Complex-valued Neural Networks with Non-parametric Activation Functions

Simone Scardapane, Steven Van Vaerenbergh, *Senior Member, IEEE,* Amir Hussain, *Senior Member, IEEE,* and Aurelio Uncini, *Member, IEEE*

***Abstract*—Complex-valued neural networks (CVNNs) are a powerful modeling tool for domains where data can be naturally interpreted in terms of complex numbers. However, several analytical properties of the complex domain (such as holomorphicity) make the design of CVNNs a more challenging task than their real counterpart. In this paper, we consider the problem of flexible activation functions (AFs) in the complex domain, i.e., AFs endowed with sufficient degrees of freedom to adapt their shape given the training data. While this problem has received considerable attention in the real case, a very limited literature exists for CVNNs, where most activation functions are generally developed in a split fashion (i.e., by considering the real and imaginary parts of the activation separately) or with simple phase-amplitude techniques. Leveraging over the recently proposed kernel activation functions (KAFs), and related advances in the design of complex-valued kernels, we propose the first fully complex, non-parametric activation function for CVNNs, which is based on a kernel expansion with a fixed dictionary that can be implemented efficiently on vectorized hardware. Several experiments on common use cases, including prediction and channel equalization, validate our proposal when compared to real-valued neural networks and CVNNs with fixed activation functions.**

***Index Terms*—Neural networks, Activation functions, Kernel methods, Complex domain.**

I. Introduction

OVER the last years, machine learning techniques have obtained impressive results in a wide range of fields, especially when dealing with supervised problems [1]–[3]. The majority of these applications has focused on the case of *real-valued* data: as an example, most of the deep learning frameworks currently used today can only work with floating point (or integer) numbers. Several applicative domains of interest, however, exhibit data that can be more naturally modeled using *complex-valued* algebra, from image processing to time-series prediction, bioinformatics, and robotics' control (see [4], [5] for a variety of examples). While complex data can immediately be transformed to a real domain by considering the real and imaginary components separately, the resulting loss of phase information gives rise to algorithms that are generally less efficient (or expressive) than alternative methods able to work directly in the complex domain, as evidenced by a large body of literature [6]. Due to this, many learning algorithms have been extended to deal with complex data, including linear adaptive filters [5], [7], kernel methods [8]–[10], component analysis [11], and neural networks (NNs) [12]–[18]. We consider this last class of algorithms here.

Despite the apparent similarity between the real and complex domains, working directly in the latter is challenging because of several non-intuitive analytical properties of the complex algebra. Most notably, almost all cost functions involved in the training of complex models require non-analytic (also known as non-holomorphic [8]) functions, so that standard complex derivatives cannot be used in the definition of the optimization algorithms. This is why several algorithms defined before the last decade considered optimizing the real and imaginary components separately, resulting in a more cumbersome notation which somehow hindered their development [19]. More recently, this problem has been solved by the adoption of the so-called CR-calculus (or Wirtinger's calculus), allowing to define proper complex derivatives even for non-analytic functions [20], [21], by considering explicitly their dependence on both their arguments and their complex conjugates. We describe CR-calculus more in depth in Section II.

When dealing with neural networks, another challenging task concerns the design of a proper activation function in the complex domain. In the real-valued case, the use of the rectified linear unit (ReLU) has been instrumental in the development of truly deep networks [22], [23], and has spun a wave of further research in the topic, see [24], [25] for very recent examples. In the complex case, Liouville's theorem asserts that the only complex function which is analytic and bounded at the same time is a constant one. Due to the preference for bounded activation functions before the introduction of the ReLU, many authors in the past preferred bounded functions to analytic ones, most notably in a split organization, wherein the real and independent parts of the activations are processed separately [26], or in a phase-amplitude configuration, in which the nonlinearity is applied only to the magnitude component, while the phase component is preserved [12]. Even extending the ReLU function to the complex domain has been shown to be non-trivial, and several authors have proposed different variations [14], [16].

In this paper, we consider the problem of *adapting* activation functions in the complex domain. For real-valued NNs, there is a large body of literature pointing to the fact that

S. Scardapane and A. Uncini are with the Department of Information Engineering, Electronics and Telecommunications (DIET), Sapienza University of Rome, Via Eudossiana 18, 00184 Rome, Italy. Emails: simone.scardapane, aurelio.uncini@uniroma1.it

S. Van Vaerenbergh is with the Department of Communications Engineering, University of Cantabria, Av. los Castros s/n, 39005 Santander, Cantabria, Spain. Email: steven.vanvaerenbergh@unican.es.

A. Hussain is with the Division of Computing Science & Maths, School of Natural Sciences, University of Stirling, Stirling FK9 4LA, Scotland, UK. Email: ahu@cs.stir.ac.uk.

endowing activation functions with several degrees of freedom can improve the accuracy of the trained networks, ease the flow of the back-propagated gradient, or vastly simplify the design of the network. In the simplest case, we can consider parametric functions having only a few (generally less than three) parameters per neuron, such as the parametric ReLU [27] or the S-shaped ReLU [28]. More generally, we can think of *non-parametric* activation functions, that can adapt to potentially any shape in a purely data-driven fashion, with a flexibility that can be controlled by the user, and to which standard regularization techniques can be applied. In the real-valued case, much research has been devoted to the topic, including the design of Maxout networks [29], adaptive piecewise linear (APL) units [30], spline functions [31], and the recently proposed kernel activation functions (KAFs) [32]. When dealing with complex-valued NNs (CVNNs), however, only a handful of works have considered adapting the activation functions, and only in the simplified parametric case [17], or when working in a split configuration [11]. In this sense, how to design activation functions that can adapt to the training data while remaining simple to implement remains an open question.

Contributions of the paper

We introduce a new family of non-parametric activation functions in the complex domain, building upon the idea of KAFs [32]. In particular, by building on recent works on complex-valued reproducing kernel Hilbert spaces [8] (RKHSs), we propose the first adaptable activation function directly defined in the complex domain. All the functions we introduce can leverage highly vectorized CPU/GPU libraries for matrix multiplication.

The basic idea of KAFs, which were defined in [32] only in the real-valued case, is to exploit a kernel expansion at every neuron, in which the elements of the kernel dictionary are fixed beforehand, while the mixing coefficients are adapted through standard optimization techniques. Here, we propose two different techniques to apply the idea of KAFs in the context of CVNNs. In the first case, we use a split combination where the real and the imaginary components are processed by two independent KAFs sharing the same dictionary. In the second case, based on the complex-valued RKHS theory, we are able to redefine the KAF *directly in the complex domain*, also describing several choices for the kernel function. We show via multiple experimental comparisons that CVNNs endowed with complex-valued KAFs can outperform both real-valued NNs and CVNNs having only fixed or parametric activation functions.

Organization of the paper

In Section II we introduce the basic theoretical elements underpinning optimization in a complex domain and CVNNs. Then, in Section III we summarize research on designing activation functions for CVNNs. The two proposed complex KAFs are given in Section IV (split KAF) and Section V (fully complex KAF). We briefly discuss implementation aspects of CVNNs in Section VI. Finally, we provide an experimental evaluation in Section VII before concluding in Section VIII.

Notation

We denote vectors using boldface lowercase letters, e.g., $\mathbf{a}$; matrices are denoted by boldface uppercase letters, e.g., $\mathbf{A}$. All vectors are assumed to be column vectors. A complex number $z \in \mathbb{C}$ is represented as $z = a + ib$, where $a = \Re\{z\}$ and $b = \Im\{z\}$ are, respectively, the real part and the imaginary part of the number, and $i = \sqrt{-1}$. Sometimes, we also use z_r and z_i to denote the real and imaginary parts of z for simplicity. Magnitude and phase of a complex number are given by $|z|$ and $\phi(z)$ respectively. $z^* = a - ib$ denotes the complex conjugate of z. Other notation is introduced in the text when appropriate.

