Complex Valued AI in Agriculture

Presented By:

Aashutosh Joshi (22AG10003) & **Nipun Bharadwaj (22AG30027)**

Supervisor:

Prof. Mahesh Mohan M R

COMPLEX VALUED NEURAL NETWORKS

Complex-valued neural networks (CVNNs) are a type of neural network where the weights, inputs and outputs are represented using complex numbers.

CVNNs use complex-valued activation functions. Standard activation functions like ReLU and Sigmoid don't work directly in the complex domain, so functions like the complex extension of the Sigmoid or phase-preserving activation functions like modulus-based activations are used.

The learning process in complex valued neural network requires a modification of backpropagation, where the gradients of the loss function are computed with respect to the complex-valued weights. This involves using the Wirtinger derivatives, a tool for differentiating functions with complex variables.

ACTIVATION FUNCTIONS IN CVNN

Since complex numbers have both a real and imaginary component, activation functions need to be modified to handle the complex domain.

Split Activation Function

- **Split ReLU**: The real and imaginary parts of the complex input are processed separately by the ReLU function.

$$\mathrm{ReLU}(z)=\mathrm{ReLU}(\mathrm{Re}(z))+i\cdot\mathrm{ReLU}(\mathrm{Im}(z))$$

- **Split Sigmoid**: The real and imaginary parts can be passed through a sigmoid function.

$$\mathrm{Sigmoid}(z)=\mathrm{Sigmoid}(\mathrm{Re}(z))+i\cdot\mathrm{sigmoid}(\mathrm{Im}(z))$$

ModReLU

ModReLU is one of the most commonly used activation functions in CVNNs. It modifies the modulus (or magnitude) of the complex input, while leaving the phase unchanged. For a complex input **$z=x+iy$**, the modulus is **$|z|=x^2+y^2$.** In the equation below, b is a bias term, and **$z/|z|$** keeps the phase intact. This is useful when the phase carries important information, and only the amplitude needs to be adjusted.

$$\mathrm{ModReLU}(z)=\max(|z|+b,0)*(z/|z|)$$

Complex Sigmoid

The complex extension of the sigmoid function can be used in CVNNs.

$$\mathrm{Sigmoid}(z)=1/(1+e^{(-z)})$$

Here z is a complex number. In practice, this can often split into its real and imaginary parts, making it somewhat analogous to split activations.

Complex Hyperbolic Tangent (Tanh)

Tanh can be extended to the complex domain as well. For complex input z, the hyperbolic tangent function is be defined as:

$$\mathrm{Tanh}(z)=\{e^z-e^{(-z)}\}/\{e^z+e^{(-z)}\}$$

Like complex sigmoid this can be computationally challenging due to the complex exponential terms.

zReLU

In zReLU (Complex ReLU) activation is applied only when both the real and imaginary parts are positive.

$$\mathrm{zReLU}(z)=\{z;\ \text{if } \mathrm{Re}(z)>0 \text{ and } \mathrm{Im}(z)>0, \text{otherwise } 0\}$$

This activation enforces that both components must be positive for the activation to pass the signal.

BACKPROPAGATION IN CVNN

Backpropagation in the complex domain for complex-valued neural networks (CVNNs) is an extension of the standard backpropagation algorithm used in real-valued networks. The gradients must be computed using techniques that work for complex-valued functions.

Key Concepts in Complex Backpropagation

- **Wirtinger Calculus**: The core of backpropagation in CVNNs involves using Wirtinger derivatives, a tool from complex calculus that allows for differentiation of non-analytic complex-valued functions.
  For a complex function $f(z)$, where $z=x+iy$ the Wirtinger derivatives are:

$$\partial f/\partial z^*=(\partial f/\partial x+i\partial f/\partial y)/2$$
$$\partial f/\partial z=(\partial f/\partial x-i\partial f/\partial y)/2$$

  $z^*=x-iy$ is the complex conjugate of z.

- **Forward Pass in Complex Networks**: The forward pass in a complex-valued neural network is similar to a real-valued network except the inputs, weights and biases are all complex numbers. Below “f” is the activation function.

$$z^{(l+1)}=f(W^{(l)}*z^{(l)}+b^{(l)})$$

- **Backward Pass**: The backward pass computes the gradient of the loss function L with respect to the network's parameters (weights and biases). This process is done layer by layer propagating the error backward through the network, just like in real-valued networks. In CVNNs both real and imaginary parts are taken into account.

  Using Wirtinger derivatives the gradient of the loss function with respect to the weights $W^{(l)}$ are calculated as follows:

$$\partial L/\partial W^{(l)}=\{\partial L/\partial z^{(l+1)}\}*\{\partial z^{(l+1)}/\partial W^{(l)}\}$$

  The difference here from real-valued networks is that the gradients must now be calculated with respect to both $z^{(l)}$ and $z^{(l)*}$, as complex numbers have two independent degrees of freedom.

The update rule for the complex weights $W^{(l)}$ is as follows:

$$W^{(l)} \leftarrow W^{(l)} - \eta \frac{\partial L}{\partial W^{(l)}}$$

Here η is the learning rate.

- **Chain Rule**: The chain rule in the complex domain follows a similar principle to that in real-valued networks, with the key difference being the need to account for the complex nature of the variables.

  $$\delta^{(l)} = \frac{\partial L}{\partial z^{(l+1)}} \frac{\partial z^{(l+1)}}{\partial z^{(l)}}$$

  This error term $\delta^{(l)}$ is now complex-valued, and the backpropagation algorithm must adjust the real and imaginary parts of the weights and biases based on both components.

- **Gradient of the Loss Function**: The gradient of the loss function in a CVNN is typically computed with respect to the complex-valued weights and biases. For a given layer the gradients can be calculated the Wirtinger derivatives.

