COMPLEX-VALUED NEURAL NETWORKS FOR DATA-DRIVEN SIGNAL PROCESSING AND SIGNAL UNDERSTANDING

PREPRINT

Josiah W. Smith
josiah.radar@gmail.com

ABSTRACT

Complex-valued neural networks have emerged boasting superior modeling performance for many tasks across the signal processing, sensing, and communications arenas. However, developing complex-valued models currently demands development of basic deep learning operations, such as linear or convolution layers, as modern deep learning frameworks like PyTorch and Tensor flow do not adequately support complex-valued neural networks. This paper overviews a package built on PyTorch with the intention of implementing light-weight interfaces for common complex-valued neural network operations and architectures. Similar to natural language understanding (NLU), which as recently made tremendous leaps towards text-based intelligence, *RF Signal Understanding (RFSU)* is a promising field extending conventional signal processing algorithms using a hybrid approach of signal mechanics-based insight with data-driven modeling power. Notably, we include efficient implementations for linear, convolution, and attention modules in addition to activation functions and normalization layers such as batchnorm and layernorm. Additionally, we include efficient implementations of manifold-based complex-valued neural network layers that have shown tremendous promise but remain relatively unexplored in many research contexts. Although there is an emphasis on 1-D data tensors, due to a focus on signal processing, communications, and radar data, many of the routines are implemented for 2-D and 3-D data as well. Specifically, the proposed approach offers a useful set of tools and documentation for data-driven signal processing research and practical implementation.

Keywords Complex-Valued Neural Networks (CVNNs) · Deep Learning · Artificial Intelligence · Signal Processing · Radar · Synthetic Aperture Radar

1 Introduction

Although much work as already been done for complex-valued neural networks (CVNNs), starting back in the 1990s [1], many common operations for complex-valued tensors remain unsupported by modern deep learning frameworks like PyTorch and TensorFlow. In this paper, we introduce a lightweight wrapper built on PyTorch with two main objectives as follows

- Provide an efficient interface for complex-valued deep learning using PyTorch.
- Provide open-source implementations and documentation for complex-valued operation such activation functions, normalization layers, and attention modules to increase the research rate for CVNNs.

Fundamentally, complex-valued data have two degrees of freedom (DOF), which are commonly modeled as real and imaginary parts or magnitude and phase as

$$\mathbf{z} = \mathbf{x} + j\mathbf{y} = |\mathbf{z}|e^{j\angle\mathbf{z}}, \tag{1}$$

where $j = \sqrt{-1}$ is known as the complex unit, $\mathbf{x}$ and $\mathbf{y}$ are the real and imaginary parts of $\mathbf{z}$, and $|\mathbf{z}|$ and $\angle\mathbf{z}$ are the magnitude and phase of $\mathbf{z}$.

1.1 Existing CVNN Work

Recent surveys of CVNNs and their history can be found in [2, 3]. Starting in 2018, Trabelsi *et al.* introduced *deep complex networks* (DCN) extending many common deep learning approaches to complex-valued data [4]. As mentioned in [5], some attempts have been made by PyTorch and TensorFlow to incorporate complex-values into the core architecture; however, much work remains to be done. The authors of [5] implement a CVNN framework in TensorFlow, with impressive functionality. However, recent studies have shown a drastic decline in TensorFlow usage among engineers, while PyTorch dominates industry and academia. The code publicized as part of this work was used extensively throughout the dissertation [6] in addition to [7, 8, 9, 10, 11, 12, 13, 14, 15].

One paramount issue for CVNNs is complex-valued backpropagation and computation of complex-valued gradients. Mathematical investigations for Liouville Theorem and Wirtinger Calculus can be found in [5, 4], and are omitted here for brevity. Whereas, [5] develops Wirtinger-based proper backpropagation for TensorFlow, PyTorch natively supports complex-valued backpropagation.

1.2 Installation Notes

IMPORTANT: Prior to installation, install PyTorch to your environment using your preferred method using the compute platform (CPU/GPU) settings for your machine. PyTorch will not be automatically installed with the installation of `complextorch` and MUST be installed manually by the user.

1.2.1 Install using PyPI

`complextorch` is available on the Python Package Index (PyPI) and can be installed using the following command:

```
pip install complextorch
```

1.2.2 Install using GitHub

Useful if you want to modify the source code.

```
git clone https://github.com/josiahwsmith10/complextorch.git
```

2 Complex-Valued Layers

In this section, we overview the complex-valued layers supported by `complextorch`, which can be found at https://complextorch.readthedocs.io/en/latest/nn/modules.html. Nearly all modules introduced in this section closely follow the PyTorch to improve integration and streamline user experience (UX) for new users.

