COMPLEX- AND REAL-VALUED NEURAL NETWORK ARCHITECTURES

Anonymous authors
Paper under double-blind review

ABSTRACT

Complex-value neural networks are not a new concept, however, the use of real-values has often been favoured over complex-values due to difficulties in training and accuracy of results. Existing literature ignores the number of parameters used. We compared complex- and real-valued neural networks using five activation functions. We found that when real and complex neural networks are compared using simple classification tasks, complex neural networks perform equal to or slightly worse than real-value neural networks. However, when specialised architecture is used, complex-valued neural networks outperform real-valued neural networks. Therefore, complexvalued neural networks should be used when the input data is also complex or it can be meaningfully to the complex plane, or when the network architecture uses the structure defined by using complex numbers.

1 INTRODUCTION

In recent years complex numbers in neural networks are increasingly frequently used. Complex-Valued neural networks have been sucessfully applied to a variety of tasks specifically in signal processing where the input data has a natural interpretation in the complex domain.

In most publications complex-valued neural networks are compared to real-valued architectures. We need to ensure that these architectures are comparable in their ability to approximate functions. A common metric for their capacity are the number of real-valued parameters. The number of parameters of complex-valued neural networks are rarely studied aspects. While complex numbers increase the computational complexity, their introduction also assumes a certain structure between weights and input. Hence, it is not sufficient to increase the number of parameters.

Even more important than in real-valued networks is the choice of activation function for each layer. We test 5 functions: identity or no activation function, rectifier linear unit, hyperbolic tangent, magnitude, squared magnitude.

This paper explores the performance of complex-valued multi-layer perceptrons (MLP) with varying depth and width in consideration of the number of parameters and choice of activation function on benchmark classification tasks.

In section 2 we will give an overview of the past and current developments in the applications of complex-valued neural networks. We shortly present the multi-layer perceptron architecture in section 3 using complex numbers and review the building blocks of complex-valued network.

In section 4 we consider the multi-layer perceptron with respect to the number of real-valued parameters in both the complex and real case. We construct complex MLPs with the same number of units in each layer. We propose two methods to define comparable networks: A fixed number of real-valued neurons per layer or a fixed budget of real-valued parameters.

In the same section we also consider the structure that is assumed by introducing complex numbers into a neural network.

We present the activation function to be used in our experiments in section 5. In section 6 we present our experiments and their settings. Section 7 discuss the results of different multi-layer perceptrons on MNIST digit classification, CIFAR-10 image classification, CIFAR-100 image classification, Reuters topic classification and bAbI question answering. We identify a general direction of why and how to use complex-valued neural networks.

2 Related Literature

The idea of artificial neural networks with complex-valued input, complex-valued weights and complex-valued output was proposed in the 1970s (Aizenberg, 2016). A complex-valued back-propogation algorithm to train complex multi-layer networks was proposed in the 1990s by several authors (Benvenuto & Piazza, 1992; Nitta, 1993; Georgiou & Koutsougeras, 1992). In the 2000s complex neural networks, like real-valued neural networks, have been successfully applied to a variety of tasks. These tasks included the processing and analysis of complex-valued data or data with an intuitive mapping to complex numbers. Particularly, signals in their wave form were used as input data to complex-valued neural networks (Hirose, 2009).

Another natural application of complex numbers are complex convolutions (Bruna et al., 2015), since they have an application in both image and signal processing. While real convolutions are widely used in deep neural networks for image processing, complex convolution can replace real-valued convolutions (Trabelsi et al., 2017; Guberman, 2016; Popa, 2017; Haensch & Hellwich, 2010).

The properties of complex numbers and matrices introduce constraints into deep learning models. Introduced by Arjovsky et al. (2015) and developed further by Wisdom et al. (2016) recurrent networks, which constrain their weights to be unitary, reduce the impact of the vanishing or exploding gradient problem.

More recently complex numbers have been (re)discovered by a wider audience and used in approaches to other tasks like embedding learning (Trouillon et al., 2016; Sarroff et al., 2015), knowledge base completion (Trouillon et al., 2017) or memory networks (Kobayashi, 2017).

Despite their success in signal processing tasks, complex-valued neural networks have been less popular than their real-valued counter-parts. This may be due to training and reports of varying results in related tasks. The training process and architecture design are less intuitive, which stems from difficulties in differentiability of activation functions in the complex plane (Zimmermann et al., 2011; Hirose, 2004; Nitta, 2014).

An aspect that has received little attention is an appropriate comparison of real- and complex-valued neural networks. Many publications ignore the number of parameters all together (**?**), consider only the number of parameters of the entire model (**?**) or do not distinguish in complex- or real-valued parameters (**?**). While the latter is most confusing for the reader, all three problems lead to an inappropriate comparison of the overall performance.

