Journal of Signal Processing Systems
https://doi.org/10.1007/s11265-022-01793-0

Comparison Between Equivalent Architectures of Complex-valued and Real-valued Neural Networks - Application on Polarimetric SAR Image Segmentation

José Agustín Barrachina[1,2] · **Chengfang Ren**[2] · **Christèle Morisseau**[1] · **Gilles Vieillard**[1] · **Jean-Philippe Ovarlez**[1,2]

Received: 28 February 2022 / Revised: 8 June 2022 / Accepted: 2 July 2022

Abstract
We present an in-depth statistical comparison among several Complex-Valued Neural Network (CVNN) models on the Oberpfaffenhofen Polarimetric Synthetic Aperture Radar (PolSAR) database and compare them against Real-Valued Neural Network (RVNN) architectures. The necessity to define the equivalence between the models emerges in order to compare both networks fairly. A novel definition for an equivalent-RVNN in terms of *real-valued trainable parameters* that maintain the *aspect ratio* is extended for convolutional layers based on previous work Barrachina et al. (2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP), 2021). We illustrate that CVNN obtains better statistical performance for classification on the PolSAR image across a range of architectures than a capacity equivalent-RVNN, indicating that this behavior is likely independent of the model itself.

Keywords Complex-Valued Neural Network · Real-Valued Neural Network · Polarimetric Synthetic Aperture Radar

1 Introduction

For PolSAR applications, RVNN is widely employed in the literature [2–4], which are generally acquired in complex-valued format thanks to radar I-Q channels. Hence, these networks are forced to transform these data through their absolute values, real parts, or isomorphic representations (concatenation of real and imaginary parts, absolute value and phase, etc.). In all cases, these transforms can lead to the loss of important phase information contained in each PolSAR pixels.

As a result, the signal processing community is more interested in investigating strategies for dealing with complex-valued data [5], the most common signal type in radar applications. Because the action performed at each layer of CVNNs may be understood as complex filtering, they appear to be a suitable candidate for processing and learning from this complex-valued information. These networks are better at extracting phase information [6], which might be helpful for retrieving Doppler frequency in radar signals, categorizing PolSAR data [7, 8], and radiofrequency signal processing in wireless communications, among other things [5].

CVNN has only recently been introduced to PolSAR images applications [9]. Reference [7] was one of the first to implement a Complex-Valued MultiLayer Perceptron (CV-MLP) for PolSAR applications. Although a comparison was made against Real-Valued MultiLayer Perceptron (RV-MLP), no confidence interval was given, which prevents asserting CV-MLP merits. Furthermore, a different input representation was used for each model, making it an unfair comparison. Later on, the same authors suggested giving the same input representation to get a more precise comparison between the models [10]. References [11] and [12] also used CV-MLP on a PolSAR database but did not

✉ Jean-Philippe Ovarlez
jean-philippe.ovarlez@onera.fr

José Agustín Barrachina
jose-agustin.barrachina@centralesupelec.fr

Chengfang Ren
chengfang.ren@centralesupelec.fr

Christèle Morisseau
christele.morisseau@onera.fr

Gilles Vieillard
gilles.vieillard@onera.fr

[1] DEMR, ONERA, Université Paris-Saclay, 91120 Palaiseau, France

[2] SONDRA, CentraleSupélec, Université Paris-Saclay, 91192 Gif-sur-Yvette, France

Published online: 28 July 2022

provide a comparison with RV-MLP. Ref. [13] did compare CV-MLP against a RV-MLP but even though CV-MLP performed better than RV-MLP, confidence intervals intersect, leaving room for doubt about CV-MLP outperformance.

Works using Complex-Valued Convolutional Neural Network (CV-CNN) have been published for PolSAR applications. Ref. [14] compares a CV-CNN with real-valued models but lacks confidence intervals. Other recent works [8, 15–17] use a CV-CNN for PolSAR applications but without comparing its result with a real-valued model.

Lately, [13] achieved *state-of-the-art* performance using a Complex-Valued Fully Convolutional Neural Network (CV-FCNN) model architecture. This model can adapt more naturally to the task at hand as it performs semantic segmentation by design.

All previously mentioned works, however, use different loss functions and different methods for pre-processing from a simple random flip data augmentation to an Autoencoder to extract features that are then fed to the network; this makes it impossible to compare the performance between models directly as the input representation might strongly impact on the model performances.

This paper implements several CVNN models with the same input representation providing a thorough analysis involving several independent trials for each network to infer appropriate errors and statistics. We also implement all the equivalent-RVNN extending the real-equivalent definition presented in [1] for the convolutional layers as well. This definition asserts the same quantity of *real-valued trainable parameters* for both complex- and real-valued models, maintaining the *aspect ratio* for each hidden layer.

We then show which model architecture is best suited for PolSAR applications and prove that using a Complex-Valued model is desirable regardless of the chosen architecture as long as the data format is complex-valued and its phase information matters.

The following section briefly introduces the CVNN framework to then develop the mathematics for creating a real-valued equivalent model in the next section. Sections 4 and 5 explains the models and dataset used for the experiments respectively whose results are then discussed in Sect. 6. The conclusion can be found last in Sect. 7.

2 Complex-Valued Neural Network framework

CVNN, as opposed to conventional RVNN, possess complex-valued input, which allows working with imaginary data without any pre-processing to cast its values to real. Each layer of the complex network operates analogously to a real-valued layer with the difference that its operations are on the complex domain (addition, multiplication, convolution, etc) with trainable parameters being complex-valued (weights, bias, kernels, etc.).

Activation functions are also defined on the complex domain so that $f : \mathbb{C} \to \mathbb{C}$. Reference [18] proposes two types of complex activation functions:

- Type-A: $\sigma_A(f) = \sigma_{\Re}(\Re f) + j\,\sigma_{\Im}(\Im f)$,
- Type-B: $\sigma_B(f) = \sigma_r(|f|)\, e^{j\,\sigma_\phi(\phi(f))}$,

where $f : \mathbb{C} \to \mathbb{C}$ is a complex function and $\sigma_{\Re}, \sigma_{\Im}, \sigma_r, \sigma_\phi$ are all real-valued functions.