II. Preliminaries

A. Complex algebra and CR-calculus

We start by introducing the basic theoretical concepts required to define a complex-valued function and to optimize it. We consider scalar functions first, and discuss the multivariate extension later on. Any complex-valued function $f : \mathbb{C} \to \mathbb{C}$ can be written as:

$$f(z) = u(a, b) + iv(a, b)\,, \tag{1}$$

where $u(\cdot, \cdot)$ and $v(\cdot, \cdot)$ are real-valued functions in two arguments. The function f is said to be *real-differentiable* if the partial derivatives of u and v with respect to a and b are defined. Additionally, the function is called *analytic* (or holomorphic) if it satisfies the Cauchy-Riemann conditions:

$$\frac{\partial u(a,b)}{\partial a} = \frac{\partial v(a,b)}{\partial b} \quad \text{and} \quad \frac{\partial v(a,b)}{\partial a} = -\frac{\partial u(a,b)}{\partial b}\,. \tag{2}$$

Only analytic functions admit a complex derivative in the standard sense, but most functions used in practice for CVNNs do not satisfy (2) (such as functions with real-valued outputs for which $v(a, b) = 0$ everywhere). In this case, CR-calculus [21] provides a theoretical framework to handle non-analytic functions directly in the complex domain without the need to switch back and forth between definitions in the complex domain and gradients' computations in the real one.

The main idea of CR-calculus is to consider f explicitly as a function of both z and its complex conjugate $z^* = a - ib$, which we denote as $f(z, z^*)$. If f is real-differentiable, then it is also analytic with respect to z when keeping z^* constant and vice versa. Thus, we can define a pair of (complex) derivatives as follows [20], [21]:

$$\text{R-derivative} \triangleq \left.\frac{\partial f(z,z^*)}{\partial z}\right|_{z^*=\text{const}} = \frac{1}{2}\left(\frac{\partial f}{\partial a} - i\frac{\partial f}{\partial b}\right)\,, \tag{3}$$

$$\text{R*-derivative} \triangleq \left.\frac{\partial f(z,z^*)}{\partial z^*}\right|_{z=\text{const}} = \frac{1}{2}\left(\frac{\partial f}{\partial a} + i\frac{\partial f}{\partial b}\right)\,. \tag{4}$$

Everything extends to multivariate functions $f : \mathbb{C}^n \to \mathbb{C}$ of a complex vector $\mathbf{z} \in \mathbb{C}^n$ by defining the cogradient and conjugate cogradient operators:

$$\nabla_{\mathbf{z}} = \left(\frac{\partial}{\partial z_1}, \ldots, \frac{\partial}{\partial z_n}\right)^T\,, \tag{5}$$

$$\nabla_{\mathbf{z}^*} = \left(\frac{\partial}{\partial z_1^*}, \ldots, \frac{\partial}{\partial z_n^*}\right)^T\,. \tag{6}$$

Then, a necessary and sufficient condition for $\mathbf{z}_0$ to be a minimum of f is either $\nabla_{\mathbf{z}_0} f(\mathbf{z}_0, \mathbf{z}_0^*) = 0$ or $\nabla_{\mathbf{z}_0^*} f(\mathbf{z}_0, \mathbf{z}_0^*) = 0$ [20]. CR-calculus inherits most of the standard properties of the real derivatives, including the chain rule and the differential rule, see [21]. For the important case where the output of the function is real-valued (as is the case for the loss function when optimizing CVNNs) we have the additional property:

$$\left(\nabla_{\mathbf{z}} f(\mathbf{z}, \mathbf{z}^*)\right)^* = \nabla_{\mathbf{z}^*} f(\mathbf{z}, \mathbf{z}^*) \,. \tag{7}$$

Combined with the Taylor expansion of the function, an immediate corollary of this property is that the direction of steepest ascent of f in the point $\mathbf{z}$ is given by the *conjugate* cogradient operator evaluated in that point [21]. Up to a multiplicative constant term, this result coincides with taking the steepest ascent direction with respect to the real derivatives, allowing for a straightforward implementation in most optimization libraries.

B. Complex-valued neural networks

We now turn our attention to the approximation of multivariate complex-valued functions. A generic CVNN is composed by stacking L layers via the alternation of linear and nonlinear operations. In particular, the l-th layer is described by the following equation:

$$\mathbf{h}_l = g\left(\mathbf{W}_l \mathbf{h}_{l-1} + \mathbf{b}_l\right) \,, \tag{8}$$

where $\mathbf{h}_{l-1} \in \mathbb{C}^{N_{l-1}}$ is the N_{l-1}-dimensional input to the layer, $\mathbf{W}_l \in \mathbb{C}^{N_l \times N_{l-1}}$ and $\mathbf{b}_l \in \mathbb{C}^{N_l}$ are adaptable weight matrices, and $g(\cdot)$ is a (complex-valued) activation function applied element-wise, which will be discussed more in depth later on. By definition, $\mathbf{x} = \mathbf{h}_0$ denotes the input to the network, while $\hat{y} = h_L$ denotes the final output, which we assume one-dimensional for simplicity. Some results on the approximation properties of this model are given in [13], while [17] describes techniques to initialize the adaptable linear weights in the complex domain.

Given I input/output pairs $\mathcal{S} = \{\mathbf{x}_n, y_n\}_{n=1}^I$, we train the CVNN by minimizing a cost function given by:

$$J(\mathbf{w}) = \sum_{n=1}^{I} l(y_n, \hat{y}_n) \,, \tag{9}$$

where $\mathbf{w} \in \mathbb{C}^Q$ collects all the adaptable weights of the network and $l(\cdot, \cdot)$ is a loss function, such as the squared loss:

$$l(y, \hat{y}) = |y - \hat{y}|^2 = (y - \hat{y})(y - \hat{y})^* \,. \tag{10}$$

Following the results described in the previous section, a basic steepest descent approach to optimize (9) is given by the following update equation at the t-th iteration:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \mu \nabla_{\mathbf{w}^*} J(\mathbf{w}, \mathbf{w}^*) \,, \tag{11}$$

where $\mu \in \mathbb{R}$ is the learning rate. More in general, we can use noisy versions of the gradient given by sampling a mini-batch of elements, or accelerate the optimization process by adapting most of the state-of-the-art techniques used for real-valued neural networks [33]. We can also apply some techniques that are specific to the complex domain. For example [34], inspired by the theory of widely linear adaptive filters, augments the input to the CVNN with its complex conjugate $\mathbf{x}^*$. Additional improvements can be obtained by replacing the real-valued μ with a complex-valued learning rate [35], which can speed up convergence in some scenarios.

III. Complex-valued activation functions

As we stated in the introduction, choosing a proper activation function in (8) is more challenging than in the real case because of Liouville's theorem, stating that the only complex-valued functions that are bounded and analytic everywhere are constants. So in practice, one must choose between boundedness and analyticity. Before the introduction of the ReLU activation [22], almost all activation functions in the real case were bounded. Consequently, initial approaches to design CVNNs always preferred non-analytic functions in order to preserve boundedness, most commonly by applying real-valued activation functions separately to the real and imaginary parts [26]:

$$g(z) = g_R(\Re\{z\}) + i g_R(\Im\{z\}) \,, \tag{12}$$

where z is a generic input to the activation function in (8), and $g_R(\cdot)$ is some real-valued activation function, e.g., sigmoid. This is called a *split activation function*. As a representative example, the magnitude and phase of the split-$\tanh$ when varying the activation are given in Fig. 1. Early proponents of this approach can be found in [36] and [19].