There exists a significant body of work on exploring deep learning architectures for real-valued neural networks. Deep complex-valued neural networks are still to be explored. Previous work has also shown the significance of the activation, not only for the training and gradient computation, but also for the accuracy. Therefore, the width, depth and the choice of activation function need to be considered together.

We aim to fill this gap by systematically exploring the performance of multi-layered architectures on simple classification taks.

3 Complex-Valued Neural Network Architectures

Many fundamental building blocks of neural networks can be used in the complex domain by replacing real-valued parameters with complex parameters. However, there are some differences in training complex-valued neural networks. We introduce the building blocks and consider differences in structure and training. While the convolution on the complex plane using complex-valued filters is natural, it has been investigated in related literature (see section 2). In this work we focus on layers consisting of complex-valued neurons as building blocks and their use in multi-layer architecture. We define a complex-valued neuron analogous to its real-valued counter-part. In consequence we can use projection onto a complex weight matrix to realise complex-numbered embeddings.

The complex valued neuron can be defined as:

$$o = \phi(x \cdot w + b) \tag{1}$$

with the (real or complex) input $x \in \mathbb{C}^n$, complex weight $w \in \mathbb{C}^n$ and complex bias $b \in \mathbb{C}$. Arranging m neurons into a layer:

$$o = \phi(xW + b) \tag{2}$$

with the input $x \in \mathbb{C}^n, W \in \mathbb{C}^{n \times m}, b \in \mathbb{C}^m$. Similarly, we can define the projection onto a complex matrix if the input x is a projector (e.g. one-hot vector).

The activation function ϕ in all of the above definitions can be a real function $\phi : \mathbb{C} \to \mathbb{R}$ or complex function $\phi : \mathbb{C} \to \mathbb{C}$, but the function always acts on a complex variable. We will consider the choice of the non-linear activation function ϕ in more detail in section 5.

The loss function J should be a real function $J : \mathbb{C} \to \mathbb{R}$ or $J : \mathbb{R} \to \mathbb{R}$. Since there is no total ordering on the field of complex numbers, because the $i^2 = -1$, a complex-valued function may lead to added difficulties in training. To be able to interpret the output of the last layer as probability one, can use an additional activation function. Thus the activation of the output layer is $sigmoid(\phi(z))$ resp. $softmax(\phi(z))$ with $\phi : \mathbb{C} \to \mathbb{R}$ and is used as a real number in a classical loss function (e.g. cross entropy).

Both activation and loss functions are not always complex-differentiable. Hence, the training process in the complex domain differs. Similar to a real function, a complex function $f : \mathbb{C} \to \mathbb{C}$ at a point $z_0 \in \Omega \subset \mathbb{C}$ of an open subset Ω is complex-differentiable if there exists a limit such that

$$f'(z_0) = lim_{z|toz_0} \frac{f(z) - f(z_0)}{z - z_0} \tag{3}$$

A complex-valued function of one or more complex variables that is entire and complex differentiable is called holomorphic. While in the real-valued case the existence of a limit is sufficient for differentiability, the complex definition in equation 3 implies a stronger property. We map $\mathbb{C}$ to $\mathbb{R}^2$ to illustrate this point. A complex function $f(x + iy) = u(x, y) + iv(x, y)$ with real-differentiable functions $u(x, y)$ and $v(x, y)$ is complex-differentiable if they satisfy the Cauchy-Riemann equations:

$$\begin{aligned} \frac{\partial u}{\partial x} &= \frac{\partial v}{\partial y}, \\ -\frac{\partial u}{\partial y} &= \frac{\partial v}{\partial x} \end{aligned} \tag{4}$$

We simply separate a complex number $z \in \mathbb{C}$ into two real numbers $z = x + iy$. For f to be holomorphic, the limit not only needs to exist for the two functions $u(x, y)$ and $v(x, y)$, but they must also satisfy the Cauchy-Riemann equations. That also means that a function can be non-holomorphic (not complex-differentiable) in z, but still be analytic in its parts x, y. That is exactly if the two functions are real-differentiable, but do not satisfy the Cauchy-Riemann equations.

To be able to apply the chain rule, the basic principle of backpropagation, to non-holomorphic functions, we exploit the fact that many non-holymorphic functions, are still differentiable in their real and imaginary parts. We consider the complex function f to be a function of z and its complex conjugate $\bar{z}$. Effectively, we choose a different basis for our partial derivatives.

$$\begin{aligned} \frac{\partial}{\partial z} &= \frac{1}{2}\left(\frac{\partial}{\partial x} - i\frac{\partial}{\partial y}\right), \\ \frac{\partial}{\partial \bar{z}} &= \frac{1}{2}\left(\frac{\partial}{\partial x} + i\frac{\partial}{\partial y}\right) \end{aligned} \tag{5}$$