The loss function definition must be real-valued as the notion of ordered comparison is not present for complex numbers, Sect. 4 describes how this is achieved. As the loss is real-valued, the optimizer can be the same as the one used for real-valued networks. *Wirtinger Calculus* [19] is used for computing the gradient allowing to minimize the loss function with respect to complex-valued variables, even if this function is not holomorphic. We already have shown the interest of CVNN over RVNN for non circular data in reference [20].

3 Real Equivalent Network

A Real-Valued equivalent network is necessary to assess whether a CVNN is actually of interest. Most parameters are naturally transformed into the complex plane. That is true, for example, loss functions, optimizers, which are equal, as explained in Sect. 2, or activation functions, where a Rectified Linear Unit (ReLU) can be converted to the complex plane using $\mathbb{C}$ReLU as explained in [18] and further developed on Sect. 4. However, if we keep the same amount of neurons (for fully-connected layers) or kernels (for convolutional layers), it will result in the CVNN having higher capacity than their opposed RVNN as we can consider that the complex plane $\mathbb{C}$ is isomorphic to $\mathbb{R}^2$ meaning that one complex-valued parameter ($p_{\mathbb{C}}$) is equivalent to two real-valued parameters ($p_{\mathbb{R}}$) so that $p_{\mathbb{C}} = 2\,p_{\mathbb{R}}$. The superscript $\mathbb{C}$ and $\mathbb{R}$ indicate whether it corresponds to the CVNN or RVNN respectively.

3.1 Multilayer Perceptron

To preserve the same amount of *real-valued neuron parameters* (*np*) per layer on MLP architectures, it will suffice to double the neurons of each hidden layer within the RV-MLP with respect to CV-MLP [11, 21, 22]. However, as reference [23] points out, this design leads to a bigger number of *real-valued trainable parameters* (*tp*) for the RVNN. Indeed, ignoring the layer biases that are generally added at the end, a CVNN with two consecutive hidden layers of size 10 each will result in $10 \times 10 = 100$ complex-valued

weights for connecting them, which is equivalent to a total of $tp_{\mathbb{C}} \triangleq 200$ *real-valued trainable parameters*. Using the described technique, an equivalent-RVNN will have two consecutive hidden layers of size 20 each, needing a total of $20 \times 20 = 400$ real-valued weights to connect them and, therefore, $tp_{\mathbb{R}} \triangleq 400$. Leading to the latter having potentially a higher capacity if this method is followed.

The global number of *tp* for a generic CV-MLP and RV-MLP with K hidden layers is provided by the formula [23]:

$$\begin{aligned} tp_{\mathbb{C}} &= 2\,N_0^{\mathbb{C}}\,N_1^{\mathbb{C}} + 2\sum_{i=1}^{K-1} N_i^{\mathbb{C}}\,N_{i+1}^{\mathbb{C}} + 2\,N_K^{\mathbb{C}}\,N_L^{\mathbb{C}}\,, \\ tp_{\mathbb{R}} &= N_0^{\mathbb{R}}\,N_1^{\mathbb{R}} + \sum_{i=1}^{K-1} N_i^{\mathbb{R}}\,N_{i+1}^{\mathbb{R}} + N_K^{\mathbb{R}}\,N_L^{\mathbb{R}}\,, \end{aligned} \tag{1}$$

where N_i is the number of neurons for layer $i \in 1, ..., K$. N_0 corresponds to the number of features or input size and N_L to the output size.

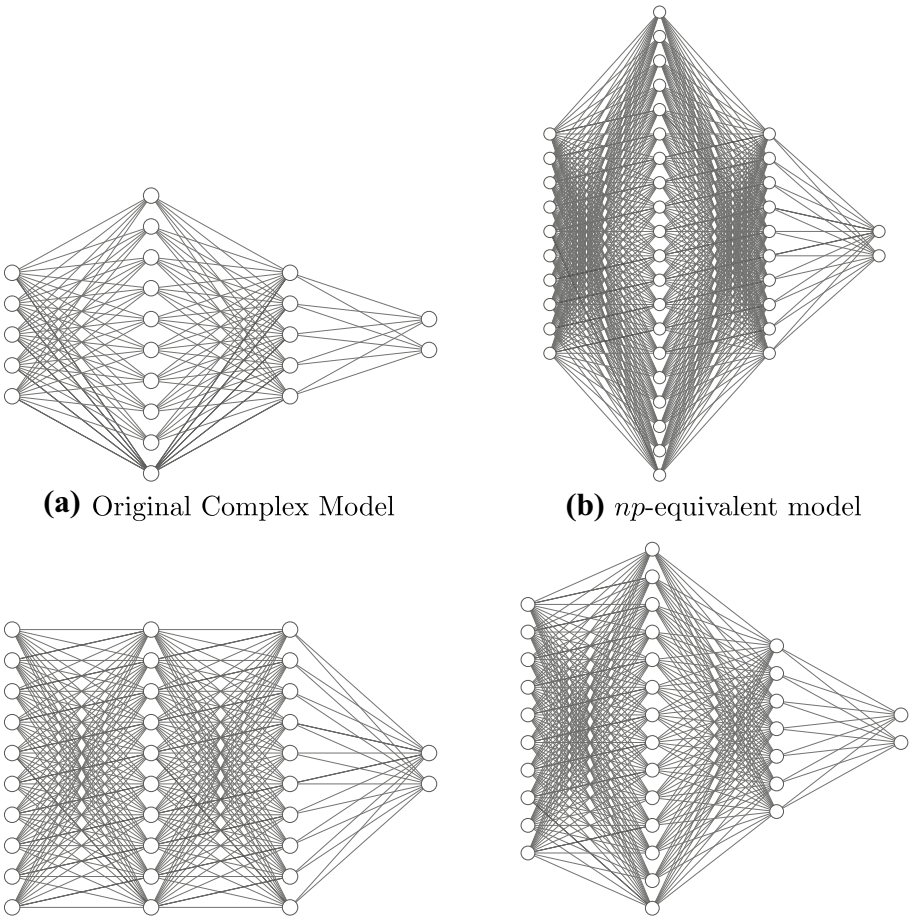
The task to solve directly determines the input and output sizes of the real network so that $N_0 = N_0^{\mathbb{R}} = 2\,N_0^{\mathbb{C}}$ and

$$N_L = N_L^{\mathbb{R}} = \begin{cases} 2\,N_L^{\mathbb{C}}, & \text{regression task} \\ N_L^{\mathbb{C}}, & \text{classification task} \end{cases}. \tag{2}$$

