Complex-Valued Neural Network, Real-Valued Neural Network, non-circularity.

1. INTRODUCTION

In the machine learning community, most neural networks are developed for processing real-valued features (voice signals, RGB images, videos, etc.). The signal processing community, however, has a higher interest in developing theory and techniques over complex fields. Indeed, complex-valued signals are encountered in a large variety of applications such as biomedical sciences, physics, communications and radar. All these fields use signal processing tools [1], which are usually based on complex filtering operations (Discrete Fourier Transform, Wavelet Transform, Wiener Filtering, Matched Filter, etc.). Thus, CVNNs appear as a natural choice to process and to learn from these complex-valued features since the operation performed at each layer of CVNNs can be interpreted as complex filtering. Notably, CVNNs are more adapted to extract phase information [2], which could be helpful, *e.g.*, for retrieving Doppler frequency in radar signals, classifying polarimetric Sythetic Aperture Radar (SAR) data [3, 4], etc. Furthermore, the hypothesis of circularity is not always satisfied as shown in [5, 6] for SAR images which depends on the region of interest. Therefore, non-circularity parameters are the key factors to improve performance in estimation and classification tasks as shown in [7, 8].

Interestingly enough, we propose to analyze the influence of the non-circular statistical property on the performance of both CVNN and RVNN networks. We show that particular structures of such complex data, as, for example, phase information or statistical correlation between real and imaginary parts, can notably benefit from using CVNN compared to its real-valued equivalent. Under this context, CVNN is potentially an attractive network to obtain better classification performance on complex datasets. Although CVNN has been investigated for particular structures of complex-valued data [2, 3, 9, 10], the difficulties in implementing CVNN models in practice have slowed down the field from growing further [11]. Then, we address this issue by providing a *Python* library to deal with the implementation of CVNN models. The code also offers statistical indicators, e.g., loss and accuracy box plots [12, 13], to compare the performance of CVNN models with its real-valued counterpart.

The paper is organized as following. Section 2 presents some mathematical background of CVNN and the circularity property for a complex-valued random variable. Section 3 discusses the fully connected feed-forward network architecture used for experiments. Section 4 illustrates the comparison of statistical performance obtained for each network. In particular, the sensitivity of CVNN and RVNN results are evaluated either by changing the dataset characteristics or the network hyper-parameters.

Although CVNN is an acronym that involves numerous complex-valued neural network architectures, in this work, we will always be referring to fully connected feed-forward ones. This convention was chosen to be coherent with the existing bibliography [2, 3, 9, 10, 14, 15].

2. MATHEMATICAL BACKGROUND

A natural way to build CVNN consists in extending RVNN for handling complex-valued neurons. The latter implies that the weights for connecting neurons and the hidden activation functions should be complex-valued. In contrast, the loss function remains real-valued to minimize an empirical risk during the learning process. Despite the architectural change for handling complex-valued inputs, the main challenge of CVNN is the way to train such neural networks.

2.1. Wirtinger derivation

When training CVNNs, the weights are updated using a complex gradient, so the back-propagation operation becomes complex-valued. In complex analysis, *Liouville's theorem* states that every bounded holomorphic function is constant [16, pp.70-71], implying that the loss and activation functions should be either constant or unbounded.

Wirtinger calculus [17] generalizes the notion of complex derivative for *non-holomorphic* functions. It states that given a complex function $f(z)$ of a complex variable $z = x + j\,y \in \mathbb{C}, (x, y) \in \mathbb{R}^2$, the partial derivatives with respect to z and $\overline{z}$ respectively are:

$$\frac{\partial f}{\partial z} \triangleq \frac{1}{2}\left(\frac{\partial f}{\partial x} - j\,\frac{\partial f}{\partial y}\right), \frac{\partial f}{\partial \overline{z}} \triangleq \frac{1}{2}\left(\frac{\partial f}{\partial x} + j\,\frac{\partial f}{\partial y}\right). \tag{1}$$

Wirtinger calculus enables one to work with *non-holomorphic* functions, providing an alternative method for computing the gradient. Following references [14] and [18], the complex gradient is then defined as:

The authors would like to thank the Délégation Générale de l'Armement (DGA) for funding.