These derivatives are a consequence of the Wirtinger calculus (or CR-calculus). With the new basis we are able allow the application of the chain rule to non-holomorphic functions for multiple complex variables z_i:

$$\frac{\partial}{\partial z_i}(f \circ g) = \sum_{j=1}^{n} \left(\frac{\partial f}{\partial z_j} \circ g\right)\frac{\partial g_j}{\partial z_i} + \sum_{j=1}^{n} \left(\frac{\partial f}{\partial \bar{z}_j} \circ g\right)\frac{\partial \bar{g}_j}{\partial z_i}, \frac{\partial}{\partial \bar{z}_i}(f \circ g) = \sum_{j=1}^{n} \left(\frac{\partial f}{\partial z_j} \circ g\right)\frac{\partial g_j}{\partial \bar{z}_i} + \sum_{j=1}^{n} \left(\frac{\partial f}{\partial \bar{z}_j} \circ g\right)\frac{\partial \bar{g}_j}{\partial \bar{z}_i} \tag{6}$$

4 Real and Complex Parameters

In this section we discuss complex- and real-valued parameters in consideration of the number of parameters per layer and the structure assumed by complex numbers.

Any complex number $z = x + iy = r * e^{i\varphi}$ can be represented by two real numbers: the real part $Re(z) = x$ and the imaginary part $Im(z) = y$ or as length or magnitude $|z| = \sqrt{x^2 + y^2} = r$ and a phase φ. Effectively the number of real parameters of each layer is doubled: $p_{\mathbb{C}} = 2p_{\mathbb{R}}$.

The number of (real-valued) parameters is a metric of capacity or the ability to approximate functions. Too many and a neural network tends to overfit the data, too few and the neural network tends to underfit.

For a comparison of architectures the real-valued parameters per layer should be equal (or at least as close as possible) in the real architecture and its complex counter part. This ensures that models have the same capacity and a comparison shows the performance difference due to the added structure, not due to varying capacity.

Consider the number of parameters in a fully-connected layer in the real case and in the complex case. Let n be the input dimension and m the number of neurons.

$$p_{\mathbb{R}} = (n \times m) + m, p_{\mathbb{C}} = 2(n \times m) + 2m \tag{7}$$

For a multi-layer perceptron with k the number of hidden layers, and output dimension c the number of real-valued parameters without bias is given by:

$$p_{\mathbb{R}} = n \times m + k(m \times m) + m \times c, p_{\mathbb{C}} = 2(n \times m) + 2k(m \times m) + 2(m \times c), \tag{8}$$

At first glance designing comparable multi-layer neural network architectures, i.e. they have the same number of real-valued parameters in each layer, is trivial. Halving the number of neurons in every layer will not achieve a comparison, because the number of neurons define the output dimensions of the layer and the following layer's input dimension. We adressed this problem by choosing MLP architectures with an even number of hidden layers k and choose the number of neurons per layer to be alternating between m and $\frac{m}{2}$. Thus we receive the same number of real parameters in each layer compared to a real-valued network. As an example, let us consider the dimensions of outputs and weights in $k = 4$. For the real-valued case:

$$\begin{aligned}
&(1 \times n)\overbrace{(n \times m_1)}^{\text{Input layer}} \rightarrow (1 \times m_1)\overbrace{(m_1 \times m_2)}^{\text{Hidden layer}} \\
\rightarrow (1 \times m_2)&\overbrace{(m_2 \times m_3)}^{\text{Hidden layer}} \rightarrow (1 \times m_3)\overbrace{(m_3 \times m_4)}^{\text{Hidden layer}} \\
\rightarrow (1 \times m_4)&\overbrace{(m_4 \times m_5)}^{\text{Hidden layer}} \rightarrow (1 \times m_5)\overbrace{(m_5 \times c)}^{\text{Output layer}} \\
&\rightarrow \overbrace{(1 \times c)}^{Modeloutput}
\end{aligned} \tag{9}$$

where m_i is the number of (complex or real) neurons of the i-th layer. The equivalent with m complex-valued neurons would be:

$$\begin{aligned}
(1 \times n)(n \times \frac{m_1}{2}) &\rightarrow (1 \times \frac{m_1}{2})(\frac{m_1}{2} \times m_2) \\
\rightarrow (1 \times m_2)(m_2 \times \frac{m_3}{2}) &\rightarrow (1 \times \frac{m_3}{2})(\frac{m_3}{2} \times m_4) \\
\rightarrow (1 \times m_4)(m_4 \times \frac{m_5}{2}) &\rightarrow (1 \times \frac{m_5}{2})(\frac{m_5}{2} \times c) \\
&\rightarrow (1 \times c)
\end{aligned} \tag{10}$$

Another approach to the design of comparable architectures is to work with a parameter budget. Given a fixed budget of real parameters $p_{\mathbb{R}}$ we can define real or complex multi-layer perceptron

with an even number k of hidden layers such that the network's parameters are within that budget. All $k + 2$ layers have the same number of real-valued neurons $m_{\mathbb{R}}$ or complex-valued neurons $m_{\mathbb{C}}$.