Reference [23] argues that a real-valued equivalent model must have the same *tp* capacity as the complex one: $tp_{\mathbb{C}} = tp_{\mathbb{R}} = tp$. To accomplish this, they propose to alternate between doubling or not the number of neurons of the real-valued model hidden layers with respect to the complex-valued model. However, this strategy only works when the number of hidden layers is even for classification tasks and an odd number for regressions tasks. To address this problem, we propose designating one hidden layer as:

$$N_i^{\mathbb{R}} = 2\,\frac{N_{i-1}^{\mathbb{C}} + N_{i+1}^{\mathbb{C}}}{N_{i-1}^{\mathbb{R}} + N_{i+1}^{\mathbb{R}}}\,N_i^{\mathbb{C}}\,. \tag{3}$$

Another proposition in [23] is to make all layers the same size. Nevertheless, this solution will not maintain the same *aspect ratio* for both CVNN and RVNN models. As exemplified in Fig. 1a and c, performing classification with a CV-MLP with two hidden layers of sizes 10 and 5 will be



(a) Original Complex Model **(b)** *np*-equivalent model

(c) *tp*-equivalent model proposed by [23] **(d)** Proposed *tp*-equivalent model [1]

Figure 1 Real equivalent MLP models example. Figures generated usingalexlenail.

converted to a RV-MLP where both hidden layer sizes are 10. This means converting a network where the first hidden layer doubles the size of the second to one where both hidden layers are the same size.

In this paper, we propose to maintain the same *aspect ratio* for each hidden layer, i.e. the amount of hidden layer neurons of RVNN is proportional to the one of RVNN, which leads to the following equation:

$$N_i^{\mathbb{R}} = r\, N_i^{\mathbb{C}},\ \forall i \in 1, ..., K \tag{4}$$

with r a positive constant real value. Replacing (4) in (1) we obtain the following second-order polynomial equation in the variable r:

$$tp = r\, N_0\, N_1^{\mathbb{C}} + \sum_{i=1}^{K-1} r^2\, N_i^{\mathbb{C}}\, N_{i+1}^{\mathbb{C}} + r\, N_K^{\mathbb{C}}\, N_L\,. \tag{5}$$

Since r should be positive as well as all parameters tp, $N_i^{\mathbb{R}}$, N_L and N_0, the only possible solution to our problem is therefore:

$$r = \frac{-b + \sqrt{b^2 - 4\,a\,(-tp)}}{2\,a}\,, \tag{6}$$

where $a = \sum_{i=1}^{K-1} N_i^{\mathbb{C}}\, N_{i+1}^{\mathbb{C}}$, $b = N_0\, N_1^{\mathbb{C}} + N_k^{\mathbb{C}}\, N_L$.

In conclusion, there are two possible definitions for an equivalent-RV-MLP. Either by setting the same *real-valued trainable parameters* (*tp*) or by its *real-valued neuron parameters* (*np*) per hidden layer (Fig. 1b). The former can be done by creating a RV-MLP where each hidden layer size is given by Eq. (4) with r being defined by (6); this will result in an equivalent-RV-MLP in terms of the *real-valued training parameters* that maintain the same aspect ratio that the CVNN hidden layers (Fig. 1d).

If we assume $r < 1$,

$$\begin{aligned} &a\, r^2 + b\, r - tp = 0 < r\, a + r\, b - tp, \\ \Rightarrow\ &0 < r\, a + r\, b - 2\, a - \beta \le a\,(r-2) + b\,(r-1), \end{aligned} \tag{7}$$

where $tp = 2\,a + \beta$ with $b \le \beta = 2\, N_0^{\mathbb{C}}\, N_1^{\mathbb{C}} + 2\, N_k^{\mathbb{C}}\, N_L^{\mathbb{C}} < 2\,b$ (Eq. (1)). As both a and b are positive, Eq. (7) is absurd, which is expected as it implies that real-valued models will never have fewer neurons than the complex-valued models. On the other hand, for $r \ge 1$:

$$\begin{aligned} &a\, r^2 + b\, r - tp = 0 \ge r\, a + r\, b - tp, \\ \Rightarrow\ &0 \ge a\,(r-2) + b\, r - \beta > a\,(r-2) + b\,(r-2)\,. \end{aligned} \tag{8}$$

Again, as a and b are positive, Eq. (8) is absurd if $r \ge 2$. Because of inequalities (7) and (8), we conclude that $1 \le r < 2$, meaning that the equivalent-RVNN should have at least the same dimension as CVNN and at most double. In particular, $r = 2$ corresponds to the case for the same value of *np*. Proving that it is not possible to reach both conditions simultaneously and one must choose between setting an equal value for *np* or *tp*.

For single hidden layer models, $a = 0$ and therefore, r will be:

$$r = \frac{\beta}{b} = 2\,\frac{N_0^{\mathbb{C}} + N_L^{\mathbb{C}}}{N_0 + N_L}\,. \tag{9}$$

As it can be derived from (9), $r = 1$ for regressions tasks while for classifications tasks, $1 < r < 2$ depending on the relationship between N_0 and N_L. Finally, as the number of hidden neurons gets bigger with respect to the input and output, or in other words, $a \gg b$, it will tend $r \to \sqrt{2}$.

Note that, the extra terms $2\sum_{i=1}^{K} N_i^{\mathbb{C}}$ and $\sum_{i=1}^{K} N_i^{\mathbb{R}}$ should be added to Eq. (1) in order to take into account the bias. This extra term will lead to a slight variation of r by changing the value of b but does not change its boundary $1 \le r < 2$.

