

---

# EVALUATION OF COMPLEX-VALUED NEURAL NETWORKS ON REAL-VALUED CLASSIFICATION TASKS

---

A PREPRINT

**Nils Mönning**

Department of Computer Science  
University of York  
York, YO10 5GH, United Kingdom  
nm819@york.ac.uk

**Suresh Manandhar**

Department of Computer Science  
University of York  
York, YO10 5GH, United Kingdom  
suresh.manandhar@york.ac.uk

November 30, 2018

## ABSTRACT

Complex-valued neural networks are not a new concept, however, the use of real-valued models has often been favoured over complex-valued models due to difficulties in training and performance. When comparing real-valued versus complex-valued neural networks, existing literature often ignores the number of parameters, resulting in comparisons of neural networks with vastly different sizes. We find that when real and complex neural networks of similar capacity are compared, complex models perform equal to or slightly worse than real-valued models for a range of real-valued classification tasks. The use of complex numbers allows neural networks to handle noise on the complex plane. When classifying real-valued data with a complex-valued neural network, the imaginary parts of the weights follow their real parts. This behaviour is indicative for a task that does not require a complex-valued model. We further investigated this in a synthetic classification task. We can transfer many activation functions from the real to the complex domain using different strategies. The weight initialisation of complex neural networks, however, remains a significant problem.

## 1 Introduction

In recent years complex-valued neural networks have been successfully applied to a variety of tasks, specifically in signal processing where the input data has a natural interpretation in the complex domain. Complex-valued neural networks are often compared to real-valued networks. We need to ensure that these architectures are comparable in their model size and capacity. This aspect of the comparison is rarely studied or only dealt with superficially. A metric for their capacity is the number of real-valued parameters. The introduction of complex numbers into a model increases the computational complexity and the number of real-valued parameters, but assumes a certain structure of weights and data input.

This paper explores the performance of complex-valued multi-layer perceptron (MLP) with varying depth and width. We consider the number of parameters and choice of activation function in benchmark classification tasks of real-valued data. We present a complex-valued multi-layer perceptron architecture and its training process. We consider various activation functions and the number of real-valued parameters in both the complex and real case. We propose two methods to construct comparable networks: 1) by setting a fixed number of real-valued neurons per layer and 2) by setting a fixed budget of real-valued parameters. As benchmark task we choose MNIST digit classification

[18], CIFAR-10 image classification [17], CIFAR-100 image classification [17], Reuters newswire topic classification (*Reuters-21578, Distribution 1.0*). We use classification of synthetic data for further investigation.

## 2 Related Literature

Complex-valued neural networks were first formally described by Clarke [8]. Several authors have since proposed complex versions of the backpropagation algorithm based on gradient descent [6, 10, 19]. Inspired by work on multi-valued threshold logic [1] from the 1970s, a multi-valued neuron and neural network was defined by Aizenberg et al. [4, 3] who also extends this idea to quaternions. In the 2000s, complex neural networks were successfully applied to a variety of tasks [22, 12, 21, 25]. These tasks mainly involved the processing and analysis of complex-valued data or data with an intuitive mapping to complex numbers. Particularly, images and signals in their wave form or Fourier transformation were used as input data to complex-valued neural networks [15].

Another natural application of complex numbers are convolutions [7] which are used in image and signal processing. While real convolutions are widely used in deep learning for image processing, it is possible to replace them with complex convolutions [26, 13, 23, 14].

The properties of complex numbers and matrices can be used to define constraints on deep learning models. Introduced by Arjovsky et al. [5], and further developed by Wisdom et al. [29], complex-valued recurrent networks, that constrain their weights to be unitary matrices, reduce the impact of vanishing or exploding gradients.

More recently, complex-valued neural networks have been used to learn filters as embeddings of images and audio signals [27, 24, 9]. In addition, tensor factorisation has been applied to complex embeddings to predict the edges between entities of knowledge bases [28].

Despite their successes, complex neural networks have been less popular than their real-valued counter-parts. Potentially, because the training process and architecture design are less intuitive, which stems from stricter requirements for the differentiability of activation functions in the complex plane [31, 16, 20].

When comparing complex-valued neural networks with real-valued neural networks, many publications ignore the number of parameters altogether [3], compare only the number of parameters of the entire model [26], or do not distinguish between complex- or real-valued parameters and units [30]. From the perspective of this paper such comparisons are equivalent to comparing models of different sizes. We systematically explore the performance of multi-layered perceptrons on simple classification tasks in consideration of the activation function, width and depth.

## 3 Complex-Valued Neural Networks

We define a complex-valued neuron analogous to its real-valued counter-part and consider its differences in structure and training. The complex neuron can be defined as:

$$o = \phi(x \cdot w + b) \quad (1)$$

with an activation function  $\phi$  applied to the 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:

$$o = \phi(xW + b) \quad (2)$$

with an input  $x \in \mathbb{C}^n$ ,  $W \in \mathbb{C}^{n \times m}$ ,  $b \in \mathbb{C}^m$ .

The activation function  $\phi$  in the above definitions can be a function  $\phi : \mathbb{C} \rightarrow \mathbb{R}$  or  $\phi : \mathbb{C} \rightarrow \mathbb{C}$ . We will consider the choice of the non-linear activation function  $\phi$  in more detail in Section 6. In this work, we choose a simple real-valued loss function but complex-valued loss functions could be subject for future work. There is no total ordering on the field of complex numbers, since  $i^2 = -1$ . A complex-valued loss function would require defining a partial ordering on complex numbers (similar to a linear matrix inequality).

The training process in the complex domain differs, because activation functions are often not entirely complex-differentiable.

**Definition 3.1.** Analogous to a real function, a complex function  $f : \mathbb{C} \rightarrow \mathbb{C}$  at a point  $z_0$  of an open subset  $\Omega \subset \mathbb{C}$  is *complex-differentiable* if there exists a limit such that

$$f'(z_0) = \lim_{z \rightarrow z_0} \frac{f(z) - f(z_0)}{z - z_0} \quad (3)$$

If the function  $f$  is complex-differentiable at all points of  $\Omega$  it is called *holomorphic*. While in the real-valued case the existence of a limit is sufficient for a function to be differentiable, the complex definition in Equation 3 implies a stronger property.

**Definition 3.2.** 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:

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}, \quad -\frac{\partial u}{\partial y} = \frac{\partial v}{\partial x} \quad (4)$$

We represent a complex number  $z \in \mathbb{C}$  with 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 the (partial) derivatives must also satisfy the Cauchy-Riemann Equations. That also means that a function can be non-holomorphic (i.e. not complex-differentiable) in  $z$ , but still be analytic in its parts  $x, y$ . Hence, to satisfy the Cauchy-Riemann Equations, real differentiability of functions  $u(x, y)$  and  $v(x, y)$  is not a sufficient condition to satisfy the Cauchy-Riemann Equations (Definition 3.2).