COMPLEX BATCH NORMALIZATION (CBN)

It is an extension of traditional batch normalization designed specifically for neural networks that deal with complex-valued data. The mathematical formulation of Complex Batch Normalization (CBN) extends the principles of real-valued batch normalization to the complex domain.

CBN Steps

- **Complex Mean:** The mean of the complex input data is calculated by treating the real and imaginary parts separately.

$$\mu(z)=E(z)=E(x+iy)=E(x)+iE(y)$$

Thus, for a batch of N samples $\{z(1),z(2),\dots,z(N)\}$ the complex mean is:

$$\mu(z)=\{\sum[x(i)+iy(i)]\}/N$$

- **Complex Covariance and Whitening:** In the complex domain, simply normalizing based on variance is insufficient because the real and imaginary parts may be correlated. Therefore, we compute the covariance matrix Σ(z), which is a 2×2 matrix representing the covariances between the real and imaginary parts.

$$\Sigma z = \begin{pmatrix} E[(x-\mu(x))(x-\mu(x))] & E[(x-\mu(x))(y-\mu(y))] \\ E[(y-\mu(y))(x-\mu(x))] & E[(y-\mu(y))(y-\mu(y))] \end{pmatrix}$$

The diagonal elements represent the variances of the real and imaginary parts and the off-diagonal elements represent their covariances.

Now to decorrelate the real and imaginary parts, whitening is performed using the covariance matrix. The whitening transformation is as follows:

z(Tilda)={Σ(z−μ(z))}^(-0/5)

here {Σ}^(-0.5) is the inverse square root of the covariance matrix. This ensures that the normalized variables are uncorrelated and have unit variance.

- **Normalization:** After whitening, the data z is now normalized. The normalized complex-valued input z(Tilda) can be written as:

$$\hat{z} = \frac{z-\mu(z)}{\{\Sigma z\}^\wedge(0.5)}$$

  This step ensures that both real and imaginary parts have zero mean and unit variance.

- **Affine Transformation:** Similar to standard batch normalization, after normalization, an affine transformation is applied to the complex input to allow the model to learn the optimal scale and shift.

$$\tilde{z} = \gamma\hat{z} + \beta$$

  Here, γ is a complex-valued scaling factor applied to both the real and imaginary parts and β is a complex-valued shift (bias) applied similarly.

  In practice, γ and β are learned parameters, where $\gamma=\gamma(r)+i\gamma(i)$ and $\beta=\beta(r)+i\beta(i)$.

COMPLEX DROPOUT

Complex dropout is an extension of standard dropout, adapted to handle the complex-valued nature of activations. In real-valued neural networks, dropout involves randomly setting some neurons' activations to zero during training, forcing the network to learn more robust representations. Complex dropout needs to maintain the mathematical properties of complex numbers, such as their phase and magnitude, while ensuring a proper regularization effect.

For complex-valued dropout, simply applying real-valued masks (like the binary mask used in real-valued dropout) is insufficient. This is because setting the real and imaginary parts to zero independently could disrupt the balance between the magnitude and phase. A more careful approach is required. In real-valued neural networks, dropout is typically applied as follows during training

$$\tilde{\mathbf{h}} = \mathbf{h} \odot \mathbf{m}$$

where h represents the activations, m is a binary mask (with values of 0 or 1, where some elements are randomly set to 0), and $\odot$ denotes element-wise multiplication.

Here are the main approaches used for complex dropout:

- **Independent Dropout for Real and Imaginary Parts:** In this approach, two independent masks are applied, one for the real part and one for the imaginary part of the complex activations. If z=x+iy represents a complex activation, the complex dropout would be:

$$\tilde{z}=(m(1)\odot x)+i(m(2)\odot y)$$

here m(1) and m(2) are independent random binary masks. This approach can lead to unintended modifications of the phase and magnitude which may not always be desirable.

- **Same Mask for Both Parts:** To preserve the relationship between the real and imaginary parts, another approach is to apply the same mask to both the real and imaginary components.

$$\tilde{z}=m\odot z=m\odot(x+iy)$$

here m is a binary mask applied to both the real and imaginary parts simultaneously. This approach preserves the complex structure more robustly but still allows for some level of regularization.

- **Magnitude-Phase Dropout:** Since the magnitude and phase are crucial components of complex numbers, another method applies dropout directly to the magnitude while leaving the phase untouched. In this case, we modify the complex activation as follows

  $$\tilde{z}=\tilde{r}e^{(i\theta)}$$

  where $\tilde{r}=m\odot r$ and m is a binary mask applied to the magnitude. This ensures that the regularization affects the magnitude but preserves the phase information, which can be important in tasks that rely on the direction of complex numbers.

  In each case, the key idea is to adapt the dropout mechanism to maintain important properties of complex numbers. These methods can be selected based on the network architecture and the nature of the problem being solved.

COMPLEX VALUED CONVOLUTION

Complex-valued convolution is an extension of real-valued convolution used in fields where signals or data can have both real and imaginary components. In complex convolution, both the input signal x and the convolution filter w are complex-valued.

Steps Involved in Complex-valued Convolution

- **Convolution Formula:** The discrete convolution of a signal x and a filter w for each output element at index t is given as follows:

$$y(t)=\sum\{x(n)*w(t-n)\}$$

For complex-valued convolution, the signal x and filter w are complex so the above equation will be as follows:

$$y(t)=\sum\{(x[real](n)+ix[imaginary](n))*(w[real](t-n)+iw[imaginary](t-n))\}$$

Here x[real](n) and x[imaginary](n) are the real and imaginary parts of the signal, and w[real](n) and w[imaginary](n) are the real and imaginary parts of the filter.