3.2 Convolutional Neural Networks

For convolutional layers, the equation of the *real-valued trainable parameters* is defined by:

$$\begin{aligned} tp_{\mathbb{C}} &= 2\,\sum_{i=1}^{K} C_i^{\mathbb{C}}\, W_i^{\mathbb{C}} H_i^{\mathbb{C}} F_i^{\mathbb{C}}\,, \\ tp_{\mathbb{R}} &= \sum_{i=1}^{K} C_i^{\mathbb{R}}\, W_i^{\mathbb{R}} H_i^{\mathbb{R}} F_i^{\mathbb{R}}\,, \end{aligned} \tag{10}$$

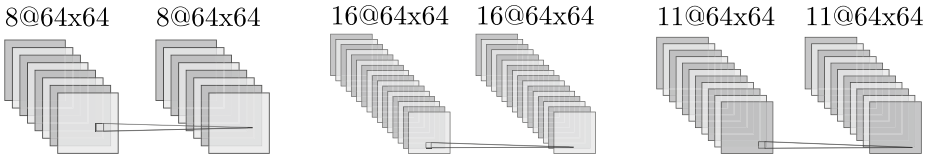
with C the channels presented on the input of the convolutional layer, W and H the filter width and height respectively and F the amount of filters or kernels of the layer.

There are a few options for convolutional layers on how to maintain the same amount of *real-valued trainable parameters*, either by extending the kernel sizes or increasing the number of kernels. The second method may seem more logical as the transformation from the complex to the real plane does not change the resolution of the image to justify changing the kernel size. By adopting the second method then $H_i^{\mathbb{C}} = H_i^{\mathbb{R}}$ and $W_i^{\mathbb{C}} = W_i^{\mathbb{R}}$. By definition, $C_i = F_{i-1}$ with F_0 the input image channel dimension. To achieve the same *real-valued trainable parameters*, we need to solve Eq. (11) for $r \in \mathbb{R}$ in order to have $F_i^{\mathbb{R}} = r\, F_i^{\mathbb{C}}$, $\forall i \ge 1$.

$$tp = 2\,\sum_{i=1}^{K} F_{i-1}^{\mathbb{C}}\, W_i\, H_i\, F_i^{\mathbb{C}} = 2 r F_0^{\mathbb{C}}\, W_1\, H_1\, F_1^{\mathbb{C}} + r^2\,\sum_{i=2}^{K} F_{i-1}^{\mathbb{C}}\, W_i\, H_i\, F_i^{\mathbb{C}}\,. \tag{11}$$

From above equation, the solution for r is:

$$r = \frac{-b}{2\,a} + \sqrt{2 + \frac{b}{a} + \frac{b^2}{4\,a^2}}\,, \tag{12}$$



(a) Original Complex Model **(b)** *np*-equivalent model **(c)** *tp*-equivalent model

Figure 2 Real equivalent convolution example of a middle hidden layer. Figures generated using alexlenail.

with $b = 2F_0^{\mathbb{C}} W_1 H_1 F_1^{\mathbb{C}}$ and $a = \sum_{i=2}^{K} F_{i-1}^{\mathbb{C}} W_i H_i F_i^{\mathbb{C}}$. The extreme case of a single convolutional layer makes $r = 1$. Similarly as for MLP, $r \to \sqrt{2}$ when $a >> b$, which can occur, for example, with deep the convolutional neural networks (see Fig. 2).

If taking into account the bias, Eq. (10) changes to:

$$\begin{aligned} tp_{\mathbb{C}} &= 2 \sum_{i=1}^{K} \left(C_i^{\mathbb{C}} W_i^{\mathbb{C}} H_i^{\mathbb{C}} + 1 \right) F_i^{\mathbb{C}}, \\ tp_{\mathbb{R}} &= \sum_{i=1}^{K} \left(C_i^{\mathbb{R}} W_i^{\mathbb{R}} H_i^{\mathbb{R}} + 1 \right) F_i^{\mathbb{R}}. \end{aligned} \tag{13}$$

With these variations, Eq. 12 changes to

$$r = \frac{-b}{2a} + \sqrt{2 + \frac{b}{a} + \frac{b^2}{4a^2} + \frac{\sum_{i=1} F_i}{a}}, \tag{14}$$

where a is as before but b is now $b = 2F_0^{\mathbb{C}} W_1 H_1 F_1^{\mathbb{C}} + \sum_{i=1} F_i$.

4 Model Architectures

To this date, difficulties in implementing CVNN models in practice have slowed down the field from growing further [23]. An open-sourced and well-documented tool has been developed, which enables and facilitates the implementation of CVNNs [24] for the community to exploit further. This tool also allows generating from a complex-valued network model its real-equivalent model as described in Sect. 3. All implementations were done using this software library. The code that contains the exact model used for this paper simulations can be found in [24].

MLP [7], CNN [14] and FCNN [13] model architectures were implemented both on the complex and real domain, respecting the equivalence definitions discussed in Sect. 3. However, some minor modifications were made to update some of those models with *state-of-the-art* parameters not popular or known at the time of those publications.

References [7] and [14] use Stochastic Gradient Descent (SGD) as an optimizer, whereas reference [13] uses a more modern optimizer known as Adam [25], which might allow models to find a lower optimal minimum. Adam was, therefore, used as the optimizer for all models. Learning rate and momentum were tweaked experimentally for each model independently, as were the epochs for each model.

Although [7] use tanh activation function for the MLP model, we decided in this work to use ReLU (Type-A ReLU also known as). Indeed, both activation functions were tested for the MLP architecture showing an interesting increase in performance when using ReLU. This small optimization, plus using the Adam optimizer, made the MLP architecture go from a median average accuracy per class (based on 15 iterations) of $83.75 \pm 0.13\%$ to $85.25 \pm 0.05\%$ for the complex model and from $83.31 \pm 0.11\%$ to $84.38 \pm 0.16\%$ for the real model. For the output layer, the *softmax* activation function [26] has been used. For the complex-valued models, Type A activation functions were used.

A Normal weight initialization by He et al. in [27] was used and the bias was initialized as zero. The adaptation for complex-valued weights initialization is described in [28, p. 6], which has to be done with care to keep the benefits of the He et al. initialization on the complex domain.

Categorical cross-entropy loss function was used for all models. The loss is computed twice for complex models, using first the real part and then the imaginary part as the prediction result. An average of the two error values is then calculated to be optimized.