$$m_{\mathbb{R}} = \begin{cases} \frac{p_{\mathbb{R}}}{n+c}, & \text{if } k = 0 \\ -\frac{n+c}{2k} + \sqrt{(\frac{n+c}{2k})^2 + \frac{p_{\mathbb{R}}}{k}}, & \text{otherwise} \end{cases} \tag{11}$$

$$m_{\mathbb{C}} = \begin{cases} \frac{p_{\mathbb{R}}}{2(n+c)}, & \text{if } k = 0 \\ -\frac{n+c}{2k} + \sqrt{(\frac{n+c}{2k})^2 + \frac{p_{\mathbb{R}}}{2k}}, & \text{otherwise} \end{cases} \tag{12}$$

Despite the straight forward use and representation in neural networks, complex numbers define an additional structure compared to real-valued networks. This interaction of the two parts becomes particularly apparant if we consider operations on complex numbers to be composed of the real and imaginary part or magnitude and phase:

$$\begin{aligned} z_1 z_2 &= (a + ib)(c + id) = (ac - bd) + i(ad + bc), \\ z_1 + z_2 &= (a + ib) + (c + id) = (a + c) + i(b + d) \end{aligned} \tag{13}$$

with complex numbers $z_1 = a + ib, z_2 = c + id$. In an equivalent representation with Euler's constant $e^{i\varphi} = cos(\varphi) + isin(\varphi)$ the real parts do not interact.

$$\begin{aligned} z_1 z_2 = (r_1 e^{i\varphi_1})(r_2 e^{i\varphi_2}) = (r_1 r_2 e^{i\varphi_1 + \varphi_2}), \\ z_1 + z_2 = (r_1 e^{i\varphi_1}) + (r_2 e^{i\varphi_2}) \\ = r_1 cos(\varphi_1) + r_2 cos(\varphi_2) + i(r_2 sin(\varphi_2) + r_1 sin(\varphi_1)) \end{aligned} \tag{14}$$

Complex parameters increase the computational complexity of a neural network, since more operations are required. Instead of a single real-valued multiplication operation, up to four real multplication and two real additions are required. Depending on the implementation and the representation, this can be significantly reduced.

Nevertheless, it is not sufficient to double the numbers of real parameters per layer to achieve the same effect as in complex-valued neural networks. This is also illustrated expressing a complex number $z = a + ib \in \mathbb{C}$ as 2×2 matrices M in the ring of $M_2(\mathbb{R})$:

$$M_{a,b} = \begin{bmatrix} x & -y \\ y & x \end{bmatrix} \text{ with } M_i = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, M_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{15}$$

so

$$M_{a,b} M_{c,d} = \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \begin{bmatrix} c & -d \\ d & c \end{bmatrix} = \begin{bmatrix} ac - bd & bc + dc \\ bc + dc & ac - bd \end{bmatrix} \tag{16}$$

An augmented representation of an input x allows the represention of complex matrix multiplication with an weight matrix W as larger real matrix multiplication:

$$Wx = \begin{bmatrix} Re(W) & -Im(W) \\ Im(W) & Re(W) \end{bmatrix} \begin{bmatrix} Re(x) \\ Im(x) \end{bmatrix} = \begin{bmatrix} Re(W)Re(x) - Im(W)Im(x) \\ Im(W)Re(x) + Im(x)Re(W) \end{bmatrix} \tag{17}$$

This added structure, however, also means that architecture design needs to be reconsidered. A deep learning architecture which performs well with real-valued parameters, may not work for complex-valued parameters and vice versa. In later sections we experimentally investigate what consequences this particular structure has for the overall performance of a model.

5 Activation Functions

In any neural network, real or complex, an important decision is the choice of non-linearity. With the same number of parameters in each layer, we are able to study the effects activation functions have

on the overall performance. The Liouville Theorem states that any bounded holomorphic function $f : \mathbb{C}to\mathbb{C}$ (that is differentiable on the entire complex plane) must be constant.

Hence, we need to choose unbounded and/or non-holomorphic activation functions. We chose the identity function to investigate the performance of complex models assuming a function which is linearly separable in the complex parameters by not introducing a non-linearity into the model. The hyperbolic tangents is a well-studied function and defined for both complex and real numbers. The rectifier linear unit is a well-studied function and illustrates the separate application of an activation function. The magnitude and squared magnitude functions are chosen to map complex numbers to real numbers.