- **Expand the Formula:** By expanding the terms for real and imaginary parts in the above equation we get:

$$y(t)=\sum[(x[real](n)*w[real](t-n)-x[imaginary](n)*w[imaginary](t-n))+i(x[real](n)*w[imaginary](t-n)+x[imaginary](n)*w[real](t-n))]$$

In the above equation the real part and the imaginary part is as follows:

Real part:

$$y[real](t)=\sum[x[real](n)*w[real](t-n)-x[imaginary](n)*w[imaginary](t-n)]$$

Imaginary part:

$$y[imaginary](t)=n\sum[x[real](n)*w[imaginary](t-n)+x[imaginary](n)*w[real](t-n)]$$

The final output y(t) is a complex number y[real](t)+iy[imaginary](t), which are the real and imaginary components, respectively.

- **Backpropagation in Complex-valued Networks:** In a neural network backpropagation will require gradients with respect to both the real and imaginary parts of the parameters.

  For the real part x[r], the gradient: $\partial L/\partial x[r]=\{(\partial L\partial y[r])*w[r]\}-\{(\partial L/\partial y[i])*w[i]\}$

  For the imaginary part x[i], the gradient: $\partial L/\partial x[i]=\{\partial L/\partial y[r]*(-w[i])\}+\{(\partial L/\partial y[i])*w[r]\}$

  Here x[r] and w[r] corresponds to real part and x[i] and w[i] corresponds to imaginary part

- **Fourier Transform Connection:** Convolution in the time domain can be efficiently computed as multiplication in the frequency domain using the Fourier Transform. For complex signals the Fourier transform works naturally on both the real and imaginary components.

$$F\{x \ast w\}(f)=F\{x\}(f) \cdot F\{w\}(f)$$

  where F{x}(f) and F{w}(f)) are the Fourier transforms of the signal and the filter respectively.

COMPLEX POOLING

Complex pooling is responsible for down sampling and reducing the spatial size of feature maps while maintaining key properties of complex-valued features. Since complex numbers carry both magnitude and phase information, the pooling strategy must be adapted to handle both components effectively.

Types of Complex Pooling

- **Complex Magnitude Pooling:** It is based on the magnitude of the complex number. The idea is to pool the complex numbers by considering their magnitudes, while potentially preserving the phase information for reconstruction in subsequent layers.

$$\text{pooled}=\max(|z1|,|z2|,\ldots,|zn|)$$

This operation can be followed by preserving the phase, where the phase θ of the selected pooled complex number is retained:

$$\theta=\arg(z)=\arctan(y/x)$$

Hence, the pooled output z(pooled) is

$$z(\text{pooled})=|z(\text{pooled})| * e^{\wedge}(i\theta)$$

- **Complex Average Pooling:** In complex average pooling, the average of the complex numbers within the pooling region is computed. Both the real and imaginary components are averaged separately.

  Given a set of complex numbers $z(1)=x(1)+iy(1)$, $z(2)=x(2)+iy(2)$,., $z(n)=x(n)+iy(n)$ the average pooled complex number $z(avg)$ is:

  $$\mathbf{z(avg)=\{\sum(x(i)+iy(i))\}/n=1/n\{\sum x(i)+i\sum y(i)\}}$$

- **Complex Max Pooling:** Similar to real-valued max pooling, complex max pooling selects the complex number with the maximum magnitude from the pooling region.

  Let $z(1),z(2),\ldots,z(n)$ be the complex numbers in the pooling region. The pooled value is the complex number $z(max)$ that has the largest magnitude:

  $$\mathbf{z(max)=z(k) \text{ where } |z(k)| =max(|z(1)| , |z(2)| ,\ldots, |z(n)|)}$$

- **Phase-Aware Pooling:** In phase-aware pooling, pooling is performed considering both the magnitude and phase of the complex numbers. Different techniques can be applied to handle the phase:
    - **Magnitude-Preserved Phase Pooling**: Pool the magnitudes first, and then compute the average or dominant phase.
    - **Phase-Max Pooling**: Select the phase corresponding to the complex number with the maximum magnitude.

  Let the magnitude $|z|$ and phase θ be separated, and pooling is done as:

  $$|z(\text{pooled})| = \max(|z(1)|, |z(2)|, \ldots, |z(n)|)$$

  $$\theta(\text{pooled}) = \arg(z(\max))$$

  This way, both the magnitude and phase are pooled together.

Complex pooling methods offer a range of techniques for handling both the magnitude and phase of complex-valued data. They ensure that the spatial and feature information from complex-valued layers is preserved and reduced meaningfully.

COMPLEX-VALUED CONVOLUTIONAL AUTOENCODER NETWORK

A complex-valued convolutional autoencoder network is an extension of traditional convolutional autoencoders (CAEs) where the input data, weights, and activations are complex numbers instead of real numbers.

Key Components

- **Convolutions:** The convolution operation between the input and the kernel applies both real and imaginary parts. This helps capture phase and amplitude information in the data.
- **Complex-Valued Activations:** Non-linear operations for complex numbers involve splitting the data into real and imaginary components or using a modulus-based activation.
- **Autoencoder Structure:** Like a real-valued convolutional autoencoder, it consists of an encoder and a decoder. Encoder reduces the dimensionality of the complex-valued input data, extracting key features in a compressed latent space. Decoder reconstructs the original complex-valued data from the latent representation.

SYNTHETIC APERTURE RADAR (SAR)

Synthetic Aperture Radar (SAR) is a form of radar that is used to create detailed images of landscapes and objects. Unlike optical imaging systems, SAR operates in the microwave spectrum, making it capable of capturing images regardless of weather conditions or time of day.

SAR systems emit microwave signals towards the Earth's surface and measure the time it takes for the reflected signals to return. By processing these returned signals, SAR can generate high-resolution images. It works in the microwave region in the electromagnetic spectrum

Amplitude Information encodes valuable information about surface features' material properties, texture, and roughness.