The MLP models presented some overfitting for what dropout with 50% rate was used which ameliorated the performance. Both architectures had two hidden layers. For the CV-MLP, 96 and 180 neurons were used for the first and second hidden layer respectively, as presented in [13]. The hidden layers sizes of the RV-MLP were dimensioned to have the same amount of *real-valued training parameters* with the same aspect ratio as explained in Sect. 3 (Fig. 1d).

Throughout literature, CV-CNN are the most popular CVNN architectures. All references [8, 14–17] identically dimensioned the model with the same amount of layers and kernels. Therefore, we decided to use the same architecture, presenting two convolutional layers with 6 and 12 kernels each for the complex model. All kernels are of size 3×3. The real model was dimensioned as explained in Sect. 3. Average-pooling was used between both convolutional layers whose extension to the complex domain is evident. The model presents a fully connected layer to perform the classification at the end.

Finally, CV-FCNN (Fig. 3) was implemented as described in [13]. Which is composed of the downsampling or feature extraction part and the upsampling part. The downsampling part presents several blocks (B1, B2, B3, B4, B5 and B6). Each block presents two sub-modules represented in Fig. 3 in green and red colors. The upsampling part presents blocks B7, B8, B9, B10 and B11, which, in term, are a combination of other two sub-modules, the second one being the same green sub-module present in the downsampling section. The first sub-module (yellow) is a max-unpooling module, as explained in [29].

The green sub-module is a combination of a convolution layer, a BatchNormalization (BN) (complex adaptations explained in [30], Sections 3.2 and 3.5) and ReLU. The convolutional filter present on each layer was of size 3×3 and the number used for each layer is represented in Fig. 3 for the complex model. As usual, the definition in Sect. 3 was used to dimension the real-valued model.

The red sub-module is a max-pooling layer whose main objective is to shrink the image into smaller ones by keeping only the maximum value within a small window, in our case, of size 2×2. For the complex case, the absolute value of the complex number is used for comparison as proposed in [14]. This layer complements the max-unpooling sub-module (yellow), which receives the locations where the maximum value was found. The max-unpooling layer enlarges the input image by placing pixels according to the maxed locations received from the corresponding max-pooling layer [29].

The last blocks of the downsampling and upsampling parts (B6 and B11) have some differences with respect to the other blocks. B6 removes the max-pooling layer (red) completely. B11, on the other hand, replaces the ReLU activation function with a *softmax* activation function to be used for the output layer.

Each model was evaluated over 50 Monte-Carlo trials to be able to extract statistics analysis.

5 PolSAR image

PolSAR images are acquired from single look complex data measured in the horizontal (H) and vertical (V) transmit/receive polarimetric channels known as the Sinclair scattering matrix:

$$\mathbf{s} = \left(S_{HH}, \sqrt{2}\, S_{HV}, S_{VV}\right)^T . \tag{15}$$

For each pixel of the Synthetic Aperture Radar (SAR) image, the four components are usually expressed in Pauli basis as one complex vector $\boldsymbol{k} \in \mathbb{C}^3$ [31], so that:

$$\boldsymbol{k} = \frac{1}{\sqrt{2}}\left(S_{HH} + S_{VV}, S_{HH} - S_{VV}, 2\, S_{HV}\right)^T . \tag{16}$$

The Hermitian so-called coherency matrix is then formally built according to

$$\boldsymbol{T} = \frac{1}{n}\sum_{j}^{n} \boldsymbol{k}_j \boldsymbol{k}_j^H , \tag{17}$$

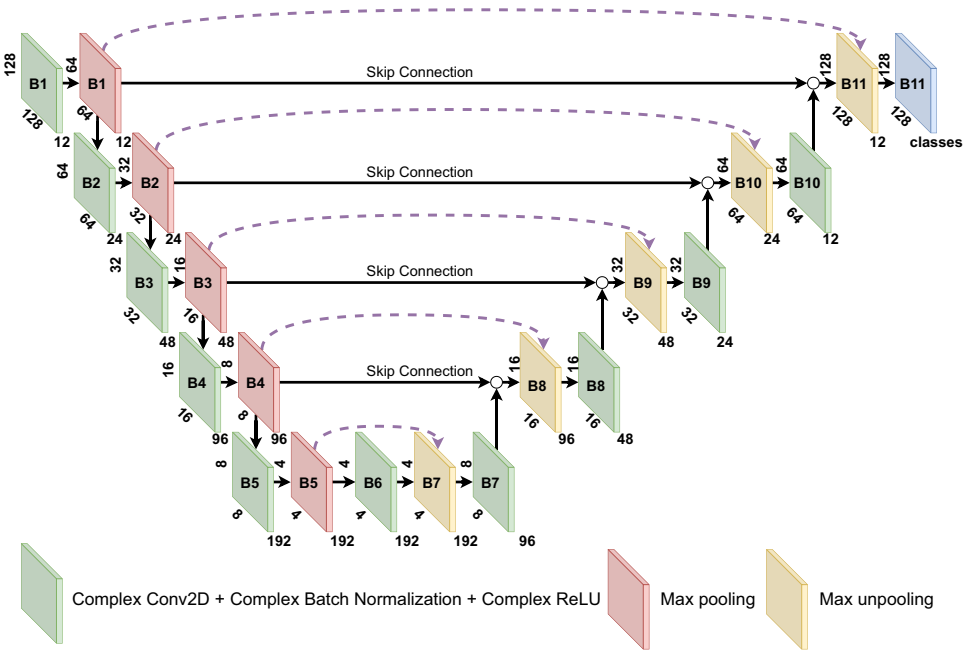


Figure 3 Complex-Valued Fully Convolutional Neural Network diagram.

(a) Polarimeric data representation **(b)** Ground Truth

Figure 4 PolSAR data of Oberpfaffenhofen, Germany. **A** Built-up Area; **B** Woodland; **C** Open Area.

where the operator ^{H} stands for complex conjugate operation and where n is the number of neighboring pixels chosen in a boxcar around the considered one. The coherency matrix is then used as input to the networks. For the real-valued network, both real and imaginary parts are injected separately into the network.