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

Also, for any real-valued loss function $\mathcal{L} : \mathbb{C} \to \mathbb{R}$, the complex derivative of the composition of $\mathcal{L}$ with any complex function $g : \mathbb{C} \to \mathbb{C}$ with $g(z) = r(z) + j\,s(z)$ is given by the following so-called chain rule:

$$\frac{\partial \mathcal{L} \circ g}{\partial \bar{z}} = \frac{\partial \mathcal{L}}{\partial r}\frac{\partial r}{\partial \bar{z}} + \frac{\partial \mathcal{L}}{\partial s}\frac{\partial s}{\partial \bar{z}}\,. \tag{3}$$

Equations (2) and (3) enable the complex backpropagation algorithm for training CVNN [19].

2.2. Circularity

The importance of circularity for CVNNs has already been mentioned in [9, 15]. Let us denote the vector $\mathbf{u} \triangleq [X, Y]^T$ as the real random vector built by stacking the real and imaginary parts of a complex random variable $Z = X + j\,Y$. The probability density function (pdf) of Z can be identified with the pdf of $\mathbf{u}$. The *variance* of Z is defined by:

$$\sigma_Z^2 \triangleq \mathbb{E}\left[|Z - \mathbb{E}[Z]|^2\right] = \sigma_X^2 + \sigma_Y^2, \tag{4}$$

where σ_X^2 and σ_Y^2 are respectively the *variance* of X and Y. The latter does not bring information about the *covariance*:

$$\sigma_{XY} \triangleq \mathbb{E}\left[(X - \mathbb{E}[X])\;(Y - \mathbb{E}[Y])\right], \tag{5}$$

but this information can be retrieved thanks to the *pseudo-variance* [20, 21]:

$$\tau_Z \triangleq \mathbb{E}\left[(Z - \mathbb{E}[Z])^2\right] = \sigma_X^2 - \sigma_Y^2 + 2\,j\,\sigma_{XY}. \tag{6}$$

The circularity quotient ϱ_Z is then: $\varrho_Z = \tau_Z/\sigma_Z^2$. If Z has a vanishing *pseudo-variance*, $\tau_Z = 0$, or equivalently, $\varrho_Z = 0$, it is said to be second-order circular. The correlation coefficient is defined as

$$\rho = \frac{\sigma_{XY}}{\sigma_X \sigma_Y}. \tag{7}$$

Therefore, complex non-circular random datasets are generated and classified, with two non-exclusive possible sources of non-circularity: X and Y have unequal variances or X and Y are correlated.

3. PROPOSED NEURAL NETWORKS

3.1. Model Architecture

3.1.1. Activation function

One of the essential characteristics of CVNN is its activation functions, which should be non-linear and complex-valued. The extension to complex field offers many possibilities to design an activation function. Among them, two main types are proposed by extending real-valued activation functions [22]:

- Type-A: $g_A(f) = g_R\left(\mathrm{Re}(f)\right) + j\,g_I(\mathrm{Im}(f))$,
- Type-B: $g_B(f) = g_r(|f|)\,e^{j\,g_\phi(\phi(f))}$,

where $f : \mathbb{C} \to \mathbb{C}$ is a complex function and g_R, g_I, g_r, g_ϕ are all real-valued functions[1].

Normally, g_ϕ is left as a linear mapping [22, 9]. Under this condition, using ReLU activation function for σ_r has a limited interest since the latter makes g_B converge to a linear function, limiting Type-B ReLU usage. Nevertheless, ReLU has increased in popularity over the others as it has proved to learn several times faster than equivalents with saturating neurons [23]. Consequently, we will adopt, in Section 4, the Type-A ReLU, which is a non-linear function, as CVNN hidden layers activation functions.

The image domain of the output layer depends on the set of data labels. For classification tasks, real-valued integers or binary numbers are frequently used to label each class. Therefore the output layer's activation function should be real-valued. For this reason, we use *softmax* (normalized exponential) as the output activation function, which maps the magnitude of complex-valued input to $[0; 1]$, so the image domain is homogeneous to a probability.