2.1 Gauss' Multiplication Trick

As pointed out in [16, 17, 18], many complex-valued operations can be more efficiently implemented by leveraging Gauss' multiplication trick. Suppose $\mathscr{L}(\cdot) = \mathscr{L}_{\mathbb{R}}(\cdot) + j\mathscr{L}_{\mathbb{I}}(\cdot)$ is a linear operator, such as multiplication or convolution, and $\mathbf{z} = \mathbf{x} + j\mathbf{y}$. Hence,

$$\mathscr{L}(\mathbf{z}) = \mathscr{L}(\mathbf{x}) + j\mathscr{L}(\mathbf{y}) = \mathscr{L}_{\mathbb{R}}(\mathbf{x}) - \mathscr{L}_{\mathbb{I}}(\mathbf{y}) + j\left(\mathscr{L}_{\mathbb{R}}(\mathbf{y}) + \mathscr{L}_{\mathbb{I}}(\mathbf{x})\right). \tag{2}$$

This is the common implementation of complex-valued operations in deep learning applications. However, it requires four computations, such as multiplications or convolutions, which can become computationally costly. Gauss' trick reduces the number of computations down to 3 as

$$\begin{aligned} t_1 &\triangleq \mathscr{L}_{\mathbb{R}}(\mathbf{x}), \\ t_2 &\triangleq \mathscr{L}_{\mathbb{I}}(\mathbf{y}), \\ t_3 &\triangleq (\mathscr{L}_{\mathbb{R}} + \mathscr{L}_{\mathbb{I}})(\mathbf{x} + \mathbf{y}), \end{aligned} \tag{3}$$

$$\mathscr{L}(\mathbf{z}) = t_1 - t_2 + j(t_3 - t_2 - t_1). \tag{4}$$

This technique is leveraged throughout `complextorch` to improve computational efficiency whenever applicable.

2.2 Complex-Valued Linear Layers

We extend linear layers, the backbone of perceptron neural networks, for CVNNs. Similar to Section 2.1, we define the complex-valued linear layer `CVLinear` as

$$H_{\texttt{CVLinear}}(\cdot) = H_{\texttt{CVLinear-}\mathbb{R}}(\cdot) + jH_{\texttt{CVLinear-}\mathbb{I}}(\cdot), \tag{5}$$

where $H_{\texttt{CVLinear-}\mathbb{R}}(\cdot)$ and $H_{\texttt{CVLinear-}\mathbb{I}}(\cdot)$ can be implemented in PyTorch as real-valued Linear layers, as detailed in Table 1. Using (4), the linear layer can be efficiently computed with the composition $H_{\texttt{CVLinear-}\mathbb{R}}(\cdot) + H_{\texttt{CVLinear-}\mathbb{I}}(\cdot)$ being a linear layer with the weights and bias of $H_{\texttt{CVLinear-}\mathbb{R}}(\cdot)$ and $H_{\texttt{CVLinear-}\mathbb{I}}(\cdot)$ summed.

`complextorch`	PyTorch
`CVLinear`	`Linear`

Table 1: PyTorch equivalent of the complex-valued linear layer.

2.3 Complex-Valued Convolution Layers

Similarly, we define a general complex-valued convolution layer `CVConv` as

$$H_{\texttt{CVConv}}(\cdot) = H_{\texttt{CVConv-}\mathbb{R}}(\cdot) + jH_{\texttt{CVConv-}\mathbb{I}}(\cdot), \tag{6}$$

where $H_{\texttt{CVConv-}\mathbb{R}}(\cdot)$ and $H_{\texttt{CVConv-}\mathbb{I}}(\cdot)$ can be implemented in PyTorch as various real-valued convolution layers, as detailed in Table 1. Using (4), the convolution layers layer can be efficiently computed with the composition $H_{\texttt{CVConv-}\mathbb{R}}(\cdot) + H_{\texttt{CVConv-}\mathbb{I}}(\cdot)$ being a convolution layer with the kernel weights and bias of $H_{\texttt{CVConv-}\mathbb{R}}(\cdot)$ and $H_{\texttt{CVConv-}\mathbb{I}}(\cdot)$ summed.

`complextorch`	PyTorch
`CVConv1d`	`Conv1d`
`CVConv2d`	`Conv2d`
`CVConv3d`	`Conv3d`
`CVConvTranpose1d`	`ConvTranpose1d`
`CVConvTranpose2d`	`ConvTranpose2d`
`CVConvTranpose3d`	`ConvTranpose3d`

Table 2: PyTorch equivalent of complex-valued convolution layers.

2.4 Complex-Valued Attention Layers

Whereas attention-based models, such as transformers, have gained significant attention for natural language processing (NLP) and image processing, their potential for implementation in complex-valued problems such as signal processing remains relatively untapped. Here, we include complex-valued variants of several attention-based techniques.

2.4.1 Complex-Valued Scaled Dot-Product Attention

The ever-popular scaled dot-product attention is the backbone of many attention-based methods [19], most notably the transformer [20, 21, 22, 23, 24, 11].