The experiments were run over the well-known Oberpfaffenhofen database that can be downloaded from the European Space Agency (ESA) website. The coherency matrix is provided as the representation of data. Because the diagonal elements of the coherency matrix are real-valued, they have been treated as a complex-valued number with the imaginary part equal to zero. Since $\boldsymbol{T}$ is Hermitian symmetric, its lower triangle, excluding the diagonal, has been discarded as it provided no additional information; this finally led to a total of 6 complex values per pixel. Figure 4a corresponds to the Red-Green-Blue color composition of the diagonal elements of $\boldsymbol{T}$, while Fig. 4b shows the ground truth for three different classes (built-up areas, woodland and open areas). These labels were obtained from [14].

As the image is very large, previous works used a small percentage of pixels for training to speed up training and validation results. References [7] and [11] used about 2% of the image pixels for training, whereas [3] and [32] used 5%. In [33], the authors adopted 10%. Finally, reference [14] tested different sampling rates and proposed, based on the results, to use 10% sampling rate for both train and validation set together. Therefore, we have chosen to use 8% as the training set and only 2% as the validation set, which corresponds to 104928 and 26232 pixels respectively. The remaining 90% was used for the test set. Both train and validation sets had the same amount of examples per class as, regardless of the class occurrences, the application does not prioritize one class over another. This may result in different accuracies for validation and test sets.

6 Experimental Results

Statistical indicators of both the Overall Accuracy (OA), which is the ratio of the number of correctly predicted pixels divided by the total number of pixels, and the Average Accuracy (AA), which is an average of the accuracy for each class independently are summarized in Table 1 for the six experimental models.

The median error was computed as in [34]; if median intervals do not overlap, there is a 95% confidence that their values differ [35]. The confidence interval of the mean is calculated for a confidence level of 99%.

Experiments were done with all *np*, alternate-*tp* and ratio-*tp* real equivalent models. The models where based on the Complex-Valued MultiLayer Perceptron (CV-MLP) described in Sect. 4 under the same conditions as the simulations presented on Sect. 6. Figure 5 shows the first 500 epochs for the validation loss value of these simulations. It can be seen that both the *np* and alternate-*tp* models presented overfitting. In contrast, this was not present (or to a very limited extent) in the ratio-*tp* and complex models. Therefore, we can conclude that using the proposed ratio-*tp* technique works best for this case of study, giving proof that our method might be the one to be favored in these cases for which we will only use this equivalent network definition for the rest of our simulations.

From Table 1, it is evident that FCNN outperforms CNN which, at the same time, outperforms MLP. CVNN outperforms its real-valued equivalent model in both OA

Table 1 Test accuracy results (%).

		Overall Accuracy (OA)			
		median	mean	IQR	range
FCNN	CV	**98.55 ± 0.21**	98.42 ± 0.09	97.99 – 98.94	99.91 – 99.44
	RV	98.23 ± 0.15	98.30 ± 0.08	98.02 – 98.69	96.83 – 99.28
CNN	CV	96.45 ± 0.04	96.45 ± 0.02	96.36 – 96.52	96.21 – 96.68
	RV	96.32 ± 0.04	96.32 ± 0.02	96.24 – 96.44	95.89 – 96.65
MLP	CV	88.87 ± 0.03	88.86 ± 0.02	87.78 – 88.93	88.61 – 89.13
	RV	88.03 ± 0.13	87.94 ± 0.06	87.64 – 88.24	86.90 – 88.91
		Average Accuracy (AA)			
		median	mean	IQR	range
FCNN	CV	**98.14 ± 0.28**	97.68 ± 0.23	97.38 – 98.68	90.97 – 99.41
	RV	97.79 ± 0.30	97.38 ± 0.22	96.93 – 98.31	91.54 – 99.00
CNN	CV	95.69 ± 0.05	95.68 ± 0.02	95.57 – 95.81	95.27 – 96.00
	RV	95.50 ± 0.06	95.47 ± 0.03	95.34 – 95.63	94.82 – 95.93
MLP	CV	85.25 ± 0.05	85.24 ± 0.04	85.13 – 85.38	84.60 – 86.03
	RV	84.38 ± 0.16	84.25 ± 0.08	83.92 – 84.62	82.59 – 85.42

Bold entries are the highest median and mean of AA and OA

and AA metrics for all three model architectures. The highest accuracy was achieved by the CV-FCNN architecture with an OA of 98.55% and an AA of 98.14%. Achieving the highest mean or median does not guarantee the most performing trained model. Indeed, we can argue that the maximum obtained value may be more important than the average accuracy as, in most cases, the most performing model will be used for end-user applications. In this case, CV-FCNN was also the model that obtained both the upper 75% of cases and the maximum highest accuracy.

Unfortunately, the dataset is highly imbalanced, having many more occurrences of class C (Open Areas) than the other classes. In particular, class C always obtained a significantly higher accuracy than class B (Woodland), as it can be appreciated on Fig. 6, which caused the OA to be higher than the AA. Because of application purposes, it is usually desired that a model performs better on classifying classes equally without any preference regardless of the class occurrences for what we decided to favor AA over OA.

Figure 7 shows a randomly selected predicted image from all models. The performance difference between FCNN, CNN and MLP remains clear based on the predicted image. However, the better generalization gained when using a complex model is much harder to visualize, although the difference can be seen in particular sections of the image.

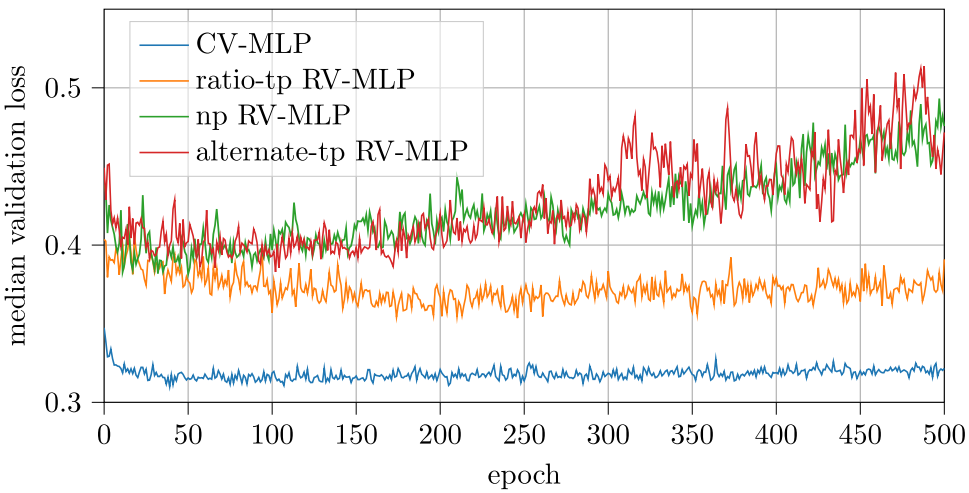


Figure 5 Real-Valued Equivalent-MLP comparison.