Another common class of non-analytic activation functions are the phase-amplitude (PA) functions popularized by [12], [37]:

$$g(z) = \frac{z}{c + |z|/r} \,, \tag{13}$$

$$g(z) = \tanh\left\{\frac{|z|}{m}\right\} \exp\{i\phi(z)\} \,, \tag{14}$$

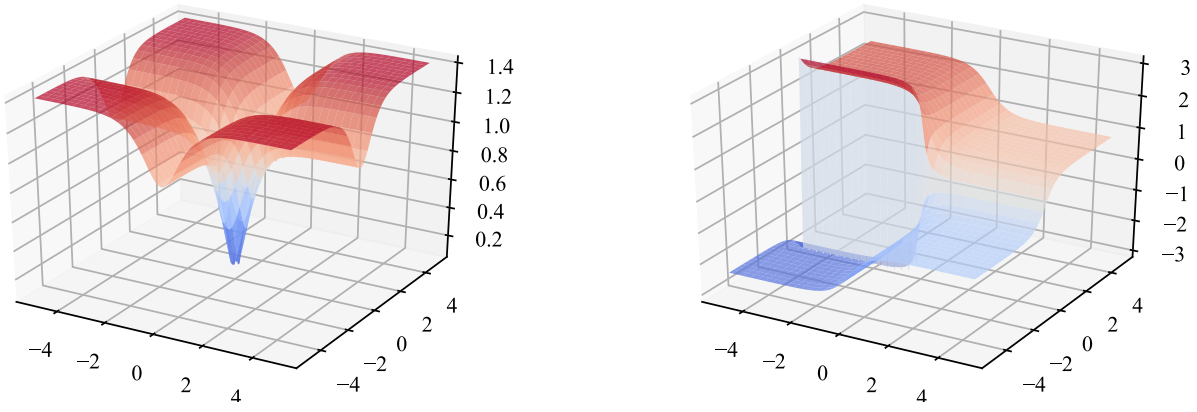
where $\phi(z)$ is the phase of z, while c, r and m are positive constant which in most cases are set equal to 1. PA functions can be seen as the natural generalization of real-valued squashing functions such as the sigmoid, because the output $g(z)$ has bounded magnitude but preserves the phase of z.

A third alternative is to use fully-complex activation functions that are analytic and bounded almost everywhere, at the cost of introducing a set of singular points. Among all possible transcendental functions, it is common to consider the complex-valued extension of the hyperbolic tangent, defined as [13]:

$$g(z) = \tanh\{z\} = \frac{\exp\{z\} - \exp\{-z\}}{\exp\{z\} + \exp\{-z\}} \,, \tag{15}$$

possessing periodic singular points at the imaginary points $i(0.5 + n)\pi$, with $n \in \mathbb{N}$. However, careful scaling of the inputs and of the initial weights allows to avoid these singularities during training.

Finally, several authors have proposed extensions of the popular real-valued ReLU function $\text{ReLU}(s) = \max\{0, s\}$. As discussed in [17], a simple split configuration as in (12)



(a) Absolute value (b) Phase

Fig. 1. Example of split activation function having $g_R(\cdot) = \tanh(\cdot)$ in (12) processing both the real and the imaginary parts of the input. (a) Magnitude of the output. (b) Phase of the output.

results in poor performance. An improved complex-valued ReLU is designed in [16] as:

$$g(z) = \begin{cases} z & \text{if } \Re\{z\}, \Im\{z\} \geq 0, \\ 0 & \text{otherwise} \end{cases}. \tag{16}$$

Alternatively, inspired by the PA functions to maintain the phase of the activation value, [14] propose the following modReLU function:

$$g(z) = \text{ReLU}\left(|z| + b\right) \exp\left\{i\phi(z)\right\}, \tag{17}$$

where b is an adaptable parameter defining a radius along which the output of the function is 0. Another extension, the complex cardioid, is advanced in [38]:

$$g(z) = \frac{1}{2}\Big(1 + \cos\left\{\phi(z)\right\}\Big) z, \tag{18}$$

maintaining phase information while attenuating the magnitude based on the phase itself. For real-valued inputs, (18) reduces to the ReLU.

Note that in all cases these proposed activation functions are fixed or endowed with a very small degree of flexibility (as in (17)). In the following sections we describe a principled technique to design non-parametric activation functions for use in CVNNs.

IV. Split Kernel Activation Functions

Our first proposal is a split function as in (12), where non-parametric (real-valued) functions for $g_R(\cdot)$ are used in place of fixed ones. Specifically, we consider the kernel activation function (KAF) proposed in [32], which will also serve as a base for the fully complex-valued proposal of the following section. Here, we introduce the basic elements of the KAF, and we refer to the original paper [32] for a fuller exposition.

The basic idea of a KAF is to model each activation function as a one-dimensional kernel model, where the kernel elements are chosen in a proper way to obtain an efficient backpropagation step. Consider the generic activation function $g_R(s)$, where s denotes either the real or the imaginary part of z as in (12). To obtain a flexible shape, we can model a linear predictor on a high-dimensional feature space $\Phi(s)$ of the activation. However, this process becomes infeasible for a large number of feature transformations, and cannot handle infinite-dimensional feature spaces. For feature maps associated to a reproducing kernel Hilbert space $\mathcal{H}$ with kernel $\kappa(\cdot,\cdot)$, we can write an equivalent linear model by exploiting the representer theorem as:

$$g_R(s) = \sum_{n=1}^{D} \alpha_n \kappa\left(s, d_n\right), \tag{19}$$

where $\{\alpha_n\}_{n=1}^D$ are the mixing coefficients and $\{d_n\}_{n=1}^D$ make up the so-called dictionary of the kernel expansion [39], [40]. Remember that a function $\kappa(\cdot,\cdot)$ is a valid kernel function if it respects the positive semi-definiteness property, i.e., for any possible choice of $\{\alpha_n\}_{n=1}^D$ and $\{d_n\}_{n=1}^D$ in (19):

$$\sum_{n=1}^{D}\sum_{m=1}^{D} \alpha_n \alpha_m \kappa\left(d_n, d_m\right) \geq 0. \tag{20}$$

In the context of a neural network, the dictionary elements cannot be selected *a priori* because they would change at every step of the optimization algorithm depending on the distribution of the activation values. Instead, we exploit the fact that we are working with one-dimensional kernels to fix the elements beforehand, and only adapt the mixing coefficients in the optimization step. In particular, we select the elements $d_1, \ldots, d_D$ by sampling D values over the x-axis, uniformly around zero. In this way, the value D becomes a hyper-parameter controlling the flexibility of the approach: for larger D we obtain a more flexible method at the cost of a larger number of adaptable parameters. In general, since the function is only a small component of a much larger neural network, values in the range $D \in [10, 20]$ are sufficient for most applications. As the number of parameters per neuron can potentially grow without bound depending on the choice of D, we refer to such activation functions as non-parametric.

The same dictionary is shared across the entire neural

network, but with two different sets of mixing coefficients for the real and imaginary parts of each neuron. Due to this, an efficient implementation of the proposed split-KAF is straightforward. In particular, consider the vector $\mathbf{z}$ containing the N_l (complex) activations of a layer following the linear operations in (8). We build the matrix $\mathbf{K}_R \in \mathbb{R}^{N_l \times D}$ by computing all the kernel values between the real part of the activations and the elements of the dictionary (and similarly for $\mathbf{K}_I$ using the imaginary parts), and we compute the final output of the layer as:

$$\mathbf{h}_l = (\mathbf{A}_R \odot \mathbf{K}_R)\,\mathbf{1} + i\,(\mathbf{A}_I \odot \mathbf{K}_I)\,\mathbf{1}\,, \tag{21}$$

where $\odot$ represents element-wise product (Hadamard product), $\mathbf{A}_R, \mathbf{A}_I \in \mathbb{R}^{N_l \times D}$ are matrices collecting row-wise all the mixing coefficients for the real and imaginary components of the layer, and $\mathbf{1} \in \mathbb{R}^D$ is a vector of ones. For handling batches of elements (or convolutive layers), we only need to slightly modify (21) by adding additional trailing dimensions.