3.1.2. Number of layers

Even though the tendency is to make the models as deep as possible for Convolutional Neural Networks (CNN), this is not the case for fully-connected feed-forward neural networks, also known as Multi-Layer Perceptron (MLP). For these models, one hidden layer (1HL) is usually sufficient for the vast majority of problems [24, 25]. Although some authors may argue that two hidden layers (2HL) may be better than one [26], all authors seem to agree that there is currently no theoretical reason to use a MLP with more than two hidden layers [27, p. 158]

References [27] and [28] recommend the neurons of the hidden layer to be between the size of the input layer and the output layer. Therefore, two models will be used as default in section 4, one with a single hidden layer of size 64 and one with two hidden layers of shape 100 and 40 for the first and second hidden layers respectively. In order to prevent the models from overfitting, dropout regularization technique [29] is used on 1HL and 2HL hidden layers. Both CVNN and RVNN are trained with a dropout factor of 0.5.

3.1.3. Equivalent RVNN

	Data A		Data B		Data C	
Class	1	2	1	2	1	2
ρ	0.5	-0.5	0	0	0.5	-0.5
σ_X^2	1	1	1	2	1	2
σ_Y^2	1	1	2	1	2	1
τ_Z	j	$-j$	-1	1	$j-1$	$1-j$

Table 1: Dataset characteristics

To define an equivalent RVNN, the strategy used in [9] is adopted, separating the input z into two real values (x, y) where $z = x + j\,y$, giving the network a double amount of inputs. The same is done for the number of neurons in each hidden layer. Although this strategy keeps the same amount of features in hidden layers, it provides a higher capacity for the RVNN with respect to the number of real-valued training parameters [11].

3.1.4. Loss function and optimizer

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

3.1.5. Weights initialization

For weights initialization, Glorot uniform (also known as Xavier uniform) [30] is used, and all biases start at zero as those are *Tensorflow*'s current (v2.1) default initialization methods for dense layers.

[1]Although not with the same notation, these two types of complex-valued activation functions are also discussed in Section 3.3 of [9]

Glorot initialization generates weight values according to the uniform distribution in [30, eq.16] where its boundaries depend on both input and output sizes of the initialized layer.

4. EXPERIMENTAL RESULTS

All the results are reproducible using the source code available at [31].

4.1. Dataset setup

As mentioned previously, to respect the equivalence between RVNN and CVNN, we use in the following input vectors of size 128 (resp. 256) for CVNNs (resp. RVNN). Each element of the feature vector is generated according to a non-circular Complex Normal distribution $\mathcal{CN}(0, \sigma_Z^2, \tau_Z)$. Two sources of non-circularity could occur in practice: $\sigma_X \neq \sigma_Y$ and/or $\rho \neq 0$, or equivalently $\tau_Z \neq 0$. Therefore, we propose to evaluate the classification performance of CVNN and RVNN for three types of datasets presented in Table 1.

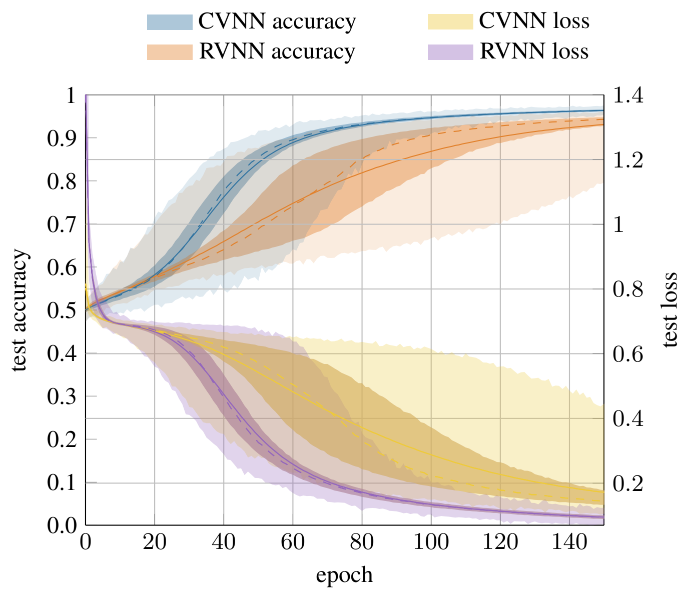

Fig. 1: Test loss and accuracy for 2HL CVNN and RVNN with a dropout of 50%. Solid line represents the mean. Dashed line the median.