Given complex-valued *query, key, and value* tensors Q, K, V, the complex-valued scaled dot-product attention can be computed as

$$\text{Attention}(Q, K, V) = \mathcal{S}(QK^T/t)V, \tag{7}$$

where t is known as the temperature (typically $t = \sqrt{d_{attn}}$) and $\mathcal{S}$ is the softmax function.

It is important to note that unlike real-valued scaled dot-product attention, the complex-valued version detailed above must employ a complex-valued version of the soft-max function as the real-valued softmax is unsuited for complex-valued data. We implemented several complex-valued softmax function options detailed in Section 2.5.

Additionally, we include an implementation of multi-head attention, which is commonly employed in transformer and attention-based models [22].

2.4.2 Complex-Valued Efficient Channel Attention (CV-ECA)

Efficient Channel Attention (ECA) was first introduced in [25]. Here, we extend ECA to complex-valued data for CV-ECA. Following the construction of ECA, we define CV-ECA as

$$\texttt{CV-ECA} = \mathscr{M}\left(H_{\texttt{CVConv1d}}(H_{\texttt{CVAdaptiveAvgPoolNd}}(\mathbf{z}))\right) \odot \mathbf{z}, \tag{8}$$

where $\mathscr{M}(\cdot)$ is the masking function (some options are implemented in Sections 2.5 and 2.6), $H_{\texttt{CVConvNd}}(\cdot)$ is the 1-D complex-valued convolution layer defined in Section 2.3, and $H_{\texttt{CVAdaptiveAvgPoolNd}}(\cdot)$ is the complex-valued global adaptive average pooling layer for the N-D input tensor detailed in Section 2.11. It is important to note that the 1-D convolution is computed along the channel dimension of pooled data. This is the notable difference between ECA and MCA.

2.4.3 Complex-Valued Masked Channel Attention (CV-MCA)

Complex-valued masked channel attention (CV-MCA) is similar to CV-ECA but employs a slightly different implementation. Generally, the masked attention module implements the following operation

$$\texttt{CV-MCA}(\mathbf{z}) = \mathscr{M}(H_{\text{ConvUp}}(\mathscr{A}(H_{\text{ConvDown}}(\mathbf{z})))) \odot \mathbf{z}, \tag{9}$$

where $\mathscr{M}(\cdot)$ is the masking function (some options are implemented in Sections 2.5 and 2.6), $H_{\text{ConvUp}}(\cdot)$ and $H_{\text{ConvDown}}(\cdot)$ are N-D convolution layers with kernel sizes of 1 that reduce the channel dimension by a factor r, and $\mathscr{A}(\cdot)$ is the complex-valued non-linear activation layer (see Section 2.8). The implementations of complex-valued masked attention are included for 1-D, 2-D, and 3-D data. For more information on complex-valued masked channel attention, see the paper that introduced it [26].

2.5 Complex-Valued Softmax Layers

Softmax is an essential function for several tasks throughout deep learning. Complex-valued softmax functions have not been explored thoroughly in the literature at the time of this paper. However, a similar route, know as masking, has seen some attention in recent research [26].

Here, we introduce several softmax layer suitable for complex-valued tensors.

2.5.1 Split Type-A Complex-Valued Softmax Layer

Using the definition of a split Type-A function from Section 2.8.1, we define the complex-valued split Type-A softmax layer (`CVSoftMax`) as

$$\texttt{CVSoftMax}(\mathbf{z}) = \texttt{SoftMax}(\mathbf{x}) + j\texttt{SoftMax}(\mathbf{y}), \tag{10}$$

where $\mathbf{z} = \mathbf{x} + j\mathbf{y}$ and `SoftMax` is the PyTorch real-valued softmax function.

2.5.2 Phase Preserving Softmax

Similar to a Type-B function from Section 2.8.2, we define the complex-valued phase preserving softmax layer (`PhaseSoftmax`) as

$$\texttt{CVSoftMax}(\mathbf{z}) = \texttt{SoftMax}(|\mathbf{z}|) \odot \frac{\mathbf{z}}{|\mathbf{z}|}, \tag{11}$$

where $\mathbf{z} = \mathbf{x} + j\mathbf{y}$ and `SoftMax` is the PyTorch real-valued softmax function.

2.5.3 Magnitude Softmax

We define the magnitude softmax layer, which simply computes the softmax over the magnitude of the input and ignores the phase, (`MagSoftmax`) as

$$\texttt{CVSoftMax}(\mathbf{z}) = \texttt{SoftMax}(|\mathbf{z}|), \tag{12}$$

where $\mathbf{z} = \mathbf{x} + j\mathbf{y}$ and `SoftMax` is the PyTorch real-valued softmax function.