In order to apply the *chain rule* for non-holomorphic functions, the property of many non-holomorphic functions to be differentiable with respect to their real and imaginary parts can be utilised. We consider the complex function  $f(z, \bar{z})$  to be a function of  $z$  and its complex conjugate  $\bar{z}$ . Effectively, we choose a different basis for our partial derivatives.

$$\frac{\partial}{\partial z} = \frac{1}{2} \left( \frac{\partial}{\partial x} - i \frac{\partial}{\partial y} \right), \quad \frac{\partial}{\partial \bar{z}} = \frac{1}{2} \left( \frac{\partial}{\partial x} + i \frac{\partial}{\partial y} \right) \quad (5)$$

These derivatives are a consequence of Wirtinger calculus (or CR-calculus). They allow the application of the chain rule to many non-holomorphic functions for multiple complex variables  $z_i$ :

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

Many non-holomorphic functions are also not entirely differentiable with respect to their real parts. The general practice of computing gradients only at specific points allows using a wide range of complex activation functions. The training process, however, can become numerically unstable. The unstable training process makes it necessary to devise special methods to avoid problematic regions of the function. The Wirtinger calculus, described above, provides an alternative method for computing the gradient that also improves the stability of the training process.

## 4 Interaction of Parameters

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 equivalently as magnitude  $|z| = \sqrt{x^2 + y^2} = r$  and phase (angle)  $\varphi = \arctan(\frac{y}{x})$ . Consequently, any complex-valued function on one or more complex-variables can be expressed as a function on two real variables  $f(z) = f(x, y) = f(r, \varphi)$ .

Despite the straight forward use and representation in neural networks, complex numbers define an interaction between the two parts. Consider the operations necessary in the regression outlined in Equation 2 to be composed of real and imaginary parts (or, equivalently, magnitude and phase). Each element  $z_1 \in \mathbb{C}$  of the weight matrix  $W \in \mathbb{C}^{n \times m}$  interacts with an element  $z_2 \in \mathbb{C}$  of an input  $x \in \mathbb{C}^n$ :

$$\begin{aligned} z_1 z_2 &= (a + ib)(c + id) = (ac - bd) + i(ad + bc), \\ z_1 + z_2 &= (a + ib) + (c + id) = (a + c) + i(b + d) \end{aligned} \quad (7)$$

In an equivalent representation with Euler's constant  $e^{i\varphi} = \cos(\varphi) + i\sin(\varphi)$  as polar form.

$$\begin{aligned} z_1 z_2 &= (r_1 e^{i\varphi_1})(r_2 e^{i\varphi_2}) = (r_1 r_2 e^{i\varphi_1 + i\varphi_2}), \\ z_1 + z_2 &= (r_1 e^{i\varphi_1}) + (r_2 e^{i\varphi_2}) \\ &= r_1 \cos(\varphi_1) + r_2 \cos(\varphi_2) + i(r_1 \sin(\varphi_1) + r_2 \sin(\varphi_2)) \end{aligned} \quad (8)$$

Complex parameters increase the computational complexity of a neural network as more operations are required. Instead of a single real-valued multiplication, up to four real multiplications and two real additions are required. As can be seen in Equations 7 and 8 the computational complexity can be significantly reduced depending on the implementation and representation chosen.

Consequently, simply doubling the number of real-valued parameters per layer is not sufficient to achieve the same effect as in complex-valued neural networks. This is illustrated when a complex number  $z = a + ib$  is expressed in a equivalent matrix representation. Specifically, as  $2 \times 2$  matrix  $M$  in the ring of  $M_2(\mathbb{R})$ :

$$M_{(a+ib)} = \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \quad \text{such that} \quad M_{(0+i1)} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \quad M_{(1+i0)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (9)$$

$$M_{(a+ib)} M_{(c+id)} = \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \begin{bmatrix} c & -d \\ d & c \end{bmatrix} = \begin{bmatrix} ac - bd & bc + dc \\ bc + dc & ac - bd \end{bmatrix} \quad (10)$$

This *augmented representation* facilitates computing the multiplication of an input  $x$  with a complex-valued weight matrix  $W$  as:

$$xW = \begin{bmatrix} \text{Re}(x) & -\text{Im}(x) \\ \text{Im}(x) & \text{Re}(x) \end{bmatrix} \begin{bmatrix} \text{Re}(W) \\ \text{Im}(W) \end{bmatrix} = \begin{bmatrix} \text{Re}(x)\text{Re}(W) - \text{Im}(x)\text{Im}(W) \\ \text{Im}(x)\text{Re}(W) + \text{Re}(x)\text{Im}(W) \end{bmatrix} \quad (11)$$

This interaction consequently means that architecture design needs to be reconsidered in order to facilitate this structure. A deep learning architecture that performs well with real-valued parameters may not work for complex-valued parameters and vice versa. Models that do not facilitate the structure or tasks that do not require complex-valued representations will not improve in performance.

Our experiments show that real-valued data does not require this structure. The imaginary part of the input  $\text{Im}(x)$  is zero, so Equations 7 and 11 simplify to:

$$\text{Re}(xW) = \text{Re}(x)\text{Re}(W), \quad \text{Im}(xW) = \text{Re}(x)\text{Im}(W) \quad (12)$$

For the training this means that the real parts  $\text{Re}(x)$  and  $\text{Re}(W)$  dominate the overall classification of a real-valued data point. In later sections we discuss our experiment results and illustrate the training with a synthetic classification task.

## 5 Capacity

The number of (real-valued) parameters is a metric to quantify the capacity of a network in its ability to approximate structurally complex functions. With too many parameters the model tends to overfit the data while with too few parameters it tends to underfit.

A consequence of representing a complex number  $a + ib$  using real numbers  $(a, b)$  is that the number of real parameters of each layer is doubled:  $p_{\mathbb{C}} = 2p_{\mathbb{R}}$ . The number of real-valued parameters per layer should be equal (or at least as close as possible) between the real-valued and its complex-valued architecture. This ensures that models have the same capacity. Performance differences are caused by introducing complex numbers as parameters and not by a capacity difference.

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, then the number of parameters of a real-valued layer  $p_{\mathbb{R}}$  and of a complex layer  $p_{\mathbb{C}}$  is given by

$$p_{\mathbb{R}} = (n \times m) + m, \quad p_{\mathbb{C}} = 2(n \times m) + 2m \quad (13)$$

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

$$\begin{aligned} p_{\mathbb{R}} &= n \times m + k(m \times m) + m \times c, \\ p_{\mathbb{C}} &= 2(n \times m) + 2k(m \times m) + 2(m \times c) \end{aligned} \quad (14)$$