- Identity (or no activation function):
$$\phi(z) = z \tag{18}$$
- Hyperbolic tangent:
$$\phi(z) = tanh(z) = \frac{sinh(z)}{cosh(z)} = \frac{e^z - e^{-z}}{e^z + e^{-z}} = \frac{e^{2z} - 1}{e^{2z} + 1} \tag{19}$$
- Rectifier linear unit (relU):
$$\phi(z) = relU(z) = relU(Re(z)) + relU(Im(z))j = max(0, Re(z)) + max(0, Im(z))j \tag{20}$$
- Magnitude squared:
$$\phi(z) = |z|^2 = x^2 + y^2 \tag{21}$$
- Magnitude (or complex absolute):
$$\phi(z) = |z| = \sqrt{x^2 + y^2} \tag{22}$$

In the last layer we apply an activation function $\phi : \mathbb{C} \to \mathbb{R}$ before using the softmax or sigmoid to use in a receive a real loss. Note that the squared magnitude allows a more efficient implementation than the magnitude. Thus we change the activation function of the last layer to:

$$sigmoid(|z|^2) = \frac{1}{1 + e^{-x^2-y^2}} \tag{23}$$

For an output vector $z = [z_0, z_1, \ldots, z_c]$

$$softmax(|z_j|^2) = \frac{e^{x^2+y^2}}{\sum_{i=1}^{c} e^{x^2+y^2}} \tag{24}$$

Applying the two functions in the opposite order as in $|sigmoid(z)|^2$ and $|softmax(z)|^2$ does not return probabilities from the last layer of a network and would take away the direct interpretability of the models output.

6 Experiments

To compare real and complex-valued neural networks and their architecture we chose various classification tasks and defined experiments. The settings are as follows:

- Experiment 1: We tested multi-layer perceptrons (MLP) with with $k = 0, 2, 4, 8$ hidden layers, fixed width of units in each layer in real-valued architectures and alternating 64, 32 units in complex-valued architectures (see section 4), no fixed budget, applied to Reuters topic classification , MNIST digit classification, CIFAR-10 Image classification, CIFAR-100 image classification. Reuters topic classification and MNIST digit classification use 64 units per layer, CIFAR-10 and CIFAR-100 use 128 units per layer. All tested activation functions are introduced in 5.

- Experiment 2: We tested multi-layer perceptrons (MLP) with fixed budget of 500,000 real-valued parameters, no fixed width, applied to MNIST digit classification, CIFAR-10 Image classification, CIFAR-100 image classification and Reuters topic classification. All tested activation functions introduced are in section 5. Used $sigmoid(|z|^2)$ function for the gates.

- Experiment 3: We tested the Memory Network architecture introduced by Weston et al. (2014) as complex-valued network in two versions - one below and one above parameter budget of the real-valued network. We used the bAbI question answering tasks with one, two and three supporting facts. Activation functions in each layer were defined by the original publication. The network used a recurrent layer, which defined by replacing the real-valued weight matrices with complex weight matrices.

For all of our experiments we used the weight initialisation discussed by (Trabelsi et al., 2017). However, to reduce the impact of the initialisation we ran each model at least 10 times. The larger memory networks were initialised 30 times. All models were trained over 100 epochs with an Adam optimisation. We used categorical or binary cross entropy as a loss function for all of our experiments and models. We used $sigmoid(|z|^2)$ or $softmax(|z|^2)$ as activation function for the last layer of the complex models.

7 Results and Discussion

Tables 1, 2, 3, 4 show the results for experiment 1. Generally, the performance of complex and real neural network in this setting is similar, altough the complex valued neural network tends to perform slightly worse. We found that the best choice for the activation function for the complex neural network is *relu* applied separatly to the imaginary and real parts. Suprisingly the hyperbolic tangents *tanh* and squared magnitude $|z|^2$ perform significantly worse

Tables 5, 6, 7, 8 show the results for experiment 2. Similar to experiment 1 the results show that the best choice for the activation function for the complex neural network is *relu* applied separatly to the imaginary and real parts. In both experiments with the depth of the architecture the performance of the complex neural network decreases significantly. These experiments illustrate that an increased width per layers outperforms an increased depth in classification tasks. This is true for the real and the complex case.

Table 9 shows the results for experiment 3. For a single supporting fact in the bABi data set real-valued neural network. In the first bABi task the real-valued version outperforms the two complex version of the memory network. In the more diffcult tasks with two or three supporting facts both, the small and large version, of the complex-valued neural network outperform the real-valued version - despite the reduce number of parameters.

We made the observation that the assumed structure with introducing complex numbers into neural networks has a regularising effect on the training procedure if used in combination with real-valued input. We also found that complex-valued neural networks are more sensitive towards their initialisation than real-valued neural networks.

Overall the complex-valued neural networks do not perform as well as expected. This may be due to the nature of the chosen tasks or the simple architecture of a multi-layer perceptron. Complex neural networks should be used if the data is naturally in the complex domain or can be mapped to complex numbers. The architecture should be selected to respect the expected structure complex numbers introduce to the network.