Phase Information - encapsulates geometric information related to the spatial arrangement and topographical characteristics of surface features

TYPES OF SAR

- **Polar SAR**: It involves the use of radar waves transmitted with different polarization states, which are then compared to the polarization states of the reflected waves. By analyzing the polarization differences, Polar SAR can detect information about the surface properties (e.g., soil moisture, vegetation, and surface roughness). It is used to measure the polarization properties of radar waves reflected from the Earth's surface.

- **ScanSAR**: It operates by sequentially switching the radar beam to different sub-swaths or strips during a single pass. It is designed to cover a wide swath (large area) at the expense of spatial resolution.

- **Spotlight SAR**: The radar beam is steered to continuously illuminate a specific area, allowing for longer illumination time and higher resolution images. It achieves high spatial resolution over a small area.

- **Strip-map SAR**: Here the radar beam is pointed to the side of the flight path and continuously collects data as the platform moves forward. This results in a long, narrow image strip. It provides continuous coverage along a strip of the Earth's surface with consistent resolution.

- **InSAR**: InSAR involves the use of two or more SAR images of the same area taken at different times. By comparing the phase difference between the images, InSAR can detect minute changes in the Earth's surface (e.g., caused by earthquakes, subsidence or glacier movement). It is used to measure ground surface deformation with high precision.

- **TOPSAR**: It creates a series of small, overlapping sub-swaths through progressive scanning, ensuring each point on the ground is observed multiple times from slightly different angles. This method maintains high resolution and consistent image quality across wide areas. It is designed to improve the quality and efficiency of SAR imagery by systematically varying the antenna beam position.

ENHANCED RADAR IMAGING USING A COMPLEX-VALUED CONVOLUTIONAL NEURAL NETWORK

The conventional sigmoid function is extended to CV-CNN. However, many recent publications suggest that Rectified Linear Unit (ReLU) is a more effective activation. So the complex-valued ReLU (cReLU) as follows:

$$f = \mathrm{cReLU}(v') = \max\left(v'_{\mathrm{R}}, 0\right) + \mathrm{j}\max\left(v'_{\mathrm{I}}, 0\right)$$

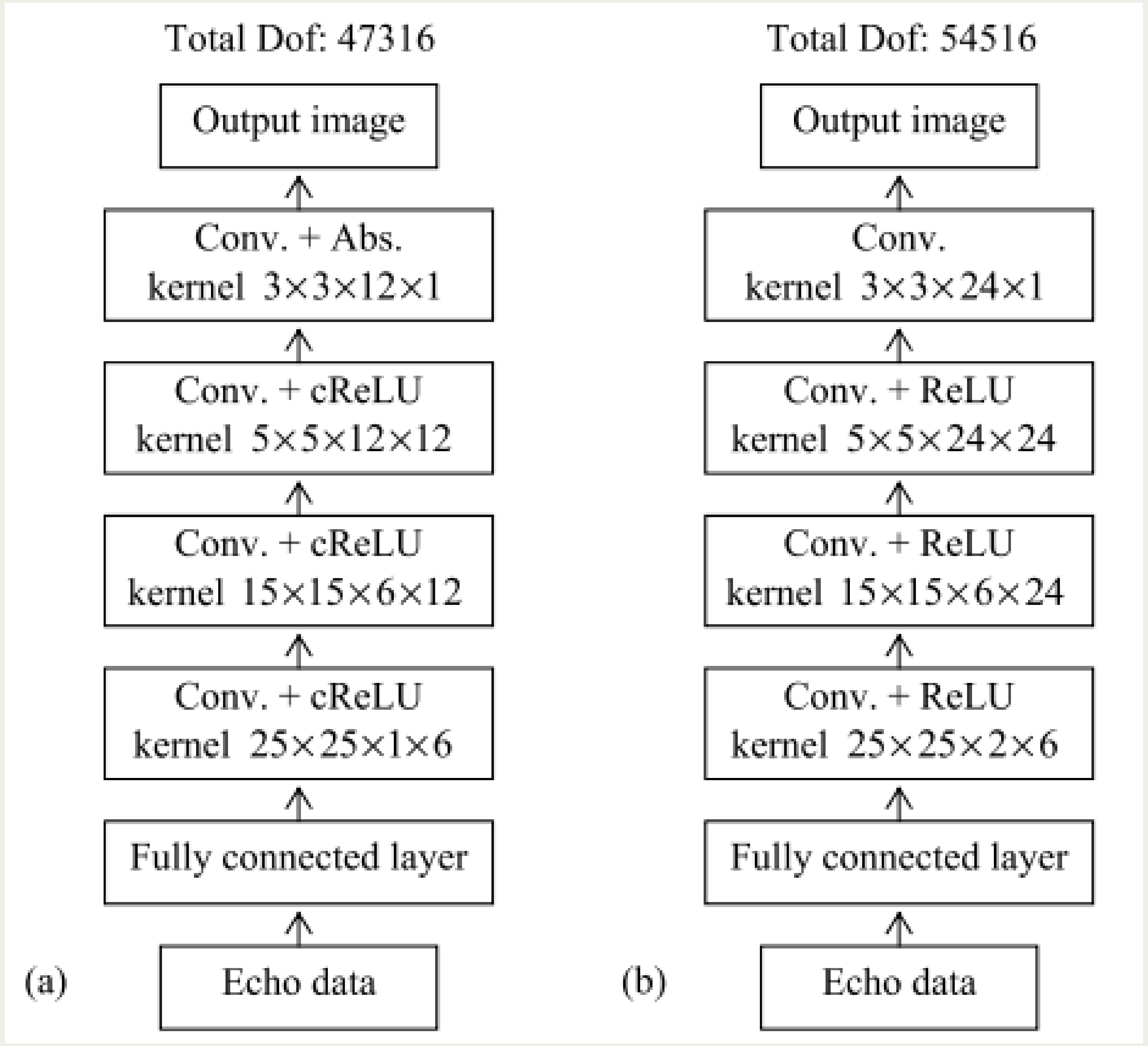
We define the corresponding output layer activation function as

$$f^{o} = \left|v_R^o + jv_I^o\right| = \sqrt{\left(v_R^o\right)^2 + \left(v_I^o\right)^2}$$

where the superscript "o" indicates the output layer and f o ∈ R. We can name this function as "Abs." Then the definition of cost function can be given

$$E = \frac{1}{2}\sum (O - f^{o})^2$$

where the summation operator is applied to all the output layer neurons, O ∈ R is the target function



Network structures of (a) proposed CV-CNN and (b) counterpart RV-CNN is shown in the figure.

