Complex Convolutional Neural Network and SAR classification

Miotto Sara

3102599

Contents

Chapter 1

Introduction

The aim of this project is to perform a polarimetric synthetic aperture radar terrain and land-use classification by means of a convolutional neural network.
Recently deep learning algorithms have proven to be very appropriate for classification problems. PolSAR classification has to resort to both polarization characteristics and spatial patterns. Polarization characteristics are derived from backscattered electromagnetic waves, while spatial pattern is commonly related to the shape and distribution of target and its neighbors. Besides, deep learning does not need task-specific feature extractors and it can learn features automatically from the data.
As far as CNNs are able to exploit the spatial information of an image, they outcome to be the most suitable type of neural network to fulfill this purpose. In fact a convolutional neural netork automatically learns hierarchical representations from the data and the neighborhood of each pixel is used as input.
There are other motivations behind this implementation choice but first of all it is appropriate to delve into the definition of CNNs. They are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers. Their two main features are local connectivity and weights sharing. Thanks to local connectivity neurons in a layer are only connected to a small region of the layer before it, while through sharing weight parameters across spatial positions it is possible to learn shift-invariant filter kernels and to reduce the number of parameters. This helps to avoid the over-fitting problem and to reduce the memory requirements.
The usage of this convolutional architecture is also supported by the fact that image data are often spatially correlated and the interesting features should be translational invariant. In other words, if a pattern appears in one part of the image, it could appear anywhere and the neural network has to be able to recognize it.
However, the CNN implemented here is not that straightforward. Let's see why though.
Phase information is unique to SAR images, and it is a crucial component in many SAR applications, such as target classification and recognition. In particular for POLSAR data the phase of off-diagonal elements of the covariance/coherency matrix is useful in identifying different types of scatterers.
As SAR images are complex it was necessary to extend the neural network in a complex regime. In this way a complex-valued CNN is defined. This CV-CNN allows to exploit both amplitude and phase information.
Here below, Figure2.2, is reported a visual representation of this neural network which will be more detailed explained in the following sections.

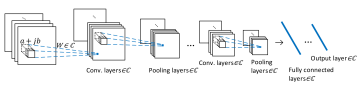


Figure 1.1: Complex CNN layout.

Although weight sharing and local connectivity help to reduce the memory problem, this is not enough: setting the neighborhoods of the pixels as the input of CNN also leads to repeated calculation and memory occupation. That is why two two ploys are put in place: sliding windows and sparse coding.
By training this neural network the scope is to learn how to classify image pixels into different classes, representing different types of land cover, such as water and terrain for whichever geographical region.
This paper has the following structure: first of all, the method used will be described as well as a

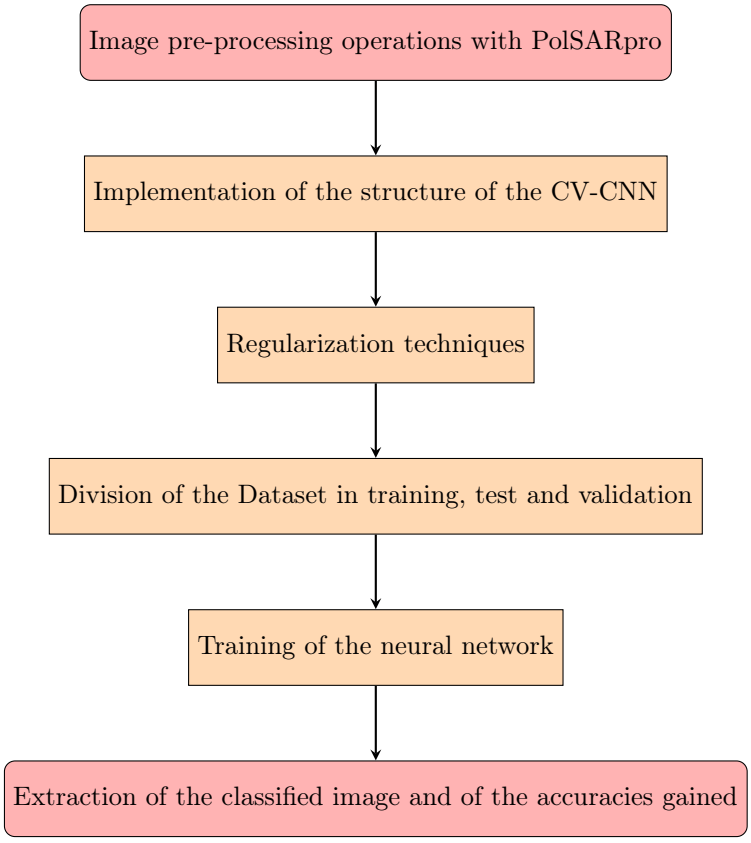
flowchart containing the different steps. This will be followed by a section with the data used for this project. Finally results will be shown and a conclusion will be provided in order to sum up everything.