In conclusion the architecture needs to reflect the interaction of the real and imaginary part. If the structure is ignored, the model will not perform as well as the corresponding real-valued network.

Acknowledgments

Acknowledgements are hidden due to double blind review.

REFERENCES

Igor Aizenberg. *Complex-Valued Neural Networks with Multi-Valued Neurons*. Springer Publishing Company, Incorporated, 2016. ISBN 3662506319, 9783662506318.

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

N. Benvenuto and F. Piazza. On the complex backpropagation algorithm. *IEEE Transactions on Signal Processing*, 40(4):967–969, Apr 1992. ISSN 1053-587X. doi: 10.1109/78.127967.

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.

G. M. Georgiou and C. Koutsougeras. Complex domain backpropagation. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 39(5):330–334, May 1992. ISSN 1057-7130. doi: 10.1109/82.142037.

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

R. Haensch and O. Hellwich. Complex-valued convolutional neural networks for object detection in polsar data. In *8th European Conference on Synthetic Aperture Radar*, pp. 1–4, June 2010.

A. Hirose. Complex-valued neural networks: The merits and their origins. In *2009 International Joint Conference on Neural Networks*, pp. 1237–1244, June 2009. doi: 10.1109/IJCNN.2009.5178754.

Akira Hirose. *Complex-Valued Neural Networks: Theories and Applications (Series on Innovative Intelligence, 5)*. World Scientific Press, 2004. ISBN 9812384642.

Masaki Kobayashi. Dual-numbered hopfield neural networks. *IEEJ Transactions on Electrical and Electronic Engineering*, 2017. ISSN 1931-4981.

T. Nitta. A back-propagation algorithm for complex numbered neural networks. In *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*, volume 2, pp. 1649–1652 vol.2, Oct 1993.

Tohru Nitta. Learning dynamics of the complex-valued neural network in the neighborhood of singular points. *Journal of Computer and Communications*, 2(1):27–32, 2014. doi: 10.4236/jcc.2014.21005.

C. A. Popa. Complex-valued convolutional neural networks for real-valued image classification. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 816–822, May 2017. doi: 10.1109/IJCNN.2017.7965936.

Andy M. Sarroff, Victor Shepardson, and Michael A. Casey. Learning representations using complex-valued nets. *CoRR*, abs/1511.06351, 2015.

Chiheb Trabelsi, Sandeep Subramanian, Negar Rostamzadeh, Soroush Mehri, Dmitriy Serdyuk, João Felipe Santos, Yoshua Bengio, and Christopher Pal. Deep complex networks. 2017.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International Conference on Machine Learning (ICML)*, volume 48, pp. 2071–2080, 2016.

Théo Trouillon, Christopher R. Dance, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Knowledge graph completion via complex tensor factorization. *CoRR*, abs/1702.06879, 2017.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *CoRR*, abs/1410.3916, 2014.

Scott Wisdom, Thomas Powers, John Hershey, Jonathan Le Roux, and Les Atlas. Full-capacity unitary recurrent neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 29*, pp. 4880–4888. Curran Associates, Inc., 2016.

Hans-Georg Zimmermann, Alexey Minin, and Victoria Kusherbaeva. Comparison of the complex valued and real valued neural networks trained with gradient descent and random search algorithms. In *ESANN*, 2011.

Table 1: Test accuaracy of multi-layer perceptron conisting of $k + 2$ dense layers each with 64 units on MNIST digit classification task. Selected from best of 10 runs with each run 100 epochs to converge.

Hidden layers k	Real parameters $p_{\mathbb{R}}$	Activation function φ	MNIST	
			$\mathbb{R}$	$\mathbb{C}$
k = 0	50,816	*identity*	0.9282	0.9509
		tanh	0.9761	0.9551
		relU	0.9780	0.9710
		$\|z\|^2$	0.9789	0.9609
		$\|z\|$	0.9770	0.9746
k = 2	59,008	*identity*	0.9274	0.9482
		tanh	0.9795	0.8923
		relU	0.9804	0.9742
		$\|z\|^2$	0.9713	0.6573
		$\|z\|$	0.9804	0.9755
k = 4	67,200	*identity*	0.9509	0.9468
		tanh	0.9802	0.2112
		relU	0.9816	0.9768
		$\|z\|^2$	0.8600	0.2572
		$\|z\|$	0.9789	0.9738
k = 8	83,584	*identity*	0.9242	0.1771
		tanh	0.9796	0.1596
		relU	0.9798	0.9760
		$\|z\|^2$	0.0980	0.098
		$\|z\|$	0.9794	0.1032

APPENDIX

Table 2: Test accuaracy of multi-layer perceptron conisting of $k + 2$ dense layers each with 64 units on Reuters topic classification task. Selected from best of 10 runs with each run 100 epochs to converge.