CNN-based method is more robust since it is adaptive to the training data. Compared with its real valued counterpart, CV-CNN achieves better performance with less parameters.

AN APPROACH OF SEA CLUTTER SUPPRESSION FOR SAR IMAGES BY SELF-SUPERVISED COMPLEX-VALUED DEEP LEARNING

In order to make full use of the amplitude and phase information of the complex SAR image, CV-UNet++ is used as the clutter suppression model. The convolutional layer, pooling layer, and activation function in UNet++ are all extended to the complex domain.
The calculation process is expressed as follows:

$$\begin{aligned}\mathbf{a}^l &= \sigma_{\Re-\Im}(\mathbf{z}^l)\\ &= \sigma\left(\Re(\mathbf{W}^l) * \Re(\mathbf{a}^{l-1}) - \Im(\mathbf{W}^l) * \Im(\mathbf{a}^{l-1})\right) + \Re(\mathbf{b}^l)\\ &\quad + j\sigma\left(\Re(\mathbf{W}^l) * \Im(\mathbf{a}^{l-1}) + \Im(\mathbf{W}^l) * \Re(\mathbf{a}^{l-1})\right) + \Im(\mathbf{b}^l)\end{aligned}$$

For the complex variable z, the activation process is as follows:

$$\sigma_{\Re-\Im}(z) = \sigma(\Re(z)) + j\sigma(\Im(z))$$

Assuming the error term of the lth convolutional layer is δ^(l) , then the error term of the l − 1th layer could be derived as

$$\begin{aligned}\delta^{l-1} = {} & \Big(\Re(\delta^l) * \left(\Re(\mathbf{W}^l)\right)^{\Theta} \\ & \quad + \Im(\delta^l) * \left(\Im(\mathbf{W}^l)\right)^{\Theta}\Big) \odot \sigma'\left(\Re(\mathbf{z}^{l-1})\right) \\ & - j\Big(\Re(\delta^l) * \left(\Im(\mathbf{W}^l)\right)^{\Theta} \\ & \quad - \Im(\delta^l) * \left(\Re(\mathbf{W}^l)\right)^{\Theta}\Big) \odot \sigma'\left(\Im(\mathbf{z}^{l-1})\right)\end{aligned}$$

The gradients of lth convolutional layer's weight and bias are as follows:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \mathbf{W}^l} &= \Re(\mathbf{a}^{l-1}) * \Re(\boldsymbol{\delta}^l) + \Im(\mathbf{a}^{l-1}) * \Im(\boldsymbol{\delta}^l) \\ &\quad + j\left(\Re(\mathbf{a}^{l-1}) * \Im(\boldsymbol{\delta}^l) - \Im(\mathbf{a}^{l-1}) * \Re(\boldsymbol{\delta}^l)\right) \\ \frac{\partial \mathcal{L}}{\partial \mathbf{b}^l} &= \sum \delta^l.\end{aligned}$$

Now the update formula of weight and bias in the lth convolutional layer are

$$\begin{aligned}\mathbf{W}^l &\leftarrow \mathbf{W}^l + \eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}^l} \\ \mathbf{b}^l &\leftarrow \mathbf{b}^l + \eta \frac{\partial \mathcal{L}}{\partial \mathbf{b}^l}\end{aligned}$$

The pooling layer could be regarded as sampling the feature maps. T pooling layer in CV-UNet++ is represented as

$$\mathrm{ave}_{\Re-\Im}(z) = \mathrm{ave}(\Re(z)) + j\,\mathrm{ave}(\Im(z))$$

The target patch set has ϰS elements and the clutter patch set ϰC. ϰS contains 20 000 target patches. ϰC contains 40 000 clutter patches. The patch size is W ×H = 128 × 128. The number of epochs is ten, and the number of training samples in each batch is 16. The initial learning rate is 0.0001 and is reduced to 0.00001 for six epochs. The convolution kernel numbers of CV-UNet++ are 4, 8, 16, 32, and 64, respectively. For a fair comparison, UNet++ has the same degree of freedom (DoF) as CV-UNet++, so the convolution kernel numbers of UNet++ are 8, 16, 32, 64, and 128, respectively.

The ratio of the radar signal to clutter and noise (SCNR) is used to compare the performance of the clutter suppression method.

Experiments show that the CV-UNet++ improves the clutter suppression performance while fully retaining the information of the target of interest compared with traditional methods.

DESPECKLING POLARIMETRIC SAR DATA USING A MULTISTREAM COMPLEX-VALUED FULLY CONVOLUTIONAL NETWORK

A multi stream complex-valued architecture named CV-deSpeckNet is used to estimate the covariance matrix C˜ and the noise N˜ .
This architecture is used to train a robust model for learning feature representation and the underlying noise distribution in a complex-valued data, i.e., two identical FCNs estimate C˜ (FCNcov) and N˜ (FCNnoise).

Both (FCNcov) and (FCNnoise) consist of three main building blocks: complex-convolution (CV-Conv), complex-activation (CReLU) and complex-batch normalization (CV-BN)

The deep learning objective is, therefore, formulated as

$$\arg\min_{\theta} \sum L(f_\theta(\hat{C}), C).$$

Here, Cˆ is the observed rank 1 covariance matrix, C is the reference covariance matrix used as a proxy for the noise free covariance matrix, and f(θ) is the deep neural network parameterized by the complex-valued parameters θ learned under the loss function L.