At first glance designing comparable multi-layer neural network architectures, i.e. with the same number of real-valued parameters in each layer, is trivial. However, halving the number of neurons in every layer will not achieve parameter comparability. The number of neurons define the output dimensions of a layer and the following layer's input dimension. We addressed this problem by choosing MLP architectures with an even number of hidden layers  $k$  and the number of neurons per layer to be alternating between  $m$  and  $\frac{m}{2}$ . We receive the same number of real parameters in each layer of a complex-valued MLP compared to a real-valued network. Let us consider the dimensions of outputs and weights with  $k = 4$  hidden layers. For the real-valued case:

$$\begin{aligned} & \text{Input layer} \quad \text{Hidden layer} \\ & (1 \times n) \overbrace{(n \times m_1)} \rightarrow (1 \times m_1) \overbrace{(m_1 \times m_2)} \\ & \text{Hidden layer} \quad \text{Hidden layer} \\ & \rightarrow (1 \times m_2) \overbrace{(m_2 \times m_3)} \rightarrow (1 \times m_3) \overbrace{(m_3 \times m_4)} \\ & \text{Hidden layer} \quad \text{Output layer} \\ & \rightarrow (1 \times m_4) \overbrace{(m_4 \times m_5)} \rightarrow (1 \times m_5) \overbrace{(m_5 \times c)} \\ & \text{Model output} \\ & \rightarrow \overbrace{(1 \times c)} \end{aligned} \quad (15)$$

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

$$\begin{aligned} & (1 \times n) \overbrace{(n \times \frac{m_1}{2})} \rightarrow (1 \times \frac{m_1}{2}) \overbrace{(\frac{m_1}{2} \times m_2)} \\ & \rightarrow (1 \times m_2) \overbrace{(m_2 \times \frac{m_3}{2})} \rightarrow (1 \times \frac{m_3}{2}) \overbrace{(\frac{m_3}{2} \times m_4)} \\ & \rightarrow (1 \times m_4) \overbrace{(m_4 \times \frac{m_5}{2})} \rightarrow (1 \times \frac{m_5}{2}) \overbrace{(\frac{m_5}{2} \times c)} \\ & \rightarrow (1 \times c) \end{aligned} \quad (16)$$

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 MLP with an even number  $k \geq 0$  of hidden layers such that the network's parameters are within that budget. The  $k$  hidden layers and the input layer have the same number of real or complex neurons  $m_{\mathbb{R}} = m_{\mathbb{C}}$ . The number of neurons in the last layer is defined by the number of classes  $c$ .

$$m_{\mathbb{R}} = \begin{cases} -\frac{n+c}{2k} + \sqrt{(\frac{n+c}{2k})^2 + \frac{p_{\mathbb{R}}}{k}}, & \text{if } k > 0 \\ \frac{p_{\mathbb{R}}}{n+c}, & \text{otherwise} \end{cases} \quad (17)$$

$$m_{\mathbb{C}} = \begin{cases} -\frac{n+c}{2k} + \sqrt{(\frac{n+c}{2k})^2 + \frac{p_{\mathbb{R}}}{2k}}, & \text{if } k > 0 \\ \frac{p_{\mathbb{R}}}{2(n+c)}, & \text{otherwise} \end{cases} \quad (18)$$

## 6 Activation Functions

In any neural network 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 that activation functions have on the overall performance. An important theorem to be considered for the choice of activation function is the Liouville Theorem. The theorem states that any bounded holomorphic function  $f : \mathbb{C} \rightarrow \mathbb{C}$  (that is differentiable on the entire complex plane) must be constant. Hence, we need to choose unbounded and/or non-holomorphic activation functions.

To investigate the performance of complex models assuming a function which is linearly separable in the complex parameters we chose the identity function. This allows us to identify tasks that may not be linearly separable in  $\mathbb{R}$  using  $m_{\mathbb{R}}$  neurons, but are linearly separable in  $\mathbb{C}$  using  $m_{\mathbb{C}}$  neurons. An example would be the approximation of the XOR function [2]. The hyperbolic tangent is a well-studied function and defined for both complex and real numbers. The rectifier linear unit is also well understood and frequently used in a real-valued setting, but has not been considered in a complex-valued setting. It illustrates separate application on the two parts of a complex number. The magnitude and squared magnitude functions are chosen to map complex numbers to real numbers.

- Identity (or no activation function):

$$\phi(z) = z \quad (19)$$

- Hyperbolic tangent:

$$\phi(z) = \tanh(z) = \frac{\sinh(z)}{\cosh(z)} = \frac{e^z - e^{-z}}{e^z + e^{-z}} = \frac{e^{2z} - 1}{e^{2z} + 1} \quad (20)$$

- Rectifier linear unit (ReLU):

$$\begin{aligned} \phi(z) &= \text{ReLU}(z) = \text{ReLU}(\text{Re}(z)) + \text{ReLU}(\text{Im}(z))j \\ &= \max(0, \text{Re}(z)) + \max(0, \text{Im}(z))j \end{aligned} \quad (21)$$

- Intensity (or magnitude squared):

$$\phi(z) = |z|^2 = x^2 + y^2 \quad (22)$$

- Magnitude (or complex absolute):

$$\phi(z) = |z| = \sqrt{x^2 + y^2} \quad (23)$$

Before applying the logistic function in the last layer we use another function  $\phi : \mathbb{C} \rightarrow \mathbb{R}$  to receive a real-valued loss. We chose the squared magnitude  $\phi(z) = |z|^2$ . The intensity or probability amplitude of two interfering waves gives us a geometrically and probabilistically interpretable output.

$$\text{sigmoid}(|z|^2) = \frac{1}{1 + e^{-x^2 - y^2}} \quad (24)$$

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

$$\text{softmax}(|z_j|^2) = \frac{e^{x^2 + y^2}}{\sum_{i=1}^c e^{x^2 + y^2}} \quad (25)$$

## 7 Experiments

To compare real and complex-valued multi-layer perceptrons (Figure 1) we investigated them in various classification tasks. In all of the following experiments the task was to assign a single class to each real-valued data point using complex-valued multi-layer perceptrons:

- Experiment 1: We tested MLPs with  $k = 0, 2, 4, 8$  hidden layers, fixed width of units in each layer in real-valued architectures and alternating 64 and 32 units in complex-valued architectures (see section 5). We applied no fixed parameter budget. We tested the models on MNIST digit classification, CIFAR-10 Image classification, CIFAR-100 image classification and Reuters topic classification. Reuters topic classification and MNIST digit classification use 64 units per layer, CIFAR-10 and CIFAR-100 use 128 units per layer.
- Experiment 2: We tested MLPs with fixed budget of 500,000 real-valued parameters. The MLPs have variable width according to the depth and the parameters and are tested on MNIST digit classification, CIFAR-10 Image classification, CIFAR-100 image classification and Reuters topic classification. All tested activation functions are introduced in Section 6. We rounded the units in Equations 17 and 18 to the next integer.

We used the weight initialisation discussed by Trabelsi et al. [26] for all our experiments. To reduce the impact of the initialisation we trained each model 10 times. Each run trained the model over 100 epochs with an Adam optimisation. We used categorical or binary cross entropy as a loss function depending on the task. We used  $\text{sigmoid}(|z|^2)$  or  $\text{softmax}(|z|^2)$  as the activation function for the last fully-connected layer.