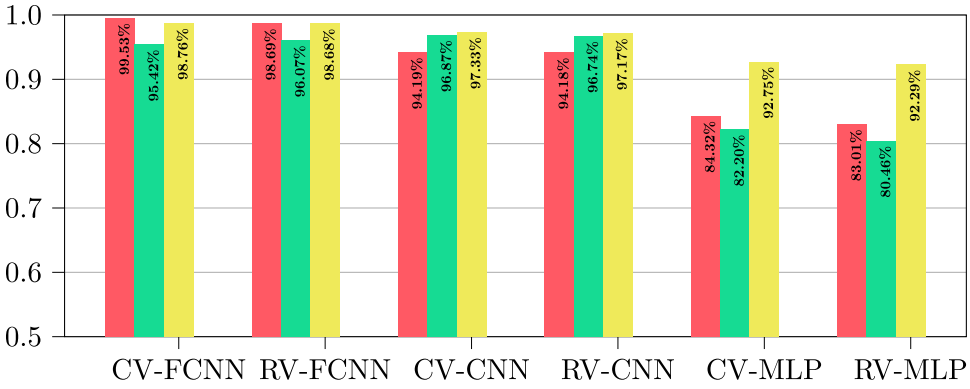


Figure 6 Accuracy per class for all models. **A** Built-up Area; **B** Woodland; **C** Open Area.

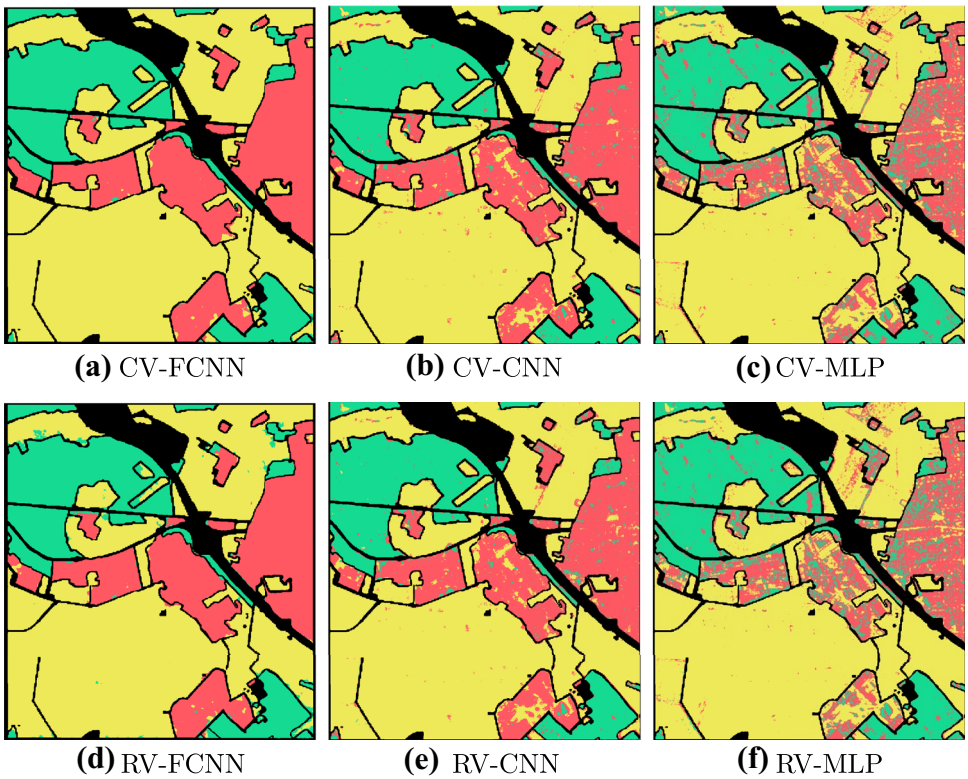


Figure 7 Comparison between model predictions. **A** Built-up Area; **B** Wood Land; **C** Open Area.

7 Conclusions

To provide a fair comparison between Complex-Valued Neural Networks and Real-Valued Neural Networks, we suggested a novel definition of an equivalent-Real-Valued Neural Network. Despite this parity, the classification performance of CVNN on the Oberpfaffenhofen PolSAR database indicates a superiority over their equivalent-RVNN. Although these differences may be considered small, Complex-Valued Neural Network out-performance is statistically justified as the confidence intervals remain very far apart. The complex structure of the PolSAR data, in which the phase information is relevant to enhance the classification accuracy, can explain the merits of CVNN.

We also proved that Fully Convolutional Neural Network generalizes better than Convolutional Neural Network and MultiLayer Perceptron models for this application.

Acknowledgements The authors would like to thank the Délégation Générale de l'Armement (DGA) for funding and the Metz campus of CentraleSupélec for providing the DCE cluster to run our simulations.