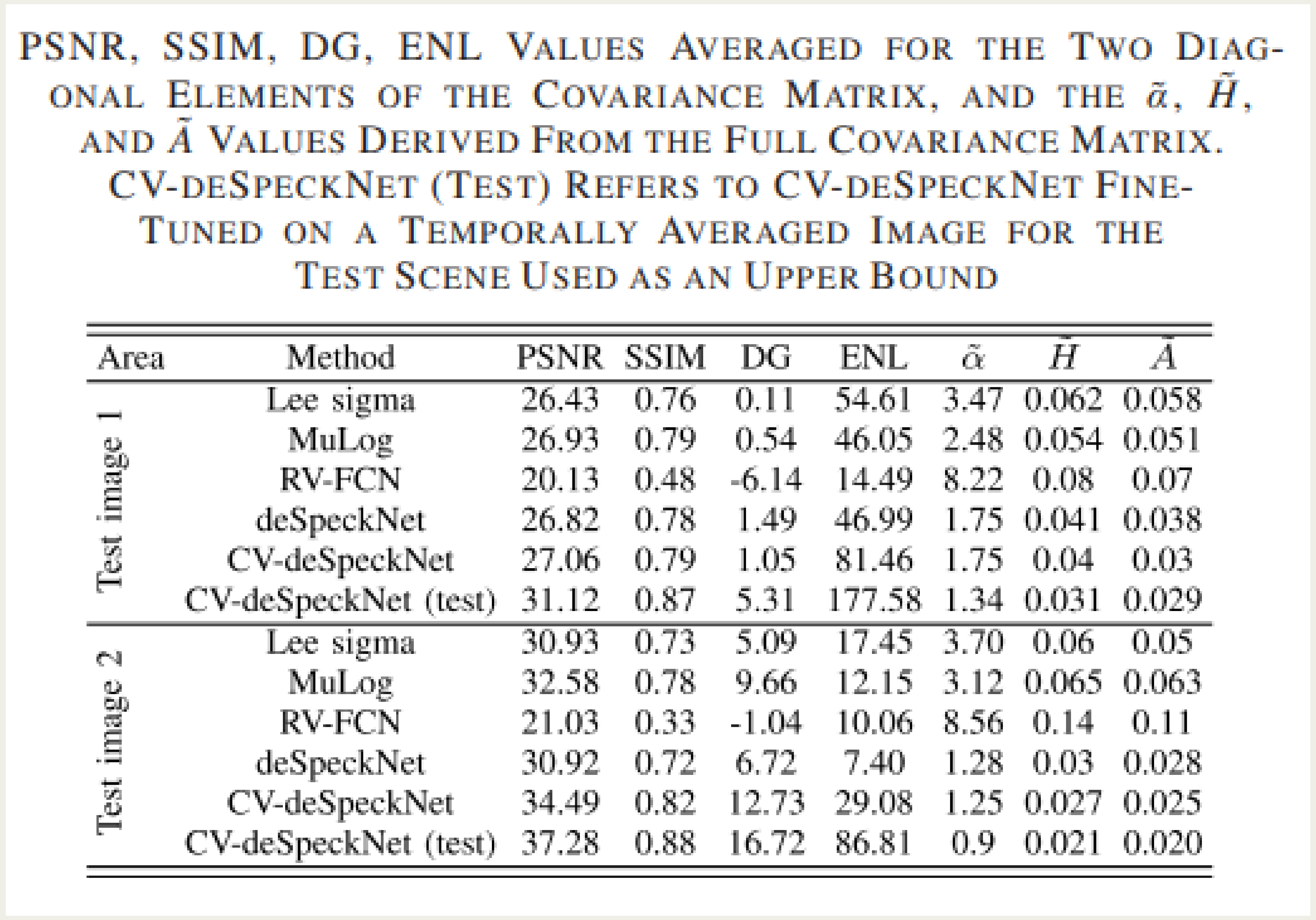
To train the network, we employ two types of loss functions, the sum squared error (SSE)-based Lcov and Lnoise, which are combined as L = Lcov + Lnoise.

$$L_{\mathrm{cov}}(\tilde{C}, C) = \mu \sum_{i=1}^{H} \sum_{j=1}^{W} \sum_{d=1}^{D} \frac{1}{2} (\tilde{C}_{ijd} - C_{ijd})^2 \qquad L_{\mathrm{noise}}(\hat{\tilde{C}}, \hat{C}) = \xi \sum_{i=1}^{H} \sum_{j=1}^{W} \sum_{d=1}^{D} \frac{1}{2} (\hat{\tilde{C}}_{ijd} - \hat{C}_{ijd})^2.$$

where H, W, and D are the height, width, and the number of feature maps in the estimated covariance matrix, respectively, and μ is the weight assigned to the loss. ξ is the weight assigned to Lnoise.

The networks were trained using the Adam optimization method. Both FCNcov and FCNnoise consist of 17 complexvalued convolutions with 48 filters. Complex-valued Batch normalization was used for every complex-valued convolution layer except the first and prediction layers. The networks were trained for 50 epochs with a learning rate of 10−3 for 30 epochs and an additional 20 epochs with a rate of 10−4. We set the weight (μ) of the Lcov to 100, and the weight (ξ) of LNoise was given a value of 1. To apply this, we used a training set of 58 368 randomly selected patches of size 40 × 40 × 6 representing the real and imaginary parts of the unique upper triangular elements of the covariance matrix. A minibatches of 64 samples and a weight decay factor of 5 × 10−4 were used.

To evaluate the despeckling ability of CV-deSpeckNet and the compared methods without reference data, we used visual inspection as a qualitative measure and the equivalent number of looks (ENL) derived in a homogeneous region as a quantitative measure for comparison.