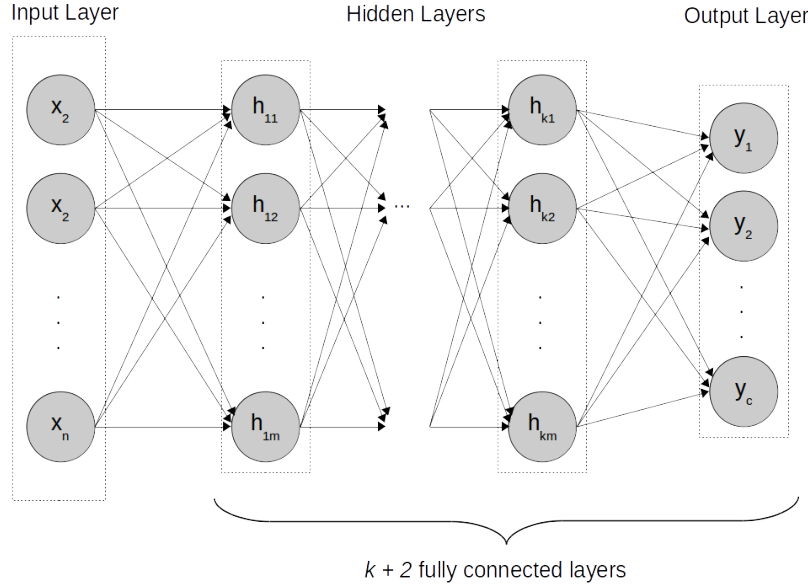


Figure 1: Multi-layer perceptron with  $k$  hidden layers. The number of hidden units per layer  $m$  computed with equation 14 for Experiment 1 and Equations 17 resp. 18 for Experiment 2.

## 8 Results

Tables 1, 2, 3, 4 show the results for MLPs with variable depth and fixed width and no parameter budget (Experiment 1). Tables 5, 6, 7, 8 show the results for MLPs with variable width according to depth and a fixed parameter budget of 500,000 real-valued parameters (experiment 2). In our experiments the achieved accuracy of complex- and real-valued multi-layer perceptrons are close to each other. Nevertheless, the real networks consistently outperform complex-valued networks and the complex-valued neural networks often fail to learn any structure from the data.

Complex-valued MLPs can be used to classify short dependencies (e.g MNIST digit classification) or a short text as a bag-of-words (e.g. Reuters topic classification). For the two image classification tasks CIFAR-10 and CIFAR-100 the results indicate that a complex-valued MLP does not learn any structure in the data. These two tasks require larger weight matrices in the first layer and weight initialisation is still a significant problem.

The best non-linearity in complex neural network is the rectifier linear unit *relu* applied to the imaginary and real parts, similarly to the real-valued models. *Identity* and hyperbolic tangents *tanh* outperform *relu* - particularly in the real-valued case. However, the results using the rectifier linear unit *relu* are much more stable. Despite the similarity of the activation functions  $|z|^2$  and  $|z|$ , their performance in all tasks differ significantly. The magnitude  $|z|$  consistently outperforms the squared magnitude  $|z|^2$ . In these classification benchmarks the activation function is the deciding factor for the overall performance of a given model. The activation may allow the network to recover from a bad initialisation and use the available parameters appropriately. An example would be the *relu* activation in CIFAR task of Experiments 1 and 2 (Tables 3, 4, 7, 8)

As expected, we observe that with a fixed number of neurons per layer (Experiment 1) and increasing depth, the complex- and real-valued accuracy increases. As we are increasing the total number of parameters, the model capacity increases. An exception here is Reuters topic classification where the performance decreases with increasing depth. When choosing the number of neurons per layer according to a given parameter budget (Experiment 2 using Equations 17, 18), the performance decreases significantly as model depth increases. In consideration with the results from Experiment 1, the width of each layer is more important than the overall depth of the complete network.

We observed that the performance variance between the 10 initialisations is very high. We hypothesized that weight initialisation in complex MLPs becomes much more difficult with increasing depth. Hence, their performance is highly unstable. We confirmed this by training a complex MLP ( $k = 2$ , *tanh*) with 100 runs (instead of 10 runs) on the Reuters classification task. The result shows a similar behaviour to the other results: The performance gap decreases

Table 1: Test accuracy of a multi-layer perceptron consisting of  $k + 2$  layers each with 64 neurons (alternating 64 and 32 neurons in the complex MLP) and an output layer with  $c = 10$  neurons on MNIST digit classification task (Experiment 1). Selected best of 10 runs. Each run was trained for 100 epochs.

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

Table 2: Test accuracy of a multi-layer perceptron consisting of  $k + 2$  layers each with 64 neurons (alternating 64 and 32 neurons in the complex MLP) and an output layer with  $c = 46$  neurons on Reuters topic classification (Experiment 1). Selected best of 10 runs. Each run was trained for 100 epochs.

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



Table 3: Test accuracy of a multi-layer perceptron consisting of  $k + 2$  layers each with 128 neurons (alternating 128 and 64 neurons in the complex MLP) and an output layer with  $c = 10$  neurons on CIFAR-10 image classification task (Experiment 1). Selected best of 10 runs. Each run was trained for 100 epochs.

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

Table 4: Test accuracy of a multi-layer perceptron consisting of  $k + 2$  layers each with 128 neurons (alternating 128 and 64 neurons in the complex MLP) and an output layer with  $c = 100$  neurons on CIFAR-100 image classification task (Experiment 1). Selected best of 10 runs. Each run was trained for 100 epochs.

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

Table 5: Test accuracy of a multi-layer perceptron consisting of  $k + 2$  dense layers with an overall budget of 500,000 real-valued parameters on MNIST digit classification (experiment 2). The last layer consists of  $c = 10$  neurons. Selected best of 10 runs. Each run was trained for 100 epochs.

Hidden layers $k$	Units		Activation function $\varphi$	CIFAR-10	
	$m_R$	$m_C$		$\mathbb{R}$	$\mathbb{C}$
$k = 0$	630	315	<i>identity</i>	0.9269	0.9464
			<i>tanh</i>	0.9843	0.9467
			<i>relu</i>	0.9846	0.9828
			$ z ^2$	0.9843	0.9654
			$ z $	0.9857	0.9780
$k = 2$	339	207	<i>identity</i>	0.9261	0.9427
			<i>tanh</i>	0.9852	0.6608
			<i>relu</i>	0.9878	0.9835
			$ z ^2$	0.9738	0.8331
			$ z $	0.9852	0.9748
$k = 4$	268	170	<i>identity</i>	0.9254	0.2943
			<i>tanh</i>	0.9838	0.2002
			<i>relu</i>	0.9862	0.9825
			$ z ^2$	0.8895	0.2875
			$ z $	0.9846	0.9870
$k = 8$	205	134	<i>identity</i>	0.9250	0.1136
			<i>tanh</i>	0.9810	0.1682
			<i>relu</i>	0.9851	0.9824
			$ z ^2$	0.0980	0.0980
			$ z $	0.9803	0.1135

Table 6: Test accuracy of a multi-layer perceptron consisting of  $k + 2$  dense layers with an overall budget of 500,000 real-valued parameters on Reuters topic classification (experiment 2). The last layer consists of  $c = 46$  neurons. Selected best of 10 runs. Each run was trained for 100 epochs.