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

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

4.2. 1HL and 2HL baseline results

To ensure that the models do not fall short of data, 10000 samples of each class were generated using 80% for the train set and the remaining 20% for testing. Accuracy and loss of both CVNN and RVNN, defined previously in Section 3.1, are statistically evaluated over 1000 Monte-Carlo trials. Each trial contained 150 epochs with a batch size of 100. This number was chosen observing that after 150 epochs, the accuracy and loss presented almost no amelioration.

Only the 2HL case is illustrated in Fig. 1 as their results are more favorable to the RVNN model. CVNN loss starts higher but decreases faster than RVNN. Both losses behave well without significant indication of overfitting. Additionally, the test accuracy of CVNN trials stays above 95%. The test accuracy of RVNN is lower than the CVNN one and presents more outliers.

Table 2 summarizes the test accuracy of 1HL and 2HL models for all three different datasets. The median error is computed as $1.57\,\mathrm{IQR}/\sqrt{n}$ [12], where IQR is the Inter-Quartile Range, and n is the number of trials. According to [13], using this error definition, if median values do not overlap there is a 95% confidence that their values differ.

Because the results are skewed, there is a big difference between mean and median accuracy, as the mean is less robust to outliers. For this reason, the median would be a better measure of central tendency in this paper's simulations. For the complex-valued model, the outliers tend to be the bad cases, whereas, for the real model, they are the good cases. This can be verified by the mean being lower than the median for CVNN and higher than the median for RVNN.

For dataset B with 2HL, both models fail to achieve excellent results on average. Despite these poor performances, CVNN still proves to be superior to RVNN by far. For 1HL, CVNN achieves very high accuracy with a median of over 84%. Dataset C presents almost the same results as dataset A with some improvement for both architectures.

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

In general, RVNN performed much better with 2HL than with 1HL. In this report we are trying to prove that CVNN outperforms RVNN regardless of the model architecture and hyper-parameters. To that end, even if all simulations to follow were done with both 1HL and 2HL, 2HL results will be prioritized, bearing in mind that 1HL cases were even more favorable to CVNN.

4.3. Phase and amplitude

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

Figure 2 shows the results for 2HL tested on dataset A with dropout. It can be seen that polar-RVNN highly improves compared to the conventional RVNN, showing higher mean accuracy but also much less variance, even lower than for CVNN. However, CVNN still outperforms both real models by a wide margin.

This higher performance of polar-RVNN against RVNN can be explained by the fact that dataset type A presents more relevant information in the phase. However, the opposite happens with dataset type B for which case, polar-RVNN completely fails to converge and achieves worst results than conventional RVNN for both 1HL and 2HL models, reason why results where omitted.

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

Table 2: Test accuracy results (%)