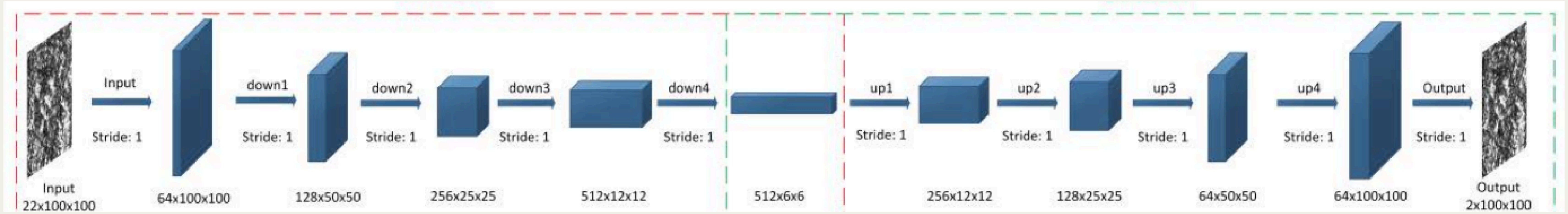
PSNR, SSIM, DG, ENL VALUES AVERAGED FOR THE TWO DIAGONAL ELEMENTS OF THE COVARIANCE MATRIX, AND THE $\tilde{\alpha}$, $\tilde{H}$, AND $\tilde{A}$ VALUES DERIVED FROM THE FULL COVARIANCE MATRIX. CV-DESPECKNET (TEST) REFERS TO CV-DESPECKNET FINE-TUNED ON A TEMPORALLY AVERAGED IMAGE FOR THE TEST SCENE USED AS AN UPPER BOUND

Area	Method	PSNR	SSIM	DG	ENL	$\tilde{\alpha}$	$\tilde{H}$	$\tilde{A}$
Test image 1	Lee sigma	26.43	0.76	0.11	54.61	3.47	0.062	0.058
	MuLog	26.93	0.79	0.54	46.05	2.48	0.054	0.051
	RV-FCN	20.13	0.48	-6.14	14.49	8.22	0.08	0.07
	deSpeckNet	26.82	0.78	1.49	46.99	1.75	0.041	0.038
	CV-deSpeckNet	27.06	0.79	1.05	81.46	1.75	0.04	0.03
	CV-deSpeckNet (test)	31.12	0.87	5.31	177.58	1.34	0.031	0.029
Test image 2	Lee sigma	30.93	0.73	5.09	17.45	3.70	0.06	0.05
	MuLog	32.58	0.78	9.66	12.15	3.12	0.065	0.063
	RV-FCN	21.03	0.33	-1.04	10.06	8.56	0.14	0.11
	deSpeckNet	30.92	0.72	6.72	7.40	1.28	0.03	0.028
	CV-deSpeckNet	34.49	0.82	12.73	29.08	1.25	0.027	0.025
	CV-deSpeckNet (test)	37.28	0.88	16.72	86.81	0.9	0.021	0.020

CV-deSpeckNet proved to be effective in estimating the covariance matrix while preserving the image quality with modest-sized training data. It was also able to learn robust feature representation that was able to adapt to a new test image. It provided better estimation results than the state-of-the-art methods and its real-valued counterpart, and a comparable performance with FCN models tuned with temporally averaged images.

COMPLEX-VALUED AUTOENCODERS WITH COHERENCE PRESERVATION FOR SAR

In this study encoding framework is used to represent the CV-SAR images from the lower resolution azimuth sub aperture images and the coherency images between them, in a lower dimension latent space, and the decoding framework is used to reconstruct the CV-SAR images. In order to exploit not only the amplitude, but also the phase information in the CV-SAR images, the AE paradigm is extended into the complex domain. In this model, all the layers including convolutional, pooling, batch normalization, and fully connected layers are in complex domain. In addition, the weights and biases in the neurons, and the activation functions are complex-valued. Furthermore, a complex-valued backpropagation, based on the stochastic gradient descent, is used for training the CV-CAE. However, the loss function remains in the real-domain to minimize the empirical issues



In the complex domain, the output of the convolution is computed by the convolution between the input feature map from the previous layer and the weights of the neurons, and then adding a bias as shown in the Equation below

$$
\begin{aligned}
y_i^{(l+1)} &= \sum_{C=1}^{C} w_{iC}^{(l+1)} * X_k^l + b_i^{(l+1)} \\
&= \sum_{C=1}^{C} (\Re(w_{iC}^{(l+1)}) . \Re(X_{iC}^l) - \Im(w_{iC}^{(l+1)}). \Im(X_{iC}^l)) \\
&\quad + j \sum_{C=1}^{C} (\Re(w_{iC}^{(l+1)}) . \Im(X_{iC}^l) - \Im(w_{iC}^{(l+1)}). \Re(X_{iC}^l)) \\
&\quad + b_i^{(l+1)} \qquad (2)
\end{aligned}
$$

Pooling layer in deep learning models, reduce the size of the input image by reducing the redundant information. Max-pooling and Mean-pooling are the most frequently used pooling operators in deep models. In the complex-domain, max-pooling cannot be directly defined, and as a result, mean-pooling has been employed in the paper. Up-pooling and deconvolution operations, which are used in the decoder framework to reconstruct the image, also can be defined in the same manner