For all our experiments, we consider the 1D Gaussian kernel defined as:

$$\kappa(s, d_n) = \exp\left\{-\gamma\,(s - d_n)^2\right\}\,, \tag{22}$$

where $\gamma \in \mathbb{R}$ is the inverse of the kernel bandwidth. In the proposed KAF scheme, the values of the dictionary are chosen according to a grid, and as such the optimal bandwidth parameter depends uniquely on the grid resolution. In particular, the following rule-of-thumb was proposed in [32] and it is used in our experiments:

$$\gamma = \frac{1}{6\Delta^2}\,, \tag{23}$$

where Δ is the distance between the grid points. In order to provide an additional degree of freedom to our method, we also optimize a single γ per layer via back-propagation after initializing it following (23).

V. Fully-complex kernel activation functions

While most of the literature on kernel methods in machine learning has focused on the real-valued case, it is well known that the original mathematical treatment originated in the complex-valued domain [41]. In the context of the kernel filtering literature, techniques to build complex-valued algorithms by separating the real and the imaginary components (as in the previous section) are called complexification methods [8]. However, recently several authors have advocated for the direct use of (pure) complex-valued kernels leveraging the complex-valued treatment of RKHSs for a variety of fields, as surveyed in the introduction.

From a theoretical standpoint, defining complex RKHSs and kernels is relatively straightforward. As an example, a one-dimensional complex-function $\kappa_{\mathbb{C}} : \mathbb{C} \times \mathbb{C} \to \mathbb{C}$ is positive semi-definite if and only if:

$$\sum_{n=1}^{D}\sum_{m=1}^{D} \alpha_n^* \alpha_m \kappa\,(d_n, d_m) \geq 0\,, \forall \alpha_n, \alpha_m, d_n, d_m \in \mathbb{C}\,, \tag{24}$$

where all values are now defined in the complex-domain. Any PSD function is then a valid kernel function. Based on this, in this paper we also propose a fully-complex, non-parametric

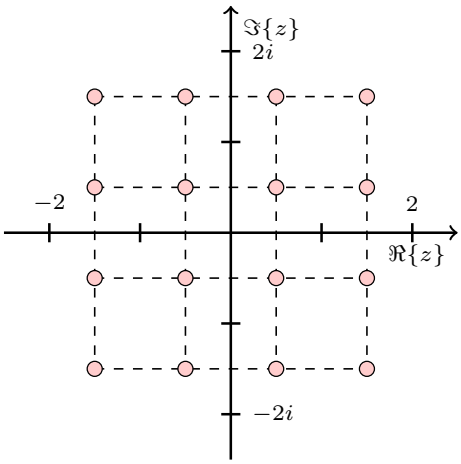


Fig. 2. A visual example of sampling the dictionary for the complex-valued KAF, in the complex plane, for $D = 4$ in the range $[-1.5, 1.5]$.

KAF by defining (19) directly in the complex domain, without the need for split functions:

$$g(z) = \sum_{n=1}^{D}\sum_{m=1}^{D} \alpha_{n,m}\kappa_{\mathbb{C}}\,(z, d_n + id_m)\,, \tag{25}$$

where the mixing coefficients $\{\alpha_{n,m}\}_{n,m=1}^{D}$ are now defined as complex numbers. Note that, in order for the dictionary to provide a dense sampling of the space of complex numbers, we now consider D^2 fixed elements arranged over a regular grid, an example of which is depicted in Fig. 2. Due to this, we now have D^2 adaptable mixing coefficients per neuron, as opposed to $2D$ in the split case. We counter-balance this by selecting a drastically smaller D (see the experimental section).

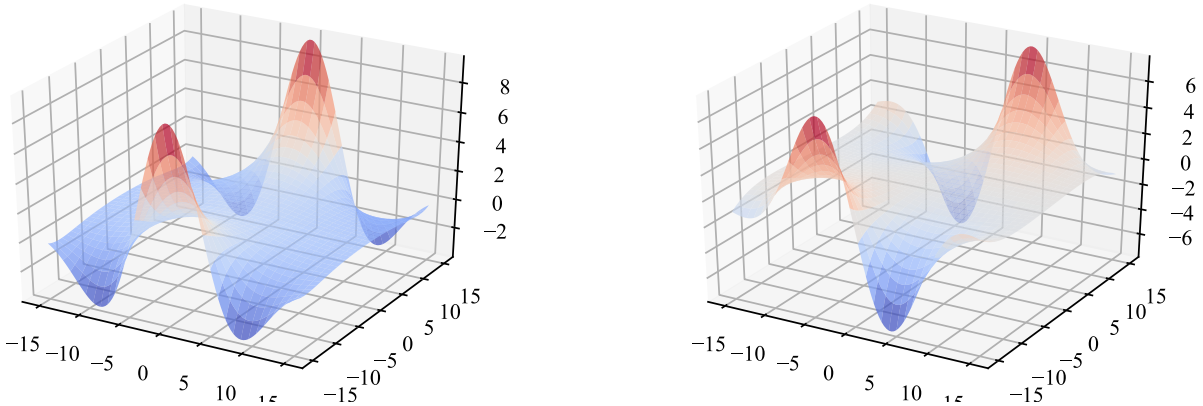
An immediate complex-valued extension of the Gaussian kernel in (22) is given by:

$$\kappa_{\mathbb{C}}(z, d) = \exp\left\{-\gamma\,(z - d^*)^2\right\}\,, \tag{26}$$

where in our experiments the bandwidth hyper-parameter γ is selected using the same rule-of-thumb as before and then adapted layer-wise. A complete analysis of the feature space associated to (26) is given in [42]. In order to gain some informal understanding, we can write the kernel explicitly in terms of the real and imaginary components of its arguments:

$$\begin{aligned}\kappa_{\mathbb{C}}(z, d) = &\exp\left\{-\gamma|z_r - d_r|^2\right\}\exp\left\{\gamma|z_i + d_i|^2\right\} \\ &\cdot\Big(\cos\left\{2\gamma\,(z_r - d_r)\,(z_i + d_i)\right\} \\ &-i\sin\left\{2\gamma\,(z_r - d_r)\,(z_i + d_i)\right\}\Big)\,.\end{aligned} \tag{27}$$

By analyzing the previous expression, we see that the complex-valued Gaussian kernel has several properties which are counter-intuitive if one is used to work with its real-valued restriction. First of all, (26) cannot be interpreted as a standard similarity measure, because it depends on its arguments only via $(z_r - d_r)$ and $(z_i + d_i)$. For the same reasons, the kernel is not stationary, and it has an additional oscillatory behavior. We refer to Fig. 3 (or to [10, Section IV-A]) for an illustration of the kernel when fixing the second argument.



(a) Absolute value (b) Phase

Fig. 3. Example of Gaussian complex kernel in (26) with $d = 0 + i0$ and $\gamma = 0.01$. Notice the scale of the axes (more details are provided in the text). (a) Real part of the output. (b) Imaginary part of the output.

For these reasons, another extension of the Gaussian kernel to the complex domain is given in [8], where the authors propose to build a whole family of complex-valued kernels starting from any real-valued one $\kappa_{\mathbb{R}}$ as follows:

$$\kappa_{\mathbb{C}}\left(z, d\right) = \kappa_{\mathbb{R}}\left(z_r, d_r\right) + \kappa_{\mathbb{R}}\left(z_i, d_i\right) + i\left(\kappa_{\mathbb{R}}\left(z_r, d_i\right) - \kappa_{\mathbb{R}}\left(z_i, d_r\right)\right) . \tag{28}$$

The new complex-valued kernel is called an *independent* kernel. By plugging the real-valued Gaussian kernel (22) in the previous expression, we obtain a complex-valued expression that can still be interpreted as a similarity measure between the two points.