2.6 Complex-Valued Masking Layers

2.6.1 Complex Ratio Mask (cRM) or Phase Preserving Sigmoid

Detailed in Eq. (23) of [26], the complex ratio mask (cRM) applies the traditional sigmoid function to the magnitude of the signal while leaving the phase information unchanged as

$$\texttt{ComplexRatioMask}(\mathbf{z}) = \texttt{Sigmoid}(|\mathbf{z}|) \odot \frac{\mathbf{z}}{|\mathbf{z}|} \tag{13}$$

where $\mathbf{z} = \mathbf{x} + j\mathbf{y}$ and `Sigmoid` is the PyTorch real-valued softmax function.

2.6.2 Magnitude Min-Max Normalization Layer

The min-max norm, which has proven useful in complex-value data normalization [13], is another option for a complex-valued softmax function as

$$\texttt{MagMinMaxNorm}(\mathbf{z}) = \frac{\mathbf{z} - \mathbf{z}_{\text{min}}}{\mathbf{z}_{\text{max}} - \mathbf{z}_{\text{min}}} \tag{14}$$

where $\mathbf{z} = \mathbf{x} + j\mathbf{y}$ and $\mathbf{z}_{\text{min}}$ and $\mathbf{z}_{\text{max}}$ indicate the minimum and maximum of $|\mathbf{z}|$.

2.7 Complex-Valued Normalization Layers

Normalization layers are a crucial aspect of modern deep learning algorithms facilitating improved convergence and model robustness. As discussed in [4], traditional standardization of complex-valued data is not sufficient to translate and scale the data to unit variance and zero mean. Rather, a whitening procedure is necessary to ensure a circular distribution with equal variance for the real and imaginary parts of the signal. The whitening algorithm is derived in greater detail in [4], but we employ the same procedure for both batch normalization and layer normalization.

2.8 Complex-Valued Activation Layers

Complex-valued activation functions are a key element of CVNNs that different significantly from real-valued neural networks. The popular real-valued activation layers (such as ReLU and GeLU) cannot be directly applied to complex-valued data without some modification. Complex-valued activation functions must take into account the 2 degrees-of-freedom inherent to complex-valued data, typically represented as real and imaginary parts or magnitude and phase.

We highlight four categories of complex-valued activation layers:

- Split Type-A Activation Layers
- Split Type-B Activation Layers
- Fully Complex Activation Layers
- ReLU-Based Complex-Valued Activation Layers

Split *Type-A* and *Type-B* activation layers apply real-valued activation functions to either the real and imaginary or magnitude and phase, respectively, of the input signal [2, 5]. Fully complex activation layers are entirely complex-valued, while ReLU-based complex-valued activation layers are the family of complex-valued activation functions that extend the ever-popular ReLU to the complex plane.

2.8.1 Split Type-A Complex-Valued Activation Layers

Type-A activation functions consist of two real-valued functions, $G_{\mathbb{R}}(\cdot)$ and $G_{\mathbb{I}}(\cdot)$, which are applied to the real and imaginary parts of the input tensor, respectively, as

$$G(\mathbf{z}) = G_{\mathbb{R}}(\mathbf{x}) + jG_{\mathbb{I}}(\mathbf{y}). \tag{15}$$

In most cases, $G_{\mathbb{R}}(\cdot) = G_{\mathbb{I}}(\cdot)$; however, $G_{\mathbb{R}}(\cdot)$ and $G_{\mathbb{I}}(\cdot)$ can also be distinct functions. Table 3 details the Type-A activation functions included in `complextorch`. Additionally, a generalized Type-A activation function is included allowing the user to adopt any set of $G_{\mathbb{R}}(\cdot)$ and $G_{\mathbb{I}}(\cdot)$ desired.

`complextorch` Activation Layer	$G_{\mathbb{R}}(\mathbf{z}) = G_{\mathbb{I}}(\mathbf{z})$	Reference
`CVSplitTanh`	$\tanh(\mathbf{z})$	Eq. (15) [27]
`CTanh`	$\tanh(\mathbf{z})$	Eq. (15) [27]
`CVSplitSigmoid`	$\sigma(\mathbf{z})$	-
`CSigmoid`	$\sigma(\mathbf{z})$	-
`CVSplitAbs`	$\lvert\mathbf{z}\rvert$	Section III-C [28]

Table 3: Type-A activation functions.

2.8.2 Polar Type-B Complex-Valued Activation Layers

Similarly, *Type-B* activation functions consist of two real-valued functions, $G_{||}(\cdot)$ and $G_{\angle}(\cdot)$, which are applied to the magnitude (modulus) and phase (angle, argument) of the input tensor, respectively, as

$$G(\mathbf{z}) = G_{||}(|\mathbf{z}|) \odot \exp(jG_{\angle}(\angle\mathbf{z})). \tag{16}$$

Table 4 details the Type-B activation functions included in `complextorch`. Where $G_{\angle}(\cdot)$ is omitted, the phase information is unchanged and the activation is effectively a masking function as in Section 2.6. Additionally, a generalized Type-B activation function is included allowing the user to adopt any set of $G_{||}(\cdot)$ and $G_{\angle}(\cdot)$ desired.