Hidden layers k	Real parameters $p_{\mathbb{R}}$	Activation function φ	Reuters	
			$\mathbb{R}$	$\mathbb{C}$
k = 0	642,944	*identity*	0.8116	0.7939
		tanh	0.8117	0.7912
		relU	0.8081	0.7934
		$\|z\|^2$	0.8050	0.7885
		$\|z\|$	0.8068	0.7992
k = 2	651,136	*identity*	0.8005	0.7836
		tanh	0.7978	0.7320
		relU	0.7921	0.7854
		$\|z\|^2$	0.7725	0.6874
		$\|z\|$	0.7996	0.7823
k = 4	659,328	*identity*	0.7925	0.7787
		tanh	0.7814	0.4199
		relU	0.7734	0.7671
		$\|z\|^2$	0.5895	0.0650
		$\|z\|$	0.7863	0.7694
k = 8	675,712	*identity*	0.7929	0.7796
		tanh	0.7542	0.1861
		relU	0.7555	0.7676
		$\|z\|^2$	0.0053	0.0053
		$\|z\|$	0.7671	0.7524

Table 3: Test accuaracy of multi-layer perceptron conisting of $k+2$ dense layers each with 128 units on CIFAR-10 image classification task. Selected from best of 10 runs with each run 100 epochs to converge.

Hidden layers k	Real parameters $p_{\mathbb{R}}$	Activation function φ	CIFAR-10	
			$\mathbb{R}$	$\mathbb{C}$
k = 0	394,496	*identity*	0.4044	0.1063
		tanh	0.4885	0.1431
		relU	0.4902	0.4408
		$\|z\|^2$	0.5206	0.1000
		$\|z\|$	0.5256	0.1720
k = 2	427,264	*identity*	0.4039	0.1000
		tanh	0.5049	0.1672
		relU	0.5188	0.496
		$\|z\|^2$	0.1451	0.1361
		$\|z\|$	0.5294	0.1000
k = 4	460,032	*identity*	0.4049	0.1000
		tanh	0.4983	0.1549
		relU	0.8445	0.6810
		$\|z\|^2$	0.1000	0.1000
		$\|z\|$	0.5273	0.1000
k = 8	525,568	*identity*	0.4005	0.1027
		tanh	0.4943	0.1365
		relU	0.5072	0.4939
		$\|z\|^2$	0.1000	0.1000
		$\|z\|$	0.5276	0.1000

Table 4: Test accuaracy of multi-layer perceptron conisting of $k + 2$ dense layers each with 128 real and 64 complex units on CIFAR-100 image classification task. Selected from best of 10 runs with each run 100 epochs to converge.

Hidden layers k	Real parameters $p_{\mathbb{R}}$	Activation function φ	CIFAR-100	
			$\mathbb{R}$	$\mathbb{C}$
k = 0	406,016	*identity*	0.1758	0.0182
		tanh	0.2174	0.0142
		relU	0.1973	0.1793
		$\|z\|^2$	0.2314	0.0158
		$\|z\|$	0.2423	0.0235
k = 2	438,784	*identity*	0.1720	0.0100
		tanh	0.2314	0.0146
		relU	0.2400	0.2123
		$\|z\|^2$	0.0143	0.0123
		$\|z\|$	0.2411	0.0100
k = 4	471,552	*identity*	0.1685	0.0100
		tanh	0.2178	0.0157
		relU	0.2283	0.2059
		$\|z\|^2$	0.0109	0.0100
		$\|z\|$	0.2313	0.0100
k = 8	537,088	*identity*	0.1677	0.0100
		tanh	0.2000	0.0130
		relU	0.2111	0.1956
		$\|z\|^2$	0.0100	0.0100
		$\|z\|$	0.2223	0.0100

Table 5: Test accuaracy of multi-layer perceptron conisting of $k + 2$ dense layers with an overall budget of $500,000$ real-valued parameters on MNIST digit classification. Selected from best of 10 runs with each run 100 epochs to converge.

Hidden layers k	Units		Activation function φ	MNIST	
	$m_{\mathbb{R}}$	$m_{\mathbb{C}}$		$\mathbb{R}$	$\mathbb{C}$
k = 0	500000	250000	*identity*	0.0000	0.0000
			tanh	0.0000	0.0000
			relU	0.0000	0.0000
			$\|z\|^2$	0.0000	0.0000
			$\|z\|$	0.0000	0.0000
k = 2	500000	250000	*identity*	0.0000	0.0000
			tanh	0.0000	0.0000
			relU	0.0000	0.0000
			$\|z\|^2$	0.0000	0.0000
			$\|z\|$	0.0000	0.0000
k = 4	500000	250000	*identity*	0.0000	0.0000
			tanh	0.0000	0.0000
			relU	0.0000	0.0000
			$\|z\|^2$	0.0000	0.0000
			$\|z\|$	0.0000	0.0000
k = 8	500000	250000	*identity*	0.0000	0.0000
			tanh	0.0000	0.0000
			relU	0.0000	0.0000
			$\|z\|^2$	0.0000	0.0000
			$\|z\|$	0.0000	0.0000

Table 6: Test accuaracy of multi-layer perceptron conisting of $k + 2$ dense layers with an overall budget of $500,000$ real-valued parameters on Reuters topic classification. Selected from best of 10 runs with each run 100 epochs to converge.