Hidden layers $k$	Units		Activation function $\varphi$	Reuters	
	$m_R$	$m_C$		$\mathbb{R}$	$\mathbb{C}$
$k = 0$	50	25	<i>identity</i>	0.8072	0.7970
			<i>tanh</i>	0.8112	0.7832
			<i>relu</i>	0.8054	0.7925
			$ z ^2$	0.8037	0.7929
			$ z $	0.8059	0.7912
$k = 2$	49	25	<i>identity</i>	0.7992	0.7809
			<i>tanh</i>	0.7952	0.7289
			<i>relu</i>	0.7898	0.7751
			$ z ^2$	0.7778	0.6887
			$ z $	0.7716	0.7911
$k = 4$	49	25	<i>identity</i>	0.7636	0.7854
			<i>tanh</i>	0.7796	0.4550
			<i>relu</i>	0.7658	0.7676
			$ z ^2$	0.5823	0.0289
			$ z $	0.7809	0.7573
$k = 8$	48	24	<i>identity</i>	0.7760	0.7663
			<i>tanh</i>	0.7449	0.1799
			<i>relu</i>	0.7182	0.7484
			$ z ^2$	0.0053	0.0053
			$ z $	0.7449	0.7302

Table 7: Test accuracy of a multi-layer perceptron consisting of  $k + 2$  dense layers with an overall budget of 500,000 real-valued parameters on CIFAR-10 image classification (experiment 2). The last layer consists of  $c = 10$  neurons. Selected best of 10 runs. Each run was trained for 100 epochs.

Hidden layers $k$	Units		Activation function $\varphi$	MNIST	
	$m_R$	$m_C$		R	C
$k = 0$	162	81	<i>identity</i>	0.4335	0.1006
			<i>tanh</i>	0.5032	0.1676
			<i>relu</i>	0.5007	0.4554
			$ z ^2$	0.5179	0.1006
			$ z $	0.5263	0.2381
$k = 2$	148	77	<i>identity</i>	0.4069	0.1000
			<i>tanh</i>	0.5205	0.1673
			<i>relu</i>	0.5269	0.4963
			$ z ^2$	0.1395	0.1273
			$ z $	0.5315	0.1000
$k = 4$	138	74	<i>identity</i>	0.4052	0.1000
			<i>tanh</i>	0.5218	0.1475
			<i>relu</i>	0.5203	0.4975
			$ z ^2$	0.1065	0.1010
			$ z $	0.5234	0.1000
$k = 8$	123	69	<i>identity</i>	0.4050	0.1003
			<i>tanh</i>	0.5162	0.1396
			<i>relu</i>	0.5088	0.4926
			$ z ^2$	0.1000	0.1000
			$ z $	0.5194	0.1000

Table 8: Test accuracy of a multi-layer perceptron consisting of  $k + 2$  dense layers with an overall budget of 500,000 real-valued parameters on CIFAR-100 image classification (experiment 2). The last layer consists of  $c = 100$  neurons. Selected best of 10 runs. Each run was trained for 100 epochs.

Hidden layers $k$	Units		Activation function $\varphi$	CIFAR-100	
	$m_R$	$m_C$		R	C
$k = 0$	158	79	<i>identity</i>	0.2807	0.0314
			<i>tanh</i>	0.2308	0.0193
			<i>relu</i>	0.2153	0.1935
			$ z ^2$	0.2364	0.0124
			$ z $	0.2439	0.0279
$k = 2$	144	75	<i>identity</i>	0.1723	0.0100
			<i>tanh</i>	0.2440	0.0203
			<i>relu</i>	0.2481	0.2224
			$ z ^2$	0.0155	0.0151
			$ z $	0.2453	0.0100
$k = 4$	135	72	<i>identity</i>	0.1727	0.0100
			<i>tanh</i>	0.2397	0.0150
			<i>relu</i>	0.2381	0.2147
			$ z ^2$	0.0122	0.0100
			$ z $	0.2390	0.0100
$k = 8$	121	67	<i>identity</i>	0.1706	0.0100
			<i>tanh</i>	0.2209	0.0164
			<i>relu</i>	0.2167	0.2027
			$ z ^2$	0.0100	0.0100
			$ z $	0.2191	0.0100

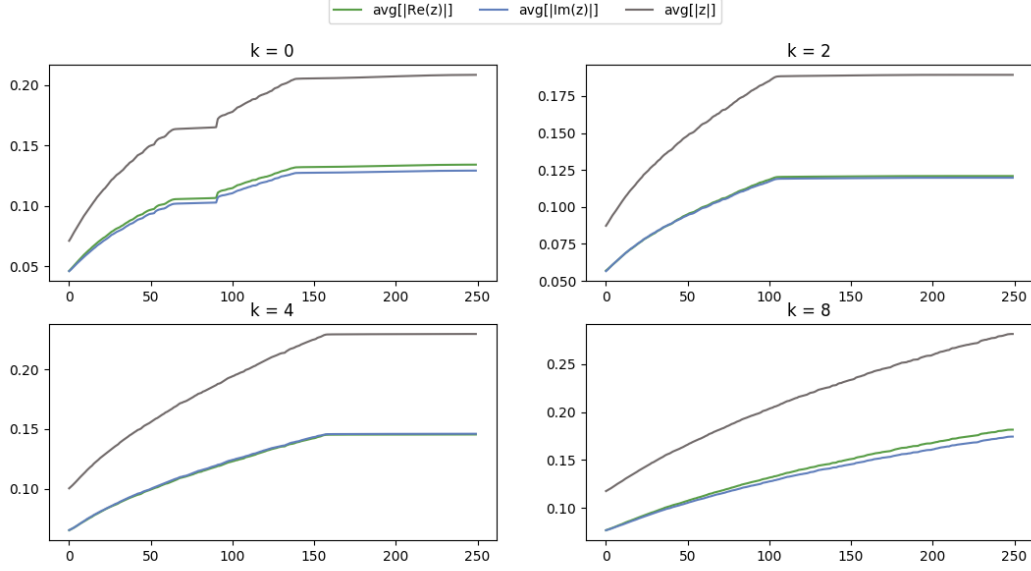


Figure 2: Average absolutes of real  $|Re(z)|$ , imaginary parts  $|Im(z)|$  and average complex magnitude  $|z|$  of all weights  $W_i$  over training epochs of the MNIST classification task.

if initialised more often. We found a test accuracy of 0.7748 in complex-valued case in comparison to 0.7978 in the real-valued case (Table 2).

## 9 Discussion

For many applications that involve data that has an interpretation on the complex plane (e.g. signals) complex-valued neural networks have already shown that they are superior [15]. All selected tasks in our work use real-valued input data. We observe that complex-valued neural networks do not perform as well as expected for the selected tasks and real-valued architectures outperform their complex version. This finding seems counter-intuitive at first, since every real value is just a special case of a complex numbers with a zero imaginary part. Solving a real-valued problem with a complex-valued model allows the model greater degree of freedom to approximate the function. The question why complex-valued models are inferior to real models for the classification of real-valued data arises.