Chapter 2

Method

2.1 Modus operandi

In this section it is explained the "modus-operandi" which was followed in the development of the project:



2.2 Pre-processing phase

First, it was necessary to carry out a preparation of the desired image both to make it acceptable to the neural network and to clean it up from the noise.
POLSAR images are acquired by the polarimetric radar system, and each resolution cell of the basic SLC format is expressed by a 2×2 complex scattering matrix:

$$S = \begin{bmatrix} S_H H & S_H V \\ S_V H & S_V V \end{bmatrix}$$

S_pq denotes the backscattering coefficient of p and q which are the polarization scattered and incident field respectively.
When a horizontally polarized wave is incident upon a target, the backscattered wave can have contributions in both horizontal and vertical polarizations. The same applies to a vertically polarized incident wave. The elements of the scattering matrix are complex, and can be obtained from the magnitudes and phases measured by the four channels of a polarimetric radar. The measured scattering properties of the target apply only at that frequency and radar beam angle used in the mission.
S_{HV} and S_{VH} are considered equal and so the matrix can be reduced to a 3-D scattering vector k.
Using the Pauli decomposition [29], k can be expressed as: $k = \frac{1}{\sqrt{2}}[S_{HH} + S_{VV} S_{HH} - S_{VV} 2S_{HV}]^T$
The San Francisco image has been loaded into PolSARpro software in a Single Look Complex format. Then the image was extracted in full resolution and as an output of the data, the so-called Sinclair elements were selected, with respect to covariance and coherence, to calibrate it.
Calibration is important in order to derive a more accurate geophysical product and its aim is to provide imagery in which the pixel values can be directly related to the radar backscatter of the scene.
Ainsworth calibration is the chosen one to do so. It is an a posteriori means to calibrate both cross talk between channels and imbalances in the channel gains and it imposes only the constraint of scattering reciprocity.
Next a speckle filter is applied, in this case the Sigma Lee filter. Speckle is a granular noise texture which degrades the quality of an image as a consequence of interference among wave fronts in coherent imaging systems. SO, it is not an external noise; rather, it is an inherent fluctuation in diffuse reflections, because the scatterers are not identical for each cell, and the coherent illumination wave is highly sensitive to small variations in phase changes. To have better chances of classification it is important to try to delete or at least to decrease it.
This filter is constructed on the basis of the two-sigma probabilities of the Gaussian distribution. These two probability distributions effectively suppresses the speckle noise. It also assumes that the 95.5% of pixels are located within the two-sigma range from its mean which is unknown and needs to be estimated. The reduction of speckle is achieved through the replacement of the center pixel of a scanning window with the average of those pixels within the two-sigma range of the center pixel. All the other pixels are considered as outliers, and they are not included in computing the sample mean.
One last pre-processing step is the multilooking one. Multilooking averages the image, supressing speckle and creating a more usable geometry.

2.3 Layout of CV-CNN

After the pre-processing operations that have been applied to the image, the data result in ML format and expressed in the 3 × 3 Pauli-based polarimetric coherency matrix $[T]$. This coherency matrix is essentially an alternative representation of a full polarimetric dataset allowing the analysis of incoherent targets and the phenomenon of depolarisation (transformation of incoming fully polarised wave into a partially polarised wave by creating a variety of different types of polarisation during media interaction). The coherency matrix of PolSAR data for the multi-look case is obtained as:

$$T = \frac{1}{L} \sum k_i k_i^H = \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix}$$

where L is the number of looks. The superscript T denotes the transpose and H denotes the conjugate transpose. Apparently, the coherency matrix T is a Hermitian matrix whose diagonal elements are real number, while off-diagonal elements are complex.
Each pixel is represented by a local patch defined by a window of size $m1m2$. It thus captures not only the polarimetric characteristics but also the spatial imaging pattern surrounding the center pixel, and the CV-CNN extracts these features. This process generates a significant amount of patches which are used as the training and validation samples.
This matrix had to be transformed into a vector in order to be able to feed into the convolutional neural network. Hence, the upper triangular $T_{11}, T_{12}, T_{13}, T_{22}, T_{23}, T_{33}$ of $[T]$ matrix was adopted as the input data.

2.4 Complex-value Convolutional Neural Network

The CV-CNN shares the same features of a normal CNN: local connectivity, weight sharing, pooling and cascaded layers.