Hidden layers k	Units		Activation function φ	Reuters	
	$m_{\mathbb{R}}$	$m_{\mathbb{C}}$		$\mathbb{R}$	$\mathbb{C}$
k = 0	500000	250000	*identity*	0.0000	0.0000
			tanh	0.0000	0.0000
			relU	0.0000	0.0000
			$\|z\|^2$	0.0000	0.0000
			$\|z\|$	0.0000	0.0000
k = 2	500000	250000	*identity*	0.0000	0.0000
			tanh	0.0000	0.0000
			relU	0.0000	0.0000
			$\|z\|^2$	0.0000	0.0000
			$\|z\|$	0.0000	0.0000
k = 4	500000	250000	*identity*	0.0000	0.0000
			tanh	0.0000	0.0000
			relU	0.0000	0.0000
			$\|z\|^2$	0.0000	0.0000
			$\|z\|$	0.0000	0.0000
k = 8	500000	250000	*identity*	0.0000	0.0000
			tanh	0.0000	0.0000
			relU	0.0000	0.0000
			$\|z\|^2$	0.0000	0.0000
			$\|z\|$	0.0000	0.0000

Table 7: Test accuaracy of multi-layer perceptron conisting of $k + 2$ dense layers with an overall budget of $500,000$ real-valued parameters on CIFAR-10 image classification. Selected from best of 10 runs with each run 100 epochs to converge.

Hidden layers k	Units		Activation function φ	CIFAR-10	
	$m_{\mathbb{R}}$	$m_{\mathbb{C}}$		$\mathbb{R}$	$\mathbb{C}$
k = 0	500000	250000	*identity*	0.0000	0.0000
			tanh	0.0000	0.0000
			relU	0.0000	0.0000
			$\|z\|^2$	0.0000	0.0000
			$\|z\|$	0.0000	0.0000
k = 2	500000	250000	*identity*	0.0000	0.0000
			tanh	0.0000	0.0000
			relU	0.0000	0.0000
			$\|z\|^2$	0.0000	0.0000
			$\|z\|$	0.0000	0.0000
k = 4	500000	250000	*identity*	0.0000	0.0000
			tanh	0.0000	0.0000
			relU	0.0000	0.0000
			$\|z\|^2$	0.0000	0.0000
			$\|z\|$	0.0000	0.0000
k = 8	500000	250000	*identity*	0.0000	0.0000
			tanh	0.0000	0.0000
			relU	0.0000	0.0000
			$\|z\|^2$	0.0000	0.0000
			$\|z\|$	0.0000	0.0000

Table 8: Test accuaracy of multi-layer perceptron conisting of $k + 2$ dense layers with an overall budget of $500,000$ real-valued parameters on CIFAR-100 image classification. Selected from best of 10 runs with each run 100 epochs to converge.

Hidden layers k	Units		Activation function φ	CIFAR-100	
	$m_{\mathbb{R}}$	$m_{\mathbb{C}}$		$\mathbb{R}$	$\mathbb{C}$
k = 0	500000	250000	*identity*	0.0000	0.0000
			tanh	0.0000	0.0000
			relU	0.0000	0.0000
			$\|z\|^2$	0.0000	0.0000
			$\|z\|$	0.0000	0.0000
k = 2	500000	250000	*identity*	0.0000	0.0000
			tanh	0.0000	0.0000
			relU	0.0000	0.0000
			$\|z\|^2$	0.0000	0.0000
			$\|z\|$	0.0000	0.0000
k = 4	500000	250000	*identity*	0.0000	0.0000
			tanh	0.0000	0.0000
			relU	0.0000	0.0000
			$\|z\|^2$	0.0000	0.0000
			$\|z\|$	0.0000	0.0000
k = 8	500000	250000	*identity*	0.0000	0.0000
			tanh	0.0000	0.0000
			relU	0.0000	0.0000
			$\|z\|^2$	0.0000	0.0000
			$\|z\|$	0.0000	0.0000

Table 9: Test accuaracy of Memory Networks (Weston et al., 2014) in complex and real version on the first three bAbI tasks. Selected from best of 30 runs with each run 100 epochs to converge.

	Parameters $p_{\mathbb{R}}$	Supporting facts		
		One	Two	Three
Memory Network	24,750	**0.9920**	0.4030	0.3830
Memory Network (small)	18,716	0.9570	0.4250	0.3820
Memory Network (large)	24,456	0.9790	**0.4420**	**0.3920**