Note that several alternative kernels are also possible, many of which are specific to the complex-valued case, a prominent example being the Szego kernel [8]:

$$\kappa_{\mathbb{C}}(z, d) = \frac{1}{\left(1 - zd^*\right)^2} . \tag{29}$$

VI. Notes on Implementation

In the previous sections we have described two complete functional models of complex-valued neural networks based on non-parametric activation functions. Nevertheless, the design of the architecture and the training of a CVNN in a practical implementation involve several additional procedures. We now briefly discuss these procedures and comment on how they must be adapted for the complex-valued case w.r.t. real-valued neural networks (RVNNs).

a) Hyperparameter optimization: The optimization of real-valued hyperparameters (such as hidden layer size) in CVNNs is equivalent to RVNN practices. As such, it can be dealt with by standard hyperparameter optimization methods including grid search, randomized search [43], and Bayesian optimization [44]. The optimization of complex-valued hyperparameters (such a complex step size) has not been explored yet in the literature to the best of our knowledge, and it is beyond the scope of this work. Before considering a fully complex hyperparameter optimization, however, a simple workaround would consist in splitting the complex-valued hyperparameters in real and imaginary part, similar to the strategy followed in Section IV.

b) Weight initialization: Standard initialization procedures for the linear weights in RVNNs have been described in [45] and [27]. Recently, an extension of these procedures to the complex-valued case was proposed [17], which we adopt in the experiments of Section VII. In particular, we initialize the complex weights of the l-th layer by drawing their magnitudes from $\mathcal{N}\left(0, \frac{2}{N_l}\right)$ and their phases from $\mathcal{U}(-\pi, \pi)$, where N_l is the number of neurons in this layer.

c) Deep networks: The construction of deep complex-valued architectures requires overcoming several practical challenges, similar to those that appear in real-valued networks. In the experiments of Section VII, we only consider shallow networks that contain at most three hidden layers. Nevertheless, the interested reader may refer to the discussion on deep complex-valued networks in [17], which proposes among others a complex-valued batch normalization technique.

VII. Experimental Evaluation

In this section, we experimentally evaluate the proposed activation functions on several benchmark problems, including channel identification in Section VII-A, wind prediction in Section VII-B, and multi-class classification in the complex domain in Section VII-C. In all cases, we linearly preprocess the real and the imaginary components of the input features to lie in the $[-1, +1]$ range. We regularize all parameters with respect to their squared absolute value (which is equivalent to standard ℓ_2 regularization applied on the real and imaginary components separately), but we exclude the bias terms and the window parameter in (17). We select the strength of the regularization term and the size of the networks based on previous literature or on a cross-validation procedure, as described below. For optimization, we use a simple complex-valued extension of the Adagrad algorithm, which computes a per-parameter learning rate weighted by the squared magnitude of the gradients themselves. For each iteration, we construct

a mini-batch by randomly sampling 40 elements from the entire training dataset. All algorithms have been implemented in Python using the Autograd library [46].

A. Experiment 1 - Channel Identification

Our first experiment is a standard benchmark in the complex-valued literature, i.e. a channel identification task [47]. The input to the channel is generated as:

$$s_n = \left(\sqrt{1-\rho^2}X_n + i\rho Y_n\right), \tag{30}$$

where X_n and Y_n are Gaussian random variables, and the parameter ρ determines the circularity[1] of the signal. For $\rho = \frac{\sqrt{2}}{2}$ the input is circular, while for ρ approaching 0 or 1 the signal is highly non-circular. The output of the channel is computed by first applying a linear filtering operation:

$$t_n = \sum_{k=1}^{5} h(k) s_{n-k+1}, \tag{31}$$

where:

$$h(k) = 0.432\left(1 + \cos\left\{\frac{2\pi(k-3)}{5}\right\} - i\left(1 + \cos\left\{\frac{2\pi(k-3)}{10}\right\}\right)\right), \tag{32}$$

for $k = 1, \ldots, 5$. Then, the output of the linear filter goes through a memoryless nonlinearity:

$$r_n = t_n + (0.15 - i0.1)\, t_n^2, \tag{33}$$

and finally it is corrupted by adding white Gaussian noise in order to get the final signal $\tilde{r}_n$. The variance of the noise is selected to obtain a signal-to-noise ratio (SNR) of about 13 dB. The input to the neural network is an embedding of channel inputs:

$$\mathbf{x} = [s_{n-L+1}, s_{n-L+2}, \ldots, s_n]^T, \tag{34}$$

with $L = 5$, and the network is trained to output $\tilde{r}_n$. We generate 2000 samples of the channel, and we randomly keep 15% for testing, averaging over 15 different generations of the dataset. We compare the following algorithms:

- **LIN**: a standard linear filter [5] with complex-valued coefficients.
- **2R-NN**: a real-valued neural network taking as input the real and imaginary parts separately. For the activation functions in the hidden layers, we consider either a standard $\tanh$ or ReLUs.
- **C-NN**: complex-valued neural networks with fixed activation functions, including a split-$\tanh$, a split-ReLU, the AMP function in (13), or the complex ReLU in (16).
- **ModReLU-NN**: CVNN with adaptable activation functions with ModReLU neurons as in (17). In this case, the coefficients of the neurons are all initialized at 0.1 and later adapted.
- **Maxout**: a CVNN where we use the non-parametric Maxout activation function [29] in a split configuration.
- **Proposed KAF-NN**: CVNN with the split-KAF proposed in Section IV. We empirically select $D = 20$ elements in the dictionary sampled uniformly in $[-2, +2]$.
- **Proposed C-KAF-NN**: CVNN with the fully complex KAF proposed in Section V. In this case, we test either the complex Gaussian kernel (26), or the independent kernel with the real Gaussian kernel as base. We empirically select $D = 8$.

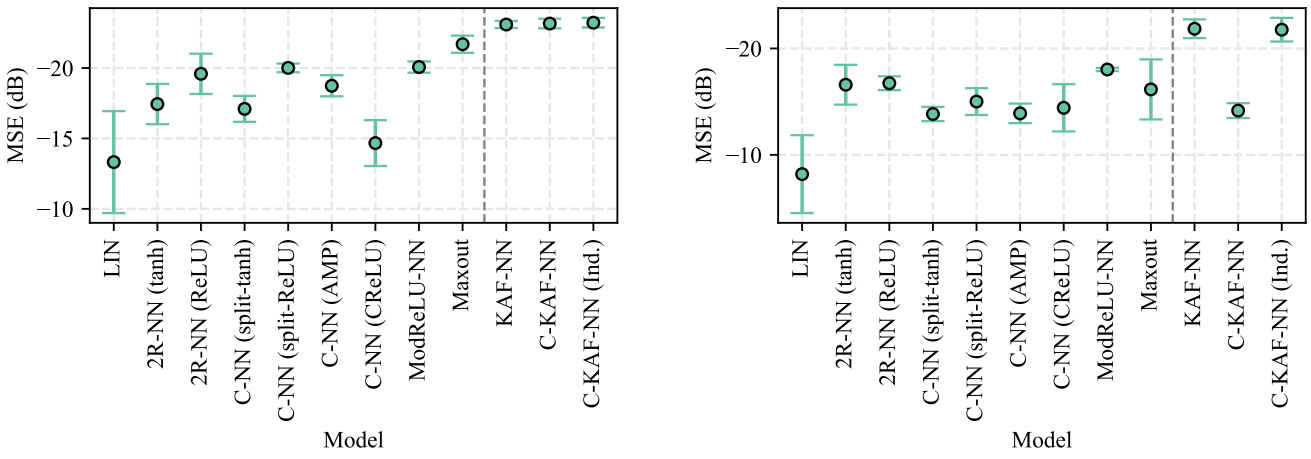
All algorithms are trained by minimizing the mean-squared error in (10) on random mini-batches of 40 elements. Following [34], in this scenario we consider one hidden layer with 10 neurons (as more layers are not found to provide significant improvements in performance). The size of the regularization factor is empirically selected as 10^{-4}. Results in terms of mean squared error (MSE) expressed in dBs are given in Table 4, by considering either $\rho = \frac{\sqrt{2}}{2}$ (circular input signal) or the more challenging scenario $\rho = 0.95$ (non-circular signal).