In further examination of the training process we observed that the imaginary parts of the complex weights always follow the real parts of the weights. We show this behaviour representatively with two tasks in Figure 2 and Figure 3.

We further investigate this behaviour with synthetic classification tasks in consideration of the information flow within a MLP. We create two synthetic classification tasks. Random complex data points  $x \in \mathbb{C}^n$  are to be classified using a complex-valued MLP according to the quadrant of its sum  $\sum_{i=0}^n x_i$  or if it is close to the origin (Figure 4). Real data points  $x \in \mathbb{R}^n$  follow. This is equivalent to a projection of the complex data points to  $\mathbb{R}$  (Figure 5). We classify  $n = 10000$  complex resp. real input with Gaussian noise  $\sigma = 0.2$  and  $d = 25$  dimensions using a complex-valued MLP with  $k = 2$  hidden layers and with each  $m = 64$  units per layer. Again, we observe that the weight initialisation is a significant problem. However, the complex model can reliably approximate the underlying complex or real functions achieving training accuracy of 0.981, test accuracy of 0.908. We observe that over the training process using complex input develops differently than for real input. Using complex input the real and imaginary parts develop independently and then reach convergence (Figure 6). Training with real-valued synthetic data the magnitudes of imaginary parts follow the real parts very closely (Figure 7). Independently of the initialisation, the real parts converge a few epochs before the imaginary part (between 3 and 5 epochs). We also tested non-linear and linear complex-valued regression approximating a real and a complex function. The imaginary part of the weights in a regression problem converges to zero or does not change if initialised with zero.

To explain the behaviour of the imaginary weights consider the computations of a complex-valued MLP combining two real weight matrices for a complex MLP in Figure 8.

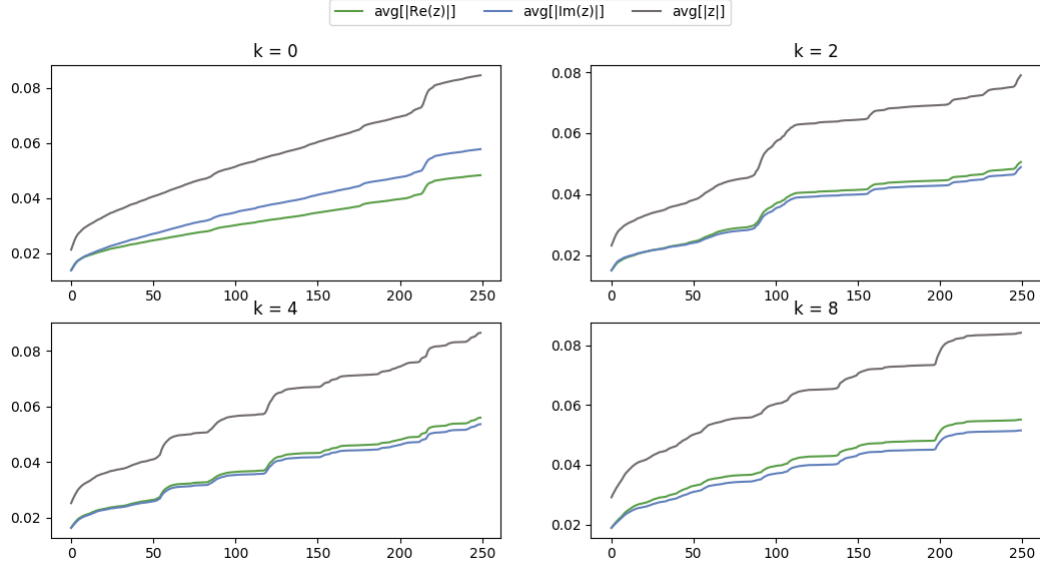


Figure 3: Average absolutes of real  $|\text{Re}(z)|$ , imaginary parts  $|\text{Im}(z)|$  and average complex magnitude  $|z|$  of all weights  $W_i$  over training epochs of the Reuters classification task.

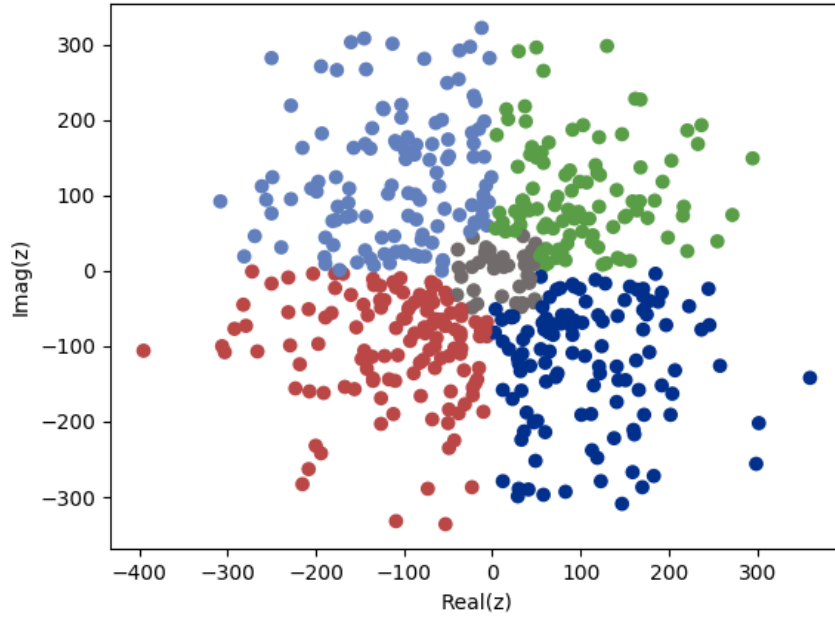


Figure 4: Example of synthetic data's class distribution. The class is determined by the location of the complex input vector's sum. Colours indicate different classes.

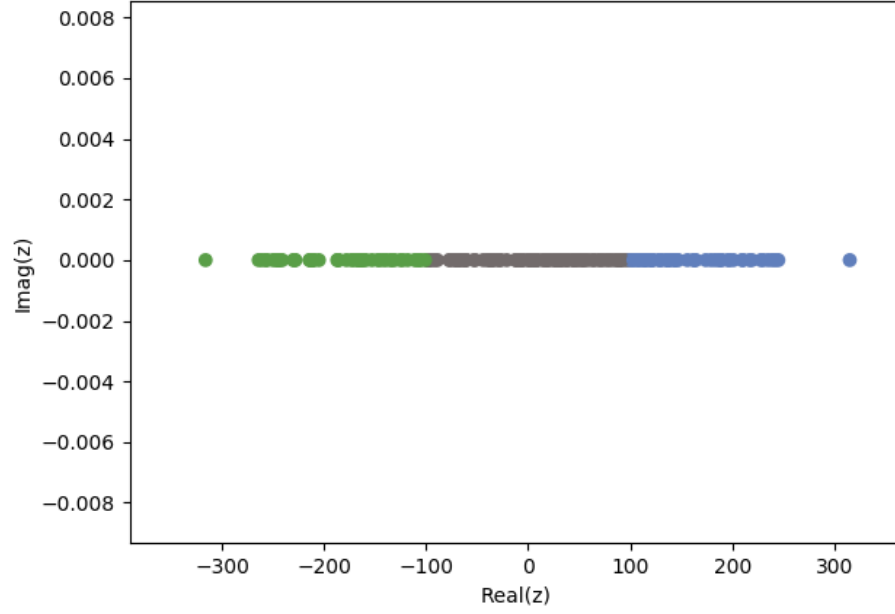


Figure 5: Example of synthetic data’s class distribution. The class is determined by the location of the real input vector’s sum. Colours indicate different classes.