`complextorch` Activation Layer	$G_{\|\|}(\lvert\mathbf{z}\rvert)$	$G_{\angle}(\angle\mathbf{z})$	Reference
`CVPolarTanh`	$\tanh(\lvert\mathbf{z}\rvert)$	-	Eq. (8) [27]
`CVPolarSquash`	$\frac{\lvert\mathbf{z}\rvert^2}{(1+\lvert\mathbf{z}\rvert^2)}$	-	Section III-C [27]
`CVPolarLog`	$\ln(\lvert\mathbf{z}\rvert + 1)$	-	Section III-C [29]
`modReLU`	$\texttt{ReLU}(\lvert\mathbf{z}\rvert + b)$	-	Eq. (8) [30]

Table 4: Type-B activation functions.

2.8.3 Fully Complex Activation Layers

Fully complex activation layers employ activation functions specifically designed for complex-valued data and hence do not have a general form. Table 5 details the fully-complex activation functions included in `complextorch`.

`complextorch` Activation Layer	$G(\mathbf{z})$	Reference
`CVSigmoid`	$\frac{1}{1+\exp(\mathbf{z})}$	Eq. (71) [31]
`zReLU`	$\begin{cases} \mathbf{z} & \text{if} \quad \angle\mathbf{z} \in [0, \pi/2] \\ 0 & \text{else} \end{cases}$	Section 4.2.1 [32]
`CVCardiod`	$\frac{1}{2}(1 + \cos(\angle\mathbf{z})) \odot \mathbf{z}$	Eq. (3) [33]
`CVSigLog`	$\frac{\mathbf{z}}{(c+1/r*\lvert\mathbf{z}\rvert)}$	Eq. (20) [34]

Table 5: Fully-complex activation functions.

2.8.4 ReLU-Based Complex-Valued Activation Layers

The ReLU is the most popular activation function in modern deep learning, and it has garnered significant attention in its extension to the complex domain. Most take a similar form to Type-A activation functions as in Section 2.8.1, operating on the real and imaginary parts of the input signal. However, some functions, like the Type-B `modReLU` and fully-complex Guberman ReLU (`zReLU`) apply ReLU-like operations. Some efforts have been made to develop insights into how ReLU-based complex-valued activation functions "activate" across different regions and quadrants of the complex plane [35, 32]. Table 6 details the fully-complex activation functions included in `complextorch`, where $\mathbf{z} = \mathbf{x} + j\mathbf{y}$.

2.9 Complex-Valued Loss Functions

In this section, we overview some common complex-valued loss functions. Whereas we emphasize regression loss, complex-valued classification and other loss functions are further explored in [5, 2]. Similar to activation functions, two general types of loss functions have similar forms to Type-A and Type-B activations, operating on the real and imaginary or magnitude and phase, respectively.

`complextorch` Activation Layer	$G(\mathbf{z})$	Reference
`CVSplitReLU`	$\texttt{ReLU}(\mathbf{x}) + j\texttt{ReLU}(\mathbf{y})$	Eq. (5) [36]
`CReLU`	$\texttt{ReLU}(\mathbf{x}) + j\texttt{ReLU}(\mathbf{y})$	Eq. (5) [36]
`CPReLU`	$\texttt{PReLU}(\mathbf{x}) + j\texttt{PReLU}(\mathbf{y})$	Eq. (2) [35]

Table 6: ReLU-based activation functions.

2.9.1 Split Loss Functions

Split loss functions apply two real-valued loss functions to the real and imaginary parts of the estimated ($\mathbf{x}$)) and ground truth ($\mathbf{y}$)) labels as

$$\mathscr{L}(\mathbf{x}, \mathbf{y}) = \mathscr{L}_{\mathbb{R}}(\mathbf{x}_{\mathbb{R}}, \mathbf{y}_{\mathbb{R}}) + \mathscr{L}_{\mathbb{I}}(\mathbf{x}_{\mathbb{I}}, \mathbf{y}_{\mathbb{I}}), \tag{17}$$

where the total loss is computed as the sum of the real and imaginary losses. Generally $\mathscr{L}_{\mathbb{R}}(\cdot,\cdot) = \mathscr{L}_{\mathbb{I}}(\cdot,\cdot)$; however, we include an generalized split loss function allowing the user to specify any combination of $\mathscr{L}_{\mathbb{R}}(\cdot,\cdot)$ and $\mathscr{L}_{\mathbb{I}}(\cdot,\cdot)$. Table 7 details the split loss functions included in `complextorch`.

`complextorch` Loss Function	$\mathscr{L}_{\mathbb{R}}(\cdot,\cdot) = \mathscr{L}_{\mathbb{I}}(\cdot,\cdot)$
`SplitL1`	$\texttt{L1}(\cdot,\cdot)$
`SplitMSE`	$\texttt{MSE}(\cdot,\cdot)$
`SplitSSIM`	$\texttt{SSIM}(\cdot,\cdot)$

Table 7: Split Activation Functions.