In order to training the CV-CAE, a complex-valued extension of the backpropagation algorithm should be derived. For this purpose, the loss function of the reconstruction process is defined as follows

$$J(\theta) = \frac{1}{2N}\sum_{i=1}^{N}\left[\left\|\Re(Y_{ic}^{l}) - \Re(X_{ic}^{l})\right\|^{2} + \left\|\Im(Y_{ic}^{l}) - \Im(X_{ic}^{l})\right\|^{2}\right]$$

$$W_{ic}^{l} = w_{ic}^{l} - \eta\frac{\partial J(\theta)}{\partial w_{ic}^{l}}$$

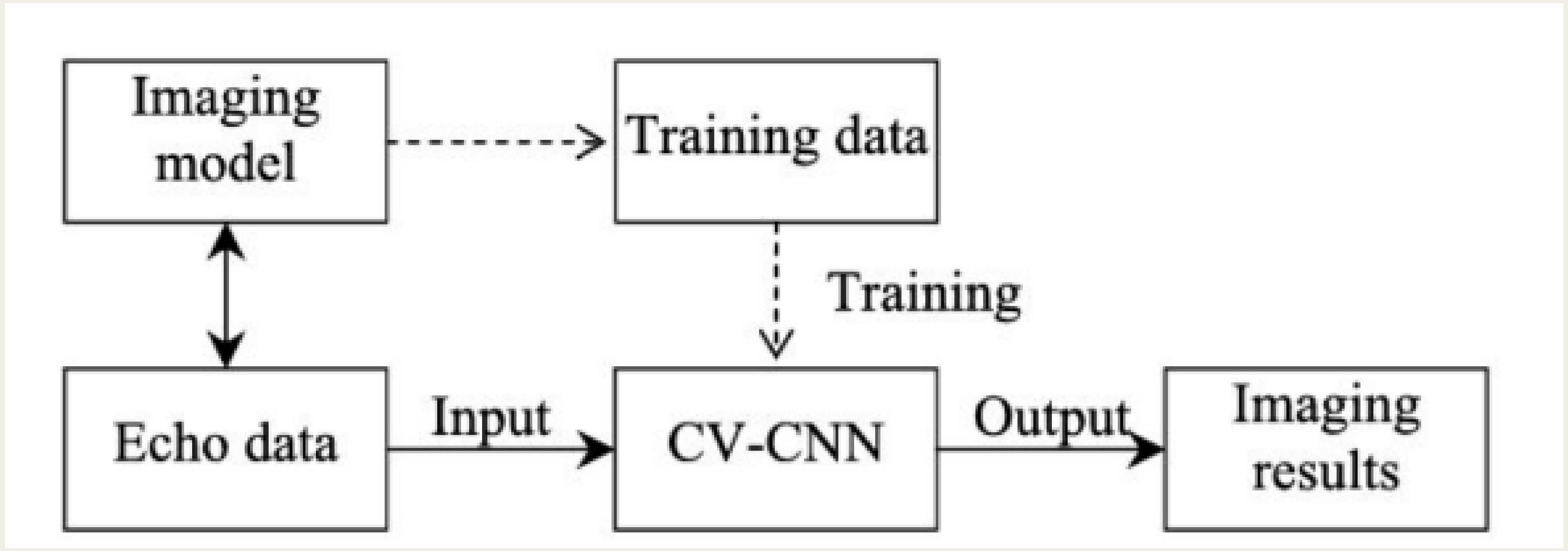
$$B_{C}^{l} = b_{C}^{l} - \eta\frac{\partial J(\theta)}{\partial b_{C}^{l}}$$

The developed CV-CAE model is trained with only 9000 samples, which is remarkably lower than the frequently used deep models. The capability of being trained with a relatively small dataset indicates the importance of the phase information. Specifically, including the phase information in the training procedure of the deep model has enabled the model to extract the necessary information and train with lower number of the samples.

The promising performance of the developed CV-CAE demonstrated the potential of the CV deep models for various CV-SAR processing applications.

TRAINING

- THE NETWORK IS TRAINED USING PAIRS OF NOISY AND CLEAN COVARIANCE MATRICES. THE CLEAN MATRICES ARE OBTAINED BY AVERAGING MULTIPLE IMAGES OVER TIME TO REDUCE NOISE.
- TO TRAIN THE NETWORK, TWO TYPES OF SUM SQUARED ERROR(SSE)-BASED LOSS FUNCTIONS ARE USED (WHICH ARE COMBINED AS L = LCOV + LNOISE):
  - LCOV: DIFFERENCE BETWEEN THE ESTIMATED AND REFERENCE CLEAN COVARIANCE MATRICES
  - LNOISE: DIFFERENCE BETWEEN THE ESTIMATED AND ACTUAL NOISY COVARIANCE MATRICES
- WE APPLY THE LCOV LOSS BETWEEN THE RECONSTRUCTED CLEAN COVARIANCE MATRIX AND THE REFERENCE TEMPORALLY AVERAGED -COVARIANCE MATRIX IN THE LOG DOMAIN.



- The CVNN model uses a Complex Rectified Linear Unit (ReLU) activation function for the inner layers

$$f = \mathrm{cReLU}(v') = \max\left(v'_{\mathrm{R}}, 0\right) + \mathrm{j}\max\left(v'_{\mathrm{I}}, 0\right)$$

- Output Layer: Produces a real-valued radar image using an “Abs” activation function that combines the real and imaginary parts
- For training, high-quality radar images are generated using a simulation approach
- Compared with its real valued counterpart, CV-CNN achieves better performance with less parameters on both real and synthetic (simulated Radar) data and in comparison to traditional non-AI methods, has higher image resolution with less noise

REFERENCES

- Complex-Valued AI in Agriculture
- Enhanced Radar Imaging Using a Complex-Valued Convolutional Neural Network
- An Approach of Sea Clutter Suppression for SAR Images by Self-Supervised Complex-Valued Deep Learning
- Despeckling Polarimetric SAR Data Using a Multi stream Complex-Valued Fully Convolutional Network
- Complex-Valued Autoencoders with Coherence Preservation for SAR
- Complex-Valued Multi-Layer Perceptrons – An Application to Polarimetric SAR Data
- Complex-Valued Neural Network (CVNN)
- COMPLEX-VALUED NEURAL NETWORKS - THEORY AND ANALYSIS

Thank you!