References

1. Barrachina, J. A., Ren, C., Vieillard, G., Morisseau, C., & Ovarlez, J.-P. (2021). About the equivalence between complex-valued and real-valued fully connected neural networks - application to

Polinsar images. In *2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP)* (pp. 1–6).
2. Chen, S., Wang, H., Xu, F., & Jin, Y.-Q. (2016). Target classification using the deep convolutional networks for SAR images. *IEEE Transactions on Geoscience and Remote Sensing, 54*(8), 4806–4817.
3. Hou, B., Kou, H., & Jiao, L. (2016). Classification of polarimetric SAR images using multilayer autoencoders and superpixels. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 9*(7), 3072–3081.
4. Zhou, Y., Wang, H., Xu, F., & Jin, Y.-Q. (2016). Polarimetric SAR image classification using deep convolutional neural networks. *IEEE Geoscience and Remote Sensing Letters, 13*(12), 1935–1939.
5. Bassey, J., Qian, L., & Li, X. (2021). A survey of complex-valued neural networks. arXiv:2101.12249
6. Hirose, A., & Yoshida, S. (2012). Generalization characteristics of complex-valued feedforward neural networks in relation to signal coherence. *IEEE Transactions on Neural Networks and learning systems, 23*(4), 541–551.
7. Hänsch, R., & Hellwich, O. (2009). Classification of polarimetric SAR data by complex valued neural networks. In *ISPRS Workshop High-resolution Earth Imaging for Geospatial Information* (vol. 38, pp. 4–7).
8. Zhao, J., Datcu, M., Zhang, Z., Xiong, H., & Yu, W. (2019). Contrastive-regulated CNN in the complex domain: A method to learn physical scattering signatures from flexible PolSAR images. *IEEE Transactions on Geoscience and Remote Sensing, 57*(12), 10116–10135.
9. Hirose, A. (2013). Complex-valued neural networks: Advances and applications. Hoboken, New Jersey: John Wiley & Sons.
10. Hänsch, R. (2010). Complex-valued multi-layer perceptrons - an application to polarimetric SAR data. *Photogrammetric Engineering & Remote Sensing, 76*(9), 1081–1088.
11. Hänsch, R., & Hellwich, O. (2010). Complex-valued convolutional neural networks for object detection in PolSAR data. In *8th European Conference on Synthetic Aperture Radar* (pp. 1–4).
12. De, S., Bruzzone, L., Bhattacharya, A., Bovolo, F., & Chaudhuri, S. (2017). A novel technique based on deep learning and a synthetic target database for classification of urban areas in PolSAR data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 11*(1), 154–170.
13. Cao, Y., Wu, Y., Zhang, P., Liang, W., & Li, M. (2019). Pixel-wise PolSAR image classification via a novel complex-valued deep fully convolutional network. *Remote Sensing, 11*(22), 2653.
14. Zhang, Z., Wang, H., Xu, F., & Jin, Y.-Q. (2017). Complex-valued convolutional neural network and its application in polarimetric SAR image classification. *IEEE Transactions on Geoscience and Remote Sensing, 55*(12), 7177–7188.
15. Sun, Q., Li, X., Li, L., Liu, X., Liu, F., & Jiao, L. (2019). Semi-supervised complex-valued GAN for polarimetric SAR image classification. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS 2019)* (pp. 3245–3248).
16. Zhao, J., Datcu, M., Zhang, Z., Xiong, H., & Yu, W. (2019). Learning physical scattering patterns from PolSAR images by using complex-valued CNN. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS 2019)* (pp. 10019–10022).
17. Qin, X., Zou, H., Yu, W., & Wang, P. (2021). Superpixel-oriented classification of polsar images using complex-valued convolutional neural network driven by hybrid data. *IEEE Transactions on Geoscience and Remote Sensing, 59*(12), 10094–10111.
18. Kuroe, Y., Yoshid, M., & Mori, T. (2003). On activation functions for complex-valued neural networks: existence of energy functions. *Artificial Neural Networks and Neural Information Processing, ICANN/ICONIP 2003* (pp. 985–992). Berlin, Heidelberg: Springer.
19. Wirtinger, W. (1927). Zur formalen theorie der funktionen von mehr komplexen veränderlichen. *Mathematische Annalen, 97*(1), 357–375.
20. Barrachina, J. A., Ren, C., Morisseau, C., Vieillard, G., & Ovarlez, J.-P. (2021). Complex-valued vs. real-valued neural networks for classification perspectives: An example on non-circular data. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 2990–2994). https://doi.org/10.1109/ICASSP39728.2021.9413814
21. Hirose, A. (2012). *Complex-valued neural networks* (Vol. 400). Tokyo, Japan: Springer.
22. Hirose, A. (2009). Complex-valued neural networks: The merits and their origins. In *2009 International Joint Conference on Neural Networks* (pp. 1237–1244).
23. Mönning, N., & Manandhar, S. (2018). Evaluation of complex-valued neural networks on real-valued classification tasks. arXiv preprint arXiv:1811.12351
24. Barrachina, J. A. (2020). NEGU93/CVNN: Complex Valued Neural Networks (CVNN). Zenodo.
25. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980
26. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press, Genetic Programming and Evolvable Machines.
27. He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1026–1034).
28. Trabelsi, C., Bilaniuk, O., Zhang, Y., Serdyuk, D., Subramanian, S., Santos, J. F., Mehri, S., Rostamzadeh, N., Bengio, Y., & Pal, C. J. (2017). Deep complex networks. arXiv preprint arXiv:1705.09792
29. Zafar, I., Tzanidou, G., Burton, R., Patel, N., & Araujo, L. (2018). Hands-on convolutional neural networks with tensorflow: solve computer vision problems with modeling in tensorflow and Python. Birmingham, UK: Packt Publishing Ltd.
30. Trabelsi, C., Bilaniuk, A., Zhang, Y., Serdyuk, D., Subramanian, S., Santos, J. F., Mehri, S., Rostamzadeh, N., Bengi, Y., & Pal, J. C. (2018). Deep Complex Networks.
31. Lee, J. S., & Pottier, E. (2017). Polarimetric radar imaging: from basics to applications. Boca Raton: CRC press.
32. Jiao, L., & Liu, F. (2016). Wishart deep stacking network for fast polsar image classification. *IEEE Transactions on Image Processing, 25*(7), 3273–3286.
33. Guo, Y., Wang, S., Gao, C., Shi, D., Zhang, D., & Hou, B. (2015). Wishart RBM based DBN for polarimetric synthetic radar data classification. In *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)* (pp. 1841–1844).
34. McGill, R., Tukey, J. W., & Larsen, W. A. (1978). Variations of box plots. *The American Statistician, 32*(1), 12–16.
35. Chambers, J. M. (2018). Graphical methods for data analysis. Florida, US: CRC Press.