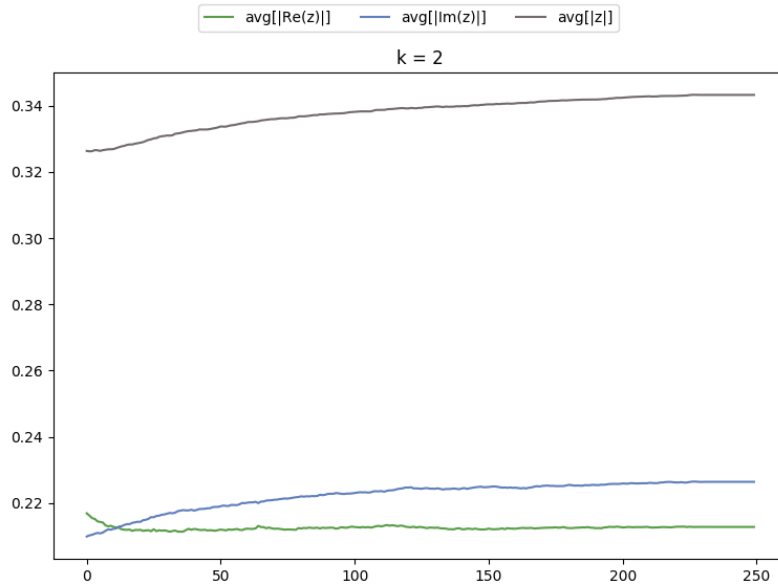


Figure 6: Example of independent learning behaviour of real and imaginary parts in the classification of synthetic complex-valued data. The real and imaginary parts change independently over the training. The exact trajectory of the graph depends on the weight initialisation.

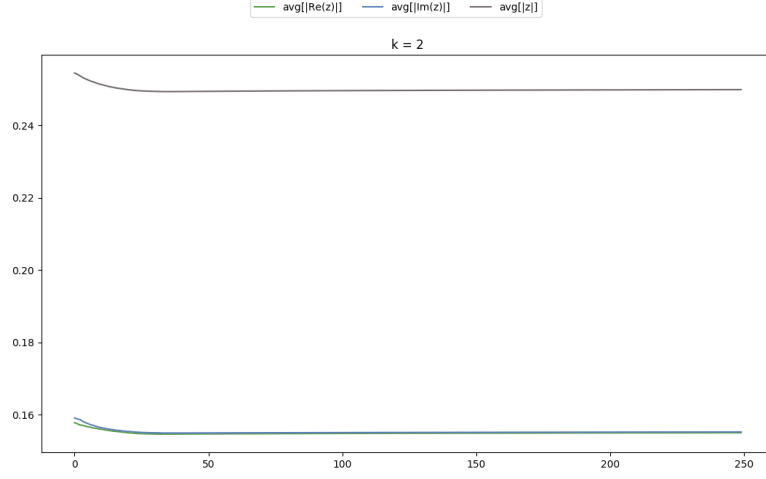


Figure 7: Example of dependent learning behaviour of the complex weights in the classification of synthetic real-valued data. The imaginary part follows the real part of the weight with every epoch. The exact trajectory of the graph depends on the weight initialisation.

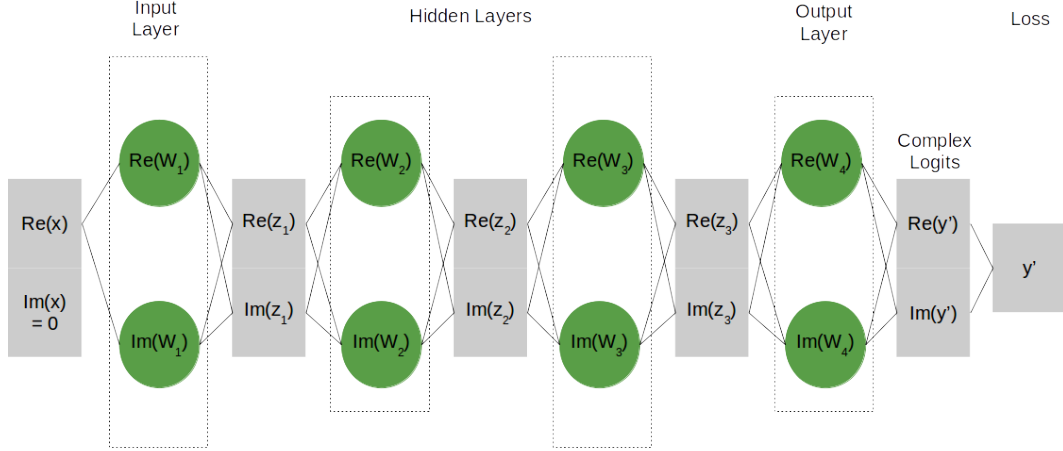


Figure 8: Interaction of real parts  $Re(W_i)$ , imaginary parts  $Im(W_i)$  with the real-valued input  $x$  and complex output  $z_i$  of the  $i$ -th layer of an MLP with  $k = 2$  layers.

We see that the real and imaginary parts of the weights act identically on the input in order to reach a classification. The classification is thus the average of two identical classifications. If in the training phase the averaged absolute values of weight’s imaginary part follow those of the real parts, the imaginary part of the input is either distributed exactly the same way as its real part, or the considered task simply does not benefit from using complex-valued hypothesis.

Moreover, we observed that complex-valued neural networks are much more sensitive towards their initialisation than real-valued neural networks. This sensitivity increases with the size of the network. The weight initialisation suggested by Trabelsi et al. [26] can reduce this problem, but does not solve it. This initialisation method is a complex generalisation of the variance scaled initialisation by Glorot et. al. [11]. Other possible initialisations include the use of the random search algorithm (RSA) [31]. This requires significantly more computation. We eventually tried to mitigate the problem by running each experiment multiple times with different random initialisation. The initialisation of complex weights, however, is still a significant and unsolved problem and requires further investigation.

The unboundedness of the activation functions can cause numerical instability of the learning process. It can lead to a failed learning process (e.g. the gradient is practically infinite). If the learning process hits such a point in the function (e.g. a singularity), it is difficult to recover the training. It is avoidable by constraining a function, normalising weights or gradients. With increasing depth and structural complexity these options may be impractical due to their computational cost. Alternatively, this can also be prevented in the design stage by choosing a bounded and entirely complex-differentiable activation function. Finding such a function is difficult. Another possibility is to avoid the problem in practice by applying separate bounded activation functions (the same or different real functions) help overcome this problem. The rectifier linear unit is one of these functions. While not entirely real-differentiable we found the training process is much more stable and the performance improved. Despite the mathematical difficulties due the differentiability, we can practically transfer a lot of insights from the real to the complex domain.