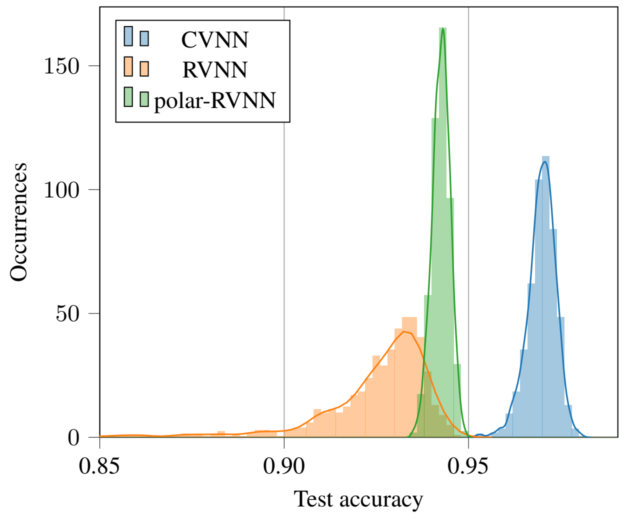

Fig. 2: Test accuracy histogram of 2HL CVNN, polar-RVNN and RVNN on dataset A

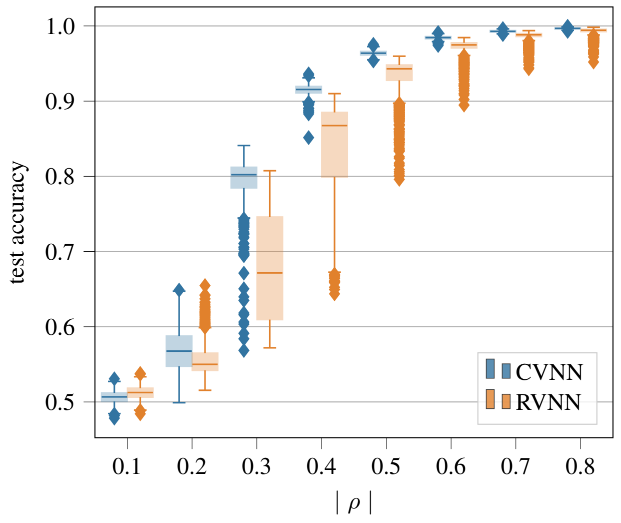

Fig. 3: Test accuracy box plot for different values of correlation coefficient ρ for 2HL model with dropout

4.4. Parameter sensibility study

In this section, a swipe through several model architectures and hyper-parameters is done to assert that the results obtained are independent of specific parameters. These simulations are done for both 1HL and 2HL networks.

Other sensibility studies were done but are not presented in this work due to paper size limitations. They concern learning rate, dataset size, feature vector size and multi-classes for all combinations of 1HL and 2HL with and without dropout.

4.4.1. Correlation coefficient

Several correlation coefficients have been tested for 1HL and 2HL models. Figure 3 shows the accuracy of 2HL models tested on datasets similar to data A (see table 1), in which the correlation coefficient varies from 0.1 to 0.8. As expected, for small ρ, both networks fail to distinguish between classes with accuracy values barely above 50%. Note that both models cannot achieve more than 85% for this case, as it was explained in section 4.1. As $|\rho|$ rises, CVNN merits become evident. When $|\rho|$ is close to one, then the link between real and imaginary parts is strengthened, which facilitates the classification of the data for both models. Results for 1HL are even more favorable for CVNN.

4.4.2. Hidden layer size

We evaluated the accuracy of 1HL for 4 sizes of the hidden layer. All these models were trained on dataset A. The median accuracy of CVNNs was always higher than the one of RVNN, no matter the number of hidden neurons. However, CVNN had low accuracy outliers for sizes 16 and 32, whereas RVNN did not. This could be explained by RVNN having higher capacity, as explained in section 3.1.3. Fortunately, this behavior disappears when the hidden size is well dimensioned.

5. CONCLUSION

In this paper, we provided a end-to-end tool for the implementation of CVNNs to fully use the complex-valued characteristics of the data. It also allows a more straightforward comparison with real equivalent networks in order to motivate further analysis and use of CVNN [31]. Moreover we showed that CVNNs stand as attractive networks to obtain higher performances than conventional RVNNs on complex-valued datasets. The latter point was illustrated by several examples of non-circular complex-valued data which cover a large amount of data types that can be encountered in signal processing and radar fields. All statistical indicators showed that CVNN clearly outperforms RVNN showing higher accuracy, smaller variance and less overfitting, regardless the model architecture and hyper-parameters. Conversely, the few cases where RVNN competes with CVNN, occurred when the dataset is small or the correlation coefficient $|\rho|$ is close to zero, rendering the discrimination from feature vectors nearly impossible. For these exceptions, neither CVNN and RVNN were actually of any practical use. Further analysis involving sensitivities to learning rates, feature vector size, multi-class dataset will be investigated to assert the generalisation of our results.

6. REFERENCES

[1] P. J. Schreier and L. L. Scharf, *Statistical Signal Processing of Complex-Valued Data*, Cambridge University Press., 2010.

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

[3] R. Hänsch and O. Hellwich, “Classification of polarimetric SAR data by complex valued neural networks,” in *Proc. ISPRS Hannover Workshop, High-Resolution Earth Imag. Geospatial Inf.*, 2009, vol. 37.

[4] Juanping Zhao, Mihai Datcu, Zenghui Zhang, Huilin Xiong, and Wenxian Yu, “Contrastive-regulated cnn in the complex domain: A method to learn physical scattering signatures from flexible polsar images,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 12, pp. 10116–10135, 2019.

[5] G. Vasile and F. C. Totir, “Circularity of complex stochastic models in polsar and multi-pass insar images,” in *2012 IEEE International Geoscience and Remote Sensing Symposium*, 2012, pp. 3720–3723.