2.9.2 Polar Loss Functions

Similarly, polar loss functions apply two real-valued loss functions to the magnitude and phase of the estimated and ground truth labels as

$$G(\mathbf{x}, \mathbf{y}) = w_{||} G_{||}(|\mathbf{x}|, |\mathbf{y}|) + w_{\angle} G_{\angle}(\angle\mathbf{x}, \angle\mathbf{y}), \tag{18}$$

where $w_{||}$ and $w_{\angle}$ are scalar weights applied based on *a priori* understanding of the problem to scale the magnitude and phase losses, particularly as the phase loss will always be less than 2π by definition. Tuning the loss weights may improve modeling performanceas To the author's understanding, at the time of this paper, there have been no efforts to apply polar loss functions. However, they may, in conjunction with split loss functions, improve modeling performance by imposing additional loss on the phase-accuracy of the algorithm.

2.9.3 Other Complex-Valued Loss Functions

Several complex-valued loss functions are implemented and detailed in Table 8 Additionally, `PerpLossSSIM` (Eq. (5),

`complextorch` Activation Layer	$G\mathscr{L}(\mathbf{x}, \mathbf{y})$	Reference
`CVQuadError`	$\frac{1}{2}\text{sum}(\lvert\mathbf{x} - \mathbf{y}\rvert^2)$	Eq. (11) [37]
`CVFourthPowError`	$\frac{1}{2}\text{sum}(\lvert\mathbf{x} - \mathbf{y}\rvert^4)$	Eq. (12) [37]
`CVCauchyError`	$\frac{1}{2}\text{sum}(c^2/2\ln(1 + \lvert\mathbf{x} - \mathbf{y}\rvert^2/c^2))$	Eq. (13) [37]
`CVLogCoshError`	$\text{sum}(\ln(\cosh(\lvert\mathbf{x} - \mathbf{y}\rvert^2))$	Eq. (14) [37]
`CVLogError`	$\text{sum}(\lvert\ln(\mathbf{x}) - \ln(\mathbf{y})\rvert^2)$	Eq. (10) [3]

Table 8: Other complex-valued loss functions.

Fig. 1 [38]) is implemented. Its mathematical formulation is omitted here, but can be found in [38].

2.10 Complex-Valued Manifold-Based Layers

In [16, 18] a complex-valued convolution operator offering similar equivariance properties to the spatial equivariance of the traditional real-valued convolution operator is introduced. By approaching the complex domain as a Riemannian homogeneous space consisting of the product of planar rotation and non-zero scaling, they define a convolution operator equivariant to phase shift and amplitude scaling. Although their paper shows promising results in reducing the number of parameters of a complex-valued network for several problems, their work has not gained mainstream support. However, some initial work has shown significant promise in reducing model sizes and improving modeling capacity

for smaller models [39, 40, 41]. Incorporating manifold-based complex-valued deep learning is a promising research area for future efforts. For full derivation and alternative implementations, please refer to [16, 17, 18].

As the authors mention in the final bullet point in Section IV-A1,

> "If d is the manifold distance in (2) for the Euclidean space that is also Riemannian, then wFM has exactly the weighted average as its closed-form solution. That is, our wFM convolution on the Euclidean manifold is reduced to the standard convolution, although with the additional convexity constraint on the weights."

Hence, the implementation closely follows the conventional convolution operator with the exception of the weight normalization. We would like to not that the weight normalization, although consistent with the authors' implementation, lacks adequate explanation from the literature and could be improved for further clarity.

`complextorch` contains implementations for 1-D and 2-D versions of the proposed wFM-based convolution operated introduced in [16, 18], dubbed `wFMConv1d` and `wFMConv2d`, respectively.

2.11 Complex-Valued Pooling Layers

Complex-valued average pooling can be computed similarly to real-valued pooling where the average can be computed over the real and imaginary parts of the signal separately as

$$\texttt{CVAdaptiveAvgPoolingNd}(\mathbf{z}) = \texttt{AdaptiveAvgPoolingNd}(\mathbf{x}) + j\texttt{AdaptiveAvgPoolingNd}(\mathbf{y}), \tag{19}$$

where $\mathbf{z} = \mathbf{x} + j\mathbf{y}$. We include implementations for `CVAdaptiveAvgPooling1d`, `CVAdaptiveAvgPooling2d`, and `CVAdaptiveAvgPooling3d`.

2.12 Complex-Valued Dropout Layers

Similarly, complex-valued dropout can be computed similarly to real-valued dropout where dropout can be computed over the real and imaginary parts of the signal separately as

$$\texttt{CVDropout}(\mathbf{z}) = \texttt{Dropout}(\mathbf{x}) + j\texttt{Dropout}(\mathbf{y}), \tag{20}$$

where $\mathbf{z} = \mathbf{x} + j\mathbf{y}$.