As expected, results are generally lower for the non-circular case, proportionally so for techniques that are not able to exploit the geometry of non-circular complex signals, such as non-widely linear models and real-valued neural networks. However, the proposed KAF-NN and C-KAF-NN are able to consistently out-perform all other methods in both scenarios in a stable fashion. Note that this difference in performance cannot be overcome by increasing the size of the other networks, thus pointing to the importance of adapting the activation functions also in the complex case. Interestingly, the complex Gaussian kernel in (26) results in a poor performance, similarly to the split-Maxout, which is solved by using the independent one.

B. Experiment 2 - Wind prediction

For the second experiment, we consider a real-world dataset for a task of wind prediction [48]. The dataset consists of 5000 hourly samples of wind intensity collected along two different axes (north axis and east axis). The dataset is provided in three settings of wind regime, namely 'low', 'medium', and 'high', from which we select the highest, being the most challenging one. In order to construct a complex-valued signal, the two samples for each hour are considered as the real and the imaginary components of a single complex number (for more motivation on the use of complex-valued information when dealing with wind forecasting, see [18], [48]–[51]). A snapshot of the absolute value and phase of the resulting signal is shown in Fig. 5 for the initial 500 samples. We consider the task of predicting both components of the wind for an 8-hour-ahead horizon, starting from an embedding of the last 10 hours of measurements. We select neural networks with 2 hidden layers (as more hidden layers are not found to provide gain in performance), and we optimize both the number of neurons and the regularization factor on a held-out validation set. We test the datasets on the last 500 components of the time-series, in terms of the R^2 coefficient of determination:

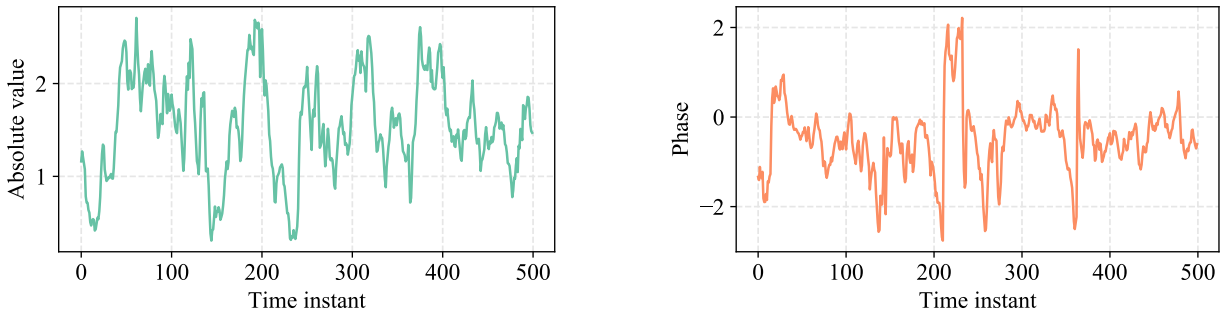
[1]A random variable Z is circular if Z and $Z\exp\{i\psi\}$ have the same probability distribution for any angle ψ. Roughly speaking, non-circular signals are harder to predict, requiring the use of widely linear techniques when using standard linear filters [8].



(a) Circular signal ($\rho = \frac{\sqrt{2}}{2}$)

(b) Non-circular signal ($\rho = 0.95$)

Fig. 4. Results for the first experiment, expressed in terms of MSE (dB). (a) Circular input signal. (b) Non-circular input signal. With a dashed line we divide the results of the proposed models.



(a) Absolute value

(b) Phase

Fig. 5. A plot of the complex-valued wind profile for the initial 500 samples of the wind time-series. (a) Absolute value of the signal. (b) Phase of the signal.

$$R^2 = 1 - \frac{\sum_{n=1}^{500} |y_n - \hat{y}_n|^2}{\sum_{n=1}^{500} |y_n - \bar{y}|^2}, \tag{35}$$

where y_n is the true value, $\hat{y}_n$ is the predicted value, and $\bar{y}$ is the mean of the true values computed from the test set. Positive values of R^2 denotes a prediction which is better than chance, with values approaching 1 for an almost-perfect prediction.

Results for the experiment are reported in Table I. We can see that, also in this scenario, the two best results are obtained by the proposed split-KAF and complex KAF neurons, significantly outperforming the other models.

C. Experiment 3: complex-valued multi-class classification

We conclude our experimental evaluation by testing the proposed algorithms on four multi-class image classification problems expressed in the complex domain. Following [47], we build each task by applying a two-dimensional fast Fourier transform (FFT) to the images in the well-known MNIST dataset,[2] comprising 60000 28×28 black-and-white images of handwritten digits split into ten classes. We then rank the coefficients of the FFT in terms of significance (by considering their mean absolute value), and keep only the 100 most significant coefficients as input to the models. In order to provide a wider comparison, we also apply the same procedure to three additional datasets:

- **Fashion MNIST** (F-MNIST) [52]: a variant of MNIST concerning images of clothing items, with the same dimensionality.
- **Extended MNIST** (EMNIST) [53]: a set of extensions of MNIST, from which we consider the 'Digits' one, comprising 240 thousand images of handwritten digits.
- **Latin OCR** [54]: an OCR problem concerning handwritten Latin digits segmented from real manuscripts of the Vatican secret archives. There are approximately 12 thousands characters belonging to 23 separate classes.

[2]http://yann.lecun.com/exdb/mnist/

TABLE I
RESULTS (MEAN AND STANDARD DEVIATION FOR THE COEFFICIENT OF DETERMINATION R^2) IN THE WIND PREDICTION TASK. BEST RESULT IS HIGHLIGHTED IN BOLD, SECOND-BEST RESULT IN UNDERLINED.

Model		$\mathbf{R^2}$
Linear	Linear	0.361 ± 0.0227
Real-valued NNs	2R-NN (tanh)	0.424 ± 0.015
	2R-NN (ReLU)	0.435 ± 0.016
CVNN	C-NN (split-tanh)	0.426 ± 0.015
	C-NN (split-ReLU)	0.438 ± 0.016
	C-NN (AMP)	0.431 ± 0.014
	C-NN (CReLU)	0.181 ± 0.106
	ModReLU-NN	0.438 ± 0.015
Proposed CVNN	KAF-NN	$\mathbf{0.444 \pm 0.015}$
	C-KAF-NN	0.424 ± 0.016
	C-KAF-NN (Ind.)	$\underline{0.442 \pm 0.016}$

We compare a real-valued NN taking the real and the imaginary components of the coefficients as separate inputs, a CVNN with modReLU activation functions, and two CVNNs employing spit-KAFs and fully-complex KAFs with independent kernels. All networks have a softmax activation function in their output layer. For the CVNNs, we use the following variation to handle the complex valued activations $\mathbf{h}$:

$$\text{softmax}_n(\mathbf{h}) = \frac{\exp\left\{\Re\left\{h_n\right\}^2 + \Im\left\{h_n\right\}^2\right\}}{\sum_{t=1}^{C} \exp\left\{\Re\left\{h_t\right\}^2 + \Im\left\{h_t\right\}^2\right\}}, \quad (36)$$

where $\mathbf{h} \in \mathbb{C}^C$, and $C = 10$ for our problem. All networks are then trained by minimizing the classical regularized cross-entropy formulation with the same optimizer as the last sections. We consider networks with three hidden layers having 100 neurons each, whose regularization term is optimized via cross-validation separately. We also apply an early stopping procedure (with the standard splits from each dataset), stopping whenever accuracy is not improving for 1000 iterations of optimization. Results on the test sets are provided in Table II.