[6] K. El-Darymli, C. Moloney, E. Gill, P. McGuire, and D. Power, “On circularity/noncircularity in single-channel synthetic aperture radar imagery,” in *2014 Oceans - St. John’s*, 2014, pp. 1–4.

[7] F. Barbaresco and P. Chevalier, “Noncircularity exploitation in signal processing overview and application to radar,” in *2008 IET Waveform Diversity Digital Radar Conference - Day 1: Waveform Diversity Design*, 2008, pp. 1–6.

[8] W. Wu, X. Li, H. Guo, L. Ferro-Famil, and L. Zhang, “Noncircularity parameters and their potential applications in uhr mmw sar data sets,” *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 10, pp. 1547–1551, 2016.

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

[10] A. Hirose, “Complex-valued neural networks: The merits and their origins,” in *2009 International joint conference on neural networks*. IEEE, 2009, pp. 1237–1244.

[11] N. Mönning and S. Manandhar, “Evaluation of complex-valued neural networks on real-valued classification tasks,” *arXiv preprint arXiv:1811.12351*, 2018.

[12] Robert McGill, John W Tukey, and Wayne A Larsen, “Variations of box plots,” *The American Statistician*, vol. 32, no. 1, pp. 12–16, 1978.

[13] J. M. Chambers, *Graphical methods for data analysis*, CRC Press, 2018.

[14] M. F. Amin, M. I. Amin, A. Y. H. Al-Nuaimi, and K. Murase, “Wirtinger calculus based gradient descent and Levenberg-Marquardt learning algorithms in complex-valued neural networks,” in *Neural Information Processing*, Berlin, Heidelberg, 2011, pp. 550–559, Springer Berlin Heidelberg.

[15] A. Hirose, *Complex-valued neural networks: Advances and applications*, vol. 18, John Wiley & Sons, 2013.

[16] B. Fine and G. Rosenberger, *The Fundamental Theorem of Algebra*, Springer Science & Business Media, 1997.

[17] W. Wirtinger, “Zur formalen theorie der funktionen von mehr komplexen veränderlichen,” *Mathematische Annalen*, vol. 97, no. 1, pp. 357–375, Dec 1927.

[18] A. Hirose, F. Amin, and K. Murase, “Learning algorithms in complex-valued neural networks using Wirtinger calculus,” in *Complex-Valued Neural Networks: Advances and Applications*, pp. 75–102. IEEE, 2013.

[19] H. Leung and S. Haykin, “The complex backpropagation algorithm,” *IEEE Transactions on Signal Processing*, vol. 39, no. 9, pp. 2101–2104, 1991.

[20] E. Ollila, “On the circularity of a complex random variable,” *IEEE Signal Processing Letters*, vol. 15, pp. 841–844, 2008.

[21] B. Picinbono, “Second-order complex random vectors and Normal distributions,” *IEEE Transactions on Signal Processing*, vol. 44, no. 10, pp. 2637–2640, Oct 1996.

[22] Y. Kuroe, M. Yoshid, and T. Mori, “On activation functions for complex-valued neural networks: existence of energy functions,” in *Artificial Neural Networks and Neural Information Processing, ICANN/ICONIP 2003*, pp. 985–992. Springer, 2003.

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

[24] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural Networks*, vol. 2, no. 5, pp. 359–366, Jan. 1989.

[25] M. Stinchcombe and H. White, “Universal approximation using feedforward networks with non-sigmoid hidden layer activation functions,” in *IJCNN International Joint Conference on Neural Networks*, 1989.

[26] A. J. Thomas, M. Petridis, S. D. Walters, S. M. Gheytassi, and R. E. Morgan, “Two Hidden Layers are Usually Better than One,” in *Engineering Applications of Neural Networks*, Cham, 2017, Communications in Computer and Information Science, pp. 279–290, Springer International Publishing.

[27] J. Heaton, *Introduction to neural networks with Java*, Heaton Research, Inc., 2008.

[28] P. Kulkarni and P. Joshi, *Artificial intelligence: building intelligent systems*, PHI Learning Pvt. Ltd., 2015.

[29] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *The journal of machine learning research*, vol. 15, pp. 1929–1958, 2014.

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

[31] J. A. Barrachina, “Complex-valued neural networks (CVNN),” `https://github.com/NEGU93/cvnn`, 2019.