In summary real-valued models pose an upper performance limit for real-valued tasks when compared to complex-valued models of similar capacity, because the real and imaginary parts act identically on the input. An investigation of the information and gradient flow, can help to identify tasks that benefit from complex-valued neural networks. In consideration of existing literature and our finding we recommend that complex neural networks should be used for classification tasks if the data is naturally in the complex domain, or can be meaningfully moved to the complex plane. The network should reflect the interaction of real and imaginary parts of weights with the input data. If the structure is ignored, the model may not be able to utilise the greater degree of freedom. It will most likely also require more initialisations and computational time due to the more complicated training process.

## 10 Conclusion

This work considers a comparison between complex- and real-valued multi-layer perceptrons in benchmark classification tasks. We found that complex-valued MLPs perform similar or worse for classification of real-valued data, even if the complex-valued model allows larger degrees of freedom. We recommend the use of complex numbers in neural networks if a) the input data has a natural mapping to complex numbers, b) the noise in the input data is distributed on the complex plane or c) complex-valued embeddings can be learned from real-valued data. We can identify tasks that would benefit from it by comparing the training behaviour (e.g. by the average absolute values) of real and imaginary weights. If the imaginary part does not follow the real parts general behaviour across epochs, the task benefits from assuming a complex hypothesis.

Other aspects to consider for the model design are activation functions, weight initialisation strategies and the trade-off between performance, model size and computational costs. In our work, the best performing activation function is the component-wise application of the rectifier linear unit. We transfer many real and complex activation function by applying them separately on the two real parts, using Wirtinger Calculus or a gradient-based approach combined with strategies to avoid certain points. The initialisation described by Trabelsi et al. [26] can help to reduce the initialisation problem, but further investigation is required. Similar to many other architectures, the introduction of complex numbers as parameters is also a decision to trade-off between the task-specific performance, the size of the model (i.e. the number of real-value parameters) and the computational cost.

## Acknowledgements

Nils Mönning was supported by the EPSRC via a Doctoral Training Grant (DTG) Studentship. Suresh Manandhar was supported by EPSRC grant EP/I037512/1, A Unified Model of Compositional & Distributional Semantics: Theory and Application.



## References

- [1] Igor Aizenberg. 'multiple-valued threshold logic' translated by claudio moraga. 1977.
- [2] Igor Aizenberg. *Complex-Valued Neural Networks with Multi-Valued Neurons*. Springer Publishing Company, Incorporated, 2016. ISBN 3662506319, 9783662506318.
- [3] Igor Aizenberg and Claudio Moraga. Multilayer feedforward neural network based on multi-valued neurons (mlmvn) and a backpropagation learning algorithm. *Soft Computing*, 11(2):169–183, Jan 2007. ISSN 1433-7479.
- [4] Naum N. Aizenberg and Igor N. Aizenberg. Cnn based on multi-valued neuron as a model of associative memory for grey scale images. In *CNNA '92 Proceedings Second International Workshop on Cellular Neural Networks and Their Applications*, pages 36–41, Oct 1992. doi: 10.1109/CNNA.1992.274330.
- [5] Martín Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. *CoRR*, abs/1511.06464, 2015.
- [6] 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.
- [7] 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.
- [8] Thomas L. Clarke. Generalization of neural networks to the complex plane. In *1990 IJCNN International Joint Conference on Neural Networks*, pages 435–440 vol.2, June 1990.
- [9] Lukas Drude, Bhiksha Raj, and Reinhold Häb-Umbach. On the appropriateness of complex-valued neural networks for speech enhancement. In *INTERSPEECH*, 2016.
- [10] 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.
- [11] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterton, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.
- [12] S.L. Goh, M. Chen, D.H. Popovi, K. Aihara, D. Obradovic, and D.P. Mandic. Complex-valued forecasting of wind profile. *Renewable Energy*, 31(11):1733–1750, 2006. doi: <https://doi.org/10.1016/j.renene.2005.07.006>.
- [13] Nitzan Guberman. On complex valued convolutional neural networks. *CoRR*, abs/1602.09046, 2016.
- [14] R. Haensch and O. Hellwich. Complex-valued convolutional neural networks for object detection in polsar data. In *8th European Conference on Synthetic Aperture Radar*, pages 1–4, June 2010.
- [15] A. Hirose. Complex-valued neural networks: The merits and their origins. In *2009 International Joint Conference on Neural Networks*, pages 1237–1244, June 2009.
- [16] Akira Hirose. *Complex-Valued Neural Networks: Theories and Applications (Series on Innovative Intelligence, 5)*. World Scientific Press, 2004. ISBN 9812384642.
- [17] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.
- [18] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [19] 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, pages 1649–1652 vol.2, Oct 1993.
- [20] 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.

- [21] Yüksel Özbay. A new approach to detection of ecg arrhythmias: Complex discrete wavelet transform based complex valued artificial neural network. *Journal of Medical Systems*, 33(6):435, Sep 2008. doi: 10.1007/s10916-008-9205-1.
- [22] Dong-Chul Park and Tae-Kyun Jung Jeong. Complex-bilinear recurrent neural network for equalization of a digital satellite channel. *IEEE Transactions on Neural Networks*, 13(3):711–725, May 2002. ISSN 1045-9227. doi: 10.1109/TNN.2002.1000135.
- [23] C. A. Popa. Complex-valued convolutional neural networks for real-valued image classification. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 816–822, May 2017. doi: 10.1109/IJCNN.2017.7965936.
- [24] Andy M. Sarroff, Victor Shepardson, and Michael A. Casey. Learning representations using complex-valued nets. *CoRR*, abs/1511.06351, 2015.
- [25] A. B. Suksmono and A. Hirose. Adaptive noise reduction of insar images based on a complex-valued mrf model and its application to phase unwrapping problem. *IEEE Transactions on Geoscience and Remote Sensing*, 40(3):699–709, March 2002. ISSN 0196-2892. doi: 10.1109/TGRS.2002.1000329.
- [26] Chiheb Trabelsi, Sandeep Subramanian, Negar Rostamzadeh, Soroush Mehri, Dmitriy Serdyuk, João Felipe Santos, Yoshua Bengio, and Christopher Pal. Deep complex networks. 2017.
- [27] 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, pages 2071–2080, 2016.
- [28] 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.
- [29] 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, editors, *Advances in Neural Information Processing Systems 29*, pages 4880–4888. Curran Associates, Inc., 2016.
- [30] Z. Zhang, H. Wang, F. Xu, and Y. Q. 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, Dec 2017. ISSN 0196-2892. doi: 10.1109/TGRS.2017.2743222.
- [31] Hans-Georg Zimmermann, Alexey Minin, and Victoria Kuserbaeva. Comparison of the complex valued and real valued neural networks trained with gradient descent and random search algorithms. In *ESANN*, 2011.