We see that working in the complex domain results in significantly better performance when compared to working in the real domain. We show a representative evolution of the loss function for MNIST in Fig. 6, where we highlight the first 10000 iterations for readability.

VIII. CONCLUSIVE REMARKS

In this paper, we considered the problem of adapting activation functions in a complex-valued neural network (CVNN). To this end, we proposed two different non-parametric models that extend the recently introduced kernel activation function (KAF) to the complex-valued case. The first model is a split configuration, where the real and the imaginary components of the activation are processed independently by two separate KAFs. In the second model, we directly redefine the KAF in the complex domain with the use of fully-complex kernels. We showed that CVNNs with adaptable functions can outperform neural networks with fixed functions in different benchmark problems, including channel identification, wind prediction, and multi-class classification. For the fully-complex KAF, the independent kernel generally outperforms a naive complex Gaussian kernel without introducing significantly more complexity.

Due to the space constraints, in this paper we have focused only on a selected number of experimental comparisons, with a limited number of complex kernels. In order to overcome these limitations, multiple future works are possible, most notably by leveraging over recent advances in the field of real-valued kernels [55] and complex-valued kernel regression and classification. One example is the use of pseudo-kernels [10] to handle more efficiently the non-circularity in the signals propagated through the network. Additionally, we plan on comparing with more datasets, e.g., [10], and analyze the temporal convergence rate comparison of CVNN with real-valued approaches. More generally, it would be interesting to extend other classes of non-parametric, real-valued activation functions (such as Maxout networks [29] or adaptive piecewise linear units [30]) to the complex domain, or adapt the proposed complex KAFs to other types of NNs, such as convolutive architectures [1], [56].

ACKNOWLEDGMENTS

The work of Simone Scardapane was supported in part by Italian MIUR, "*Progetti di Ricerca di Rilevante Interesse Nazionale*", GAUChO project, under Grant 2015YPXH4W_004. The work of Steven Van Vaerenbergh was supported by the Ministerio de Economía, Industria y Competitividad (MINECO) of Spain under grant TEC2014-57402-JIN (PRISMA). Amir Hussain was supported by the UK Engineering and Physical Science Research Council (EPSRC) grant no. EP/M026981/1.

The authors also thank the anonymous reviewers for their help in improving the manuscript.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[2] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, Feb 2018.
[3] K. Zheng, W. Q. Yan, and P. Nand, "Video dynamics detection using deep neural networks," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2017.
[4] A. Hirose, *Complex-valued neural networks: theories and applications*. World Scientific, 2003, vol. 5.
[5] P. J. Schreier and L. L. Scharf, *Statistical signal processing of complex-valued data: the theory of improper and noncircular signals*. Cambridge University Press, 2010.
[6] D. P. Mandic, S. Javidi, G. Souretis, and V. S. Goh, "Why a complex valued solution for a real domain problem," in *2007 IEEE Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2007, pp. 384–389.
[7] B. Fisher and N. Bershad, "The complex LMS adaptive algorithm–transient weight mean and covariance with applications to the ALE," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 31, no. 1, pp. 34–44, 1983.

TABLE II
TEST ACCURACY (MEAN AND STANDARD DEVIATION) IN THE COMPLEX-VALUED IMAGE CLASSIFICATION TASKS. THE BEST RESULTS FOR EACH DATASET ARE HIGHLIGHTED IN BOLD.

Model	MNIST	F-MNIST	E-MNIST	Latin OCR
Real-valued NN	92.39 ± 0.10	71.08 ± 0.45	92.78 ± 1.25	39.01 ± 3.42
CVNN (ModReLU)	95.92 ± 0.18	77.27 ± 0.61	95.53 ± 0.98	70.42 ± 0.93
CVNN (KAF-NN)	$\mathbf{97.21 \pm 0.34}$	79.73 ± 0.32	97.74 ± 0.84	$\mathbf{72.27 \pm 1.21}$
CVNN (C-KAF-NN, Ind)	97.18 ± 0.27	$\mathbf{81.94 \pm 0.91}$	$\mathbf{98.11 \pm 2.04}$	71.79 ± 2.40

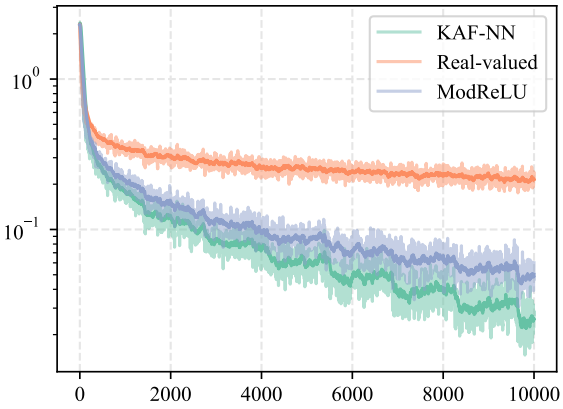


Fig. 6. Loss function evolution for three of the algorithms on the complex-valued MNIST task (detail of the first 10000 iterations). The evolution of the C-KAF-NN algorithms was very similar to split-KAF and for clarity we have left out their curves.

[8] P. Bouboulis and S. Theodoridis, "Extension of Wirtinger's calculus to reproducing kernel Hilbert spaces and the complex kernel LMS," *IEEE Transactions on Signal Processing*, vol. 59, no. 3, pp. 964–978, 2011.

[9] F. A. Tobar, A. Kuh, and D. P. Mandic, "A novel augmented complex valued kernel LMS," in *2012 IEEE 7th Sensor Array and Multichannel Signal Processing Workshop (SAM)*. IEEE, 2012, pp. 473–476.

[10] R. Boloix-Tortosa, J. J. Murillo-Fuentes, I. Santos, and F. Pérez-Cruz, "Widely linear complex-valued kernel methods for regression," *IEEE Transactions on Signal Processing*, vol. 65, no. 19, pp. 5240–5248, 2017.

[11] M. Scarpiniti, D. Vigliano, R. Parisi, and A. Uncini, "Generalized splitting functions for blind separation of complex signals," *Neurocomputing*, vol. 71, no. 10, pp. 2245–2270, 2008.

[12] G. M. Georgiou and C. Koutsougeras, "Complex domain backpropagation," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 39, no. 5, pp. 330–334, 1992.

[13] T. Kim and T. Adalı, "Approximation by fully complex multilayer perceptrons," *Neural Computation*, vol. 15, no. 7, pp. 1641–1666, 2003.

[14] M. Arjovsky, A. Shah, and Y. Bengio, "Unitary evolution recurrent neural networks," in *33rd International Conference on Machine Learning (ICML)*, 2016, pp. 1120–1128.

[15] I. Danihelka, G. Wayne, B. Uria, N. Kalchbrenner, and A. Graves, "Associative long short-term memory," *arXiv preprint arXiv:1602.03032*, 2016.

[16] N. Guberman, "On complex valued convolutional neural networks," *arXiv preprint arXiv:1602.09046*, 2016.

[17] C. Trabelsi, O. Bilaniuk, D. Serdyuk, S. Subramanian, J. F. Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, and C. J. Pal, "Deep complex networks," *arXiv preprint arXiv:1705.09792*, 2017.

[18] A. S. Shiva, M. Gogate, N. Howard, B. Graham, and A. Hussain, "Complex-valued computational model of hippocampal CA3 recurrent collaterals," in *2017 IEEE 16th International Conference on Cognitive Informatics Cognitive Computing (ICCI*CC)*, July 2017, pp. 161–166.

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

[20] D. Brandwood, "A complex gradient operator and its application in adaptive array theory," in *IEE Proceedings F - Communications, Radar and Signal Processing*, vol. 130, no. 1. IET, 1983, pp. 11–16.

[21] K. Kreutz-Delgado, "The complex gradient operator and the CR-calculus," *arXiv preprint arXiv:0906.4835*, 2009.

[22] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011, pp. 315–323.