3 Conclusion

In this paper, we introduced a PyTorch wrapper for complex-valued neural network modeling. The proposed architecture

References

[1] M Soo Kim and Clark C Guest, "Modification of backpropagation networks for complex-valued signal processing in frequency domain," in *Proc. IEEE Int. Jt. Conf. Neural Netw. (IJCNN)*, San Diego, CA, USA, June 1990, pp. 27–31.

[2] ChiYan Lee, Hideyuki Hasegawa, and Shangce Gao, "Complex-valued neural networks: A comprehensive survey," *IEEE/CAA J. Autom. Sin.*, vol. 9, no. 8, pp. 1406–1426, Aug. 2022.

[3] Joshua Bassey, Lijun Qian, and Xianfang Li, "A survey of complex-valued neural networks," *arXiv preprint arXiv:2101.12249*, Jan. 2021.

[4] Chiheb Trabelsi, Olexa Bilaniuk, Ying Zhang, Dmitriy Serdyuk, Sandeep Subramanian, Joao Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio, and Christopher J Pal, "Deep complex networks," in *Proc. Int. Conf. Learn. Repr. (ICLR)*, Vancouver, BC, Canada, Apr. 2018, pp. 27–31.

[5] Jose Agustin Barrachina, Chengfang Ren, Gilles Vieillard, Christele Morisseau, and Jean-Philippe Ovarlez, "Theory and implementation of complex-valued neural networks," *arXiv preprint arXiv:2302.08286*, Feb. 2023.

[6] J. W. Smith, *Novel Hybrid-Learning Algorithms for Improved Millimeter-Wave Imaging Systems*, Phd dissertation, Dept. Elect. Comput. Eng., Univ. Texas Dallas, Richardson, TX, USA, Apr. 2022, Available at `https://arxiv.org/abs/2306.15341`.

[7] J. W. Smith, M. E. Yanik, and M. Torlak, "Near-field MIMO-ISAR millimeter-wave imaging," in *Proc. IEEE Radar Conf. (RadarConf)*, Florance, Italy, Sept. 2020, pp. 1–6.

[8] J. W. Smith, S. Thiagarajan, R. Willis, Y. Makris, and M. Torlak, "Improved static hand gesture classification on deep convolutional neural networks using novel sterile training technique," *IEEE Access*, vol. 9, pp. 10893–10902, Jan. 2021.

[9] J. W. Smith, O. Furxhi, and M. Torlak, "An FCNN-based super-resolution mmWave radar framework for contactless musical instrument interface," *IEEE Trans. Multimedia*, vol. 24, pp. 2315–2328, May 2021.

[10] J. W. Smith and M. Torlak, "Efficient 3-D near-field MIMO-SAR imaging for irregular scanning geometries," *IEEE Access*, vol. 10, pp. 10283–10294, Jan. 2022.

[11] J. W. Smith, Y. Alimam, G. Vedula, and M. Torlak, "A vision transformer approach for efficient near-field SAR super-resolution under array perturbation," in *Proc. IEEE Tex. Symp. Wirel. Microw. Circuits Syst. (WMCS)*, Waco, TX, USA, Apr. 2022, pp. 1–6.

[12] C. Vasileiou, J. W. Smith, S. Thiagarajan, M. Nigh, Y. Makris, and M. Torlak, "Efficient CNN-based super resolution algorithms for mmWave mobile radar imaging," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Bourdeaux, France, Oct. 2022, pp. 3803–3807.

[13] J. W. Smith and M. Torlak, "Deep learning-based multiband signal fusion for 3-D SAR super-resolution," *IEEE Trans. Aerosp. Electron. Syst.*, pp. 1–17, Apr. 2023.

[14] J. W. Smith and M. Torlak, "Survey of emerging systems and algorithms for near-field THz SAR imaging," in *Proc. IEEE*, to be submitted.

[15] J. W. Smith, "Dual radar sar controller," *arXiv preprint arXiv:2309.00006*, June 2023.

[16] Rudrasis Chakraborty, Jiayun Wang, and Stella X. Yu, "Sur-real: Frechet mean and distance transform for complex-valued deep learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Long Beach, CA, USA, June 2019, pp. 889–897.

[17] Rudrasis Chakraborty, Yifei Xing, Minxuan Duan, and Stella X. Yu, "C-SURE: Shrinkage estimator and prototype classifier for complex-valued deep learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Seattle, WA, USA, June 2020, pp. 360–367.

[18] Rudrasis Chakraborty, Yifei Xing, and Stella X. Yu, "Surreal: Complex-valued learning as principled transformations on a scaling and rotation manifold," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 3, pp. 940–951, Nov. 2022.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Long Beach, CA, USA, Dec. 2017, pp. 5998–6008.

[20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al., "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Repr. (ICLR)*, Vienna, Austria, May 2021.