[23] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *30th International Conference on Machine Learning (ICML)*, vol. 30, no. 1, 2013.

[24] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," *arXiv preprint arXiv:1706.02515*, 2017.

[25] P. Ramachandran, B. Zoph, and Q. V. Le, "Swish: a self-gated activation function," *arXiv preprint arXiv:1710.05941*, 2017.

[26] T. Nitta, "An extension of the back-propagation algorithm to complex numbers," *Neural Networks*, vol. 10, no. 8, pp. 1391–1415, 1997.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1026–1034.

[28] X. Jin, C. Xu, J. Feng, Y. Wei, J. Xiong, and S. Yan, "Deep learning with S-shaped rectified linear activation units," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[29] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *30th International Conference on Machine Learning (ICML)*, 2013.

[30] F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi, "Learning activation functions to improve deep neural networks," *arXiv preprint arXiv:1412.6830*, 2014.

[31] S. Scardapane, M. Scarpiniti, D. Comminiello, and A. Uncini, "Learning activation functions from data using cubic spline interpolation," *arXiv preprint arXiv:1605.05509*, 2016.

[32] S. Scardapane, S. Van Vaerenbergh, S. Totaro, and A. Uncini, "Kafnets: kernel-based non-parametric activation functions for neural networks," *arXiv preprint arXiv:1707.04035*, 2017.

[33] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *arXiv preprint arXiv:1606.04838*, 2016.

[34] D. Xu, H. Zhang, and D. P. Mandic, "Convergence analysis of an augmented algorithm for fully complex-valued neural networks," *Neural Networks*, vol. 69, pp. 44–50, 2015.

[35] H. Zhang and D. P. Mandic, "Is a complex-valued stepsize advantageous in complex-valued gradient learning algorithms?" *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 12, pp. 2730–2735, 2016.

[36] N. Benvenuto and F. Piazza, "On the complex backpropagation algorithm," *IEEE Transactions on Signal Processing*, vol. 40, no. 4, pp. 967–969, 1992.

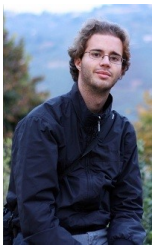
[37] A. Hirose, "Continuous complex-valued back-propagation learning," *Electronics Letters*, vol. 28, no. 20, pp. 1854–1855, 1992.

[38] P. Virtue, S. X. Yu, and M. Lustig, "Better than real: Complex-valued neural nets for MRI fingerprinting," *arXiv preprint arXiv:1707.00070*, 2017.

[39] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *The Annals of Statistics*, pp. 1171–1220, 2008.

[40] W. Liu, J. C. Principe, and S. Haykin, *Kernel adaptive filtering: a comprehensive introduction*. John Wiley & Sons, 2011, vol. 57.

[41] N. Aronszajn, "Theory of reproducing kernels," *Transactions of the American Mathematical Society*, vol. 68, no. 3, pp. 337–404, 1950.
[42] I. Steinwart, D. Hush, and C. Scovel, "An explicit description of the reproducing kernel hilbert spaces of gaussian rbf kernels," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4635–4643, 2006.
[43] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.
[44] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, 2012, pp. 2951–2959.
[45] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 9, 2010, pp. 249–256.
[46] D. Maclaurin, D. Duvenaud, and R. P. Adams, "Autograd: Effortless gradients in numpy," in *ICML 2015 AutoML Workshop*, 2015.
[47] P. Bouboulis, S. Theodoridis, C. Mavroforakis, and L. Evaggelatou-Dalla, "Complex support vector machines for regression and quaternary classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 6, pp. 1260–1274, 2015.
[48] S. Goh, M. Chen, D. Popović, K. Aihara, D. Obradovic, and D. Mandic, "Complex-valued forecasting of wind profile," *Renewable Energy*, vol. 31, no. 11, pp. 1733–1750, 2006.
[49] S. L. Goh and D. P. Mandic, "A complex-valued rtrl algorithm for recurrent neural networks," *Neural Computation*, vol. 16, no. 12, pp. 2699–2713, 2004.
[50] ——, "Nonlinear adaptive prediction of complex-valued signals by complex-valued prnn," *IEEE Transactions on Signal Processing*, vol. 53, no. 5, pp. 1827–1836, 2005.
[51] A. Kuh and D. Mandic, "Applications of complex augmented kernels to wind profile prediction," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2009, pp. 3581–3584.
[52] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.
[53] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: an extension of mnist to handwritten letters," *arXiv preprint arXiv:1702.05373*, 2017.
[54] D. Firmani, P. Merialdo, E. Nieddu, and S. Scardapane, "In codice ratio: OCR of handwritten latin documents using deep convolutional networks," in *11th International Workshop on Artificial Intelligence for Cultural Heritage (AI*CH 2017)*. CEUR Workshop Proceedings, 2017, pp. 9–16.
[55] M. Mansouri, M. N. Nounou, and H. N. Nounou, "Multiscale kernel PLS-based exponentially weighted-GLRT and its application to fault detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2017.
[56] P. Ren, W. Sun, C. Luo, and A. Hussain, "Clustering-oriented multiple convolutional neural networks for single image super-resolution," *Cognitive Computation*, pp. 1–14, 2017.



Simone Scardapane Simone Scardapane received his B.Sc. in Computer Engineering at Roma Tre university in 2009, and a M.Sc. in Artificial Intelligence and Robotics in "Sapienza" University two years later. After working one year as a software/web developer, he obtained a Ph.D. in the same university in 2016, researching mainly in the fields of distributed machine learning and adaptive audio processing. Currently, he is a post-doc fellow at "Sapienza" University, and an honorary research fellow at the CogBID laboratory at the University of Stirling (UK).

Steven Van Vaerenbergh Steven Van Vaerenbergh (M'11–SM'15) received the M.Sc. degree in electrical engineering from Ghent University, Ghent, Belgium, in 2003, and the Ph.D. degree from the University of Cantabria, Santander, Spain, in 2010. He was a Visiting Researcher with the Computational Neuroengineering Laboratory, University of Florida, Gainesville, FL, USA, in 2008. He is currently a Post-Doctoral Associate with the Department of Telecommunications Engineering, University of Cantabria. His research interests include machine learning algorithms for pattern recognition, prediction, system identification, and online machine learning.

Amir Hussain Amir Hussain obtained his BEng and PhD from the University of Strathclyde in Glasgow, Scotland, UK, in 1992 and 1997 respectively. He is currently Professor of Computing Science, and founding Director of the Cognitive Big Data Informatics (CogBID) Research Lab at the University of Stirling in Scotland, UK. He has published over 300 papers, including over a dozen books and around 120 journal papers. He is founding Editor-in-Chief of the journals Cognitive Computation (Springer Nature), and Big Data Analytics (BioMed Central/Springer Nature), and of the Springer Book Series on Socio-Affective Computing, and Cognitive Computation Trends. He is Associate Editor of the IEEE Transactions on Neural Networks and Learning Systems, the IEEE Computational Intelligence Magazine and the IEEE Transactions on Systems, Man and Cybernetics (Systems). He is a Senior Fellow of the Brain Science Foundation (USA).

Aurelio Uncini Aurelio Uncini (M'88) received the Laurea degree in Electronic Engineering from the University of Ancona, Italy, on 1983 and the Ph.D. degree in Electrical Engineering in 1994 from University of Bologna, Italy. At present time he is Full Professor with the Department of Information Engineering, Electronics and Telecommunications, where he is teaching Neural Networks, Adaptive Algorithm for Signal Processing and Digital Audio Processing, and where he is the founder and director of the 'Intelligent Signal Processing and Multimedia' (ISPAMM) group. He is a member of the Institute of Electrical and Electronics Engineers (IEEE), of the Associazione Elettrotecnica ed Elettronica Italiana (AEI), of the International Neural Networks Society (INNS) and of the Società Italiana Reti Neuroniche (SIREN).