[21] Sachin Mehta and Mohammad Rastegari, "MobileViT: light-weight, general-purpose, and mobile-friendly vision transformer," *arXiv preprint arXiv:2110.02178*, Oct. 2021.

[22] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Montreal, Canada, Oct. 2021, pp. 10012–10022.

[23] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al., "Swin transformer v2: Scaling up capacity and resolution," *arXiv preprint arXiv:2111.09883*, Nov. 2021.

[24] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte, "SwinIR: Image restoration using swin transformer," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Montreal, Canada, Oct. 2021, pp. 1833–1844.

[25] Chuang Wang, Qunying Zhang, Jianmin Hu, Chao Li, Shuyun Shi, and Guangyou Fang, "An efficient algorithm based on CSA for THz stepped-frequency SAR imaging," *IEEE Geosci. Remote Sens. Lett.*, vol. 19, pp. 1–5, Dec. 2020.

[26] Hyun-Woong Cho, Sungdo Choi, Young-Rae Cho, and Jongseok Kim, "Complex-valued channel attention and application in ego-velocity estimation with automotive radar," *IEEE Access*, vol. 9, pp. 17717–17727, Jan. 2021.

[27] Akira Hirose and Shotaro Yoshida, "Generalization characteristics of complex-valued feedforward neural networks in relation to signal coherence," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 4, pp. 541–551, Jan. 2012.

[28] Akram Marseet and Ferat Sahin, "Application of complex-valued convolutional neural network for next generation wireless networks," in *Proc. IEEE WNYISPW*, Rochester, NY, USA, Nov. 2017, pp. 1–5.

[29] Daichi Hayakawa, Takashi Masuko, and Hiroshi Fujimura, "Applying complex-valued neural networks to acoustic modeling for speech recognition," in *Proc. IEEE APSIPA ASC*, Honolulu, HI, USA, Nov. 2018, pp. 1725–1731.

[30] Martin Arjovsky, Amar Shah, and Yoshua Bengio, "Unitary evolution recurrent neural networks," in *Proc. 33rd Int. Conf. Mach. Learn. (ICML)*, New York City, NY, USA, June 2016, pp. 1120–1128.

[31] Tohru Nitta and Yasuaki Kuroe, "Hyperbolic gradient operator and hyperbolic back-propagation learning algorithms," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 5, pp. 1689–1702, Mar. 2017.

[32] Nitzan Guberman, "On complex valued convolutional neural networks," *arXiv preprint arXiv:1602.09046*, Feb. 2016.

[33] Patrick Virtue, X Yu Stella, and Michael Lustig, "Better than real: Complex-valued neural nets for MRI fingerprinting," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Beijing, China, Sept. 2017, pp. 3953–3957.

[34] George M Georgiou and Cris Koutsougeras, "Complex domain backpropagation," *IEEE Trans. Circuits Syst. II*, vol. 39, no. 5, pp. 330–334, May 1992.

[35] Handan Jing, Shiyong Li, Ke Miao, Shuoguang Wang, Xiaoxi Cui, Guoqiang Zhao, and Houjun Sun, "Enhanced millimeter-wave 3-D imaging via complex-valued fully convolutional neural network," *Electronics*, vol. 11, no. 1, pp. 147, 2022.

[36] Jingkun Gao, Bin Deng, Yuliang Qin, Hongqiang Wang, and Xiang Li, "Enhanced radar imaging using a complex-valued convolutional neural network," *IEEE Geosci. Remote Sens. Lett.*, vol. 16, no. 1, pp. 35–39, Sept. 2018.

[37] Ronny Haensch and Olaf Hellwich, "Complex-valued convolutional neural networks for object detection in PolSAR data," in *Proc. Eur. Conf. Synth. Aperture Radar*, Aachen, Germany, June 2010, pp. 1–4.

[38] Maarten L Terpstra, Matteo Maspero, Alessandro Sbrizzi, and Cornelis AT van den Berg, "perp-loss: A symmetric loss function for magnetic resonance imaging reconstruction and image registration with deep learning," *Med. Image Anal.*, vol. 80, pp. 102509, Aug. 2022.

[39] Carter N Brown, Enrico Mattei, and Andrew Draganov, "ChaRRNets: Channel robust representation networks for RF fingerprinting," *arXiv preprint arXiv:2105.03568*, May 2021.

[40] Qingshu Liu and Liang Lang, "Mmff: Multi-manifold feature fusion based neural networks for target recognition in complex-valued sar imagery," *ISPRS J. Photogramm. Remote Sens.*, vol. 180, pp. 151–162, Oct. 2021.

[41] Theresa Scarnati and Benjamin Lewis, "Complex-valued neural networks for synthetic aperture radar image classification," in *Proc. IEEE Radar Conf. (RadarConf)*, Atlanta, GA, USA, May 2021, pp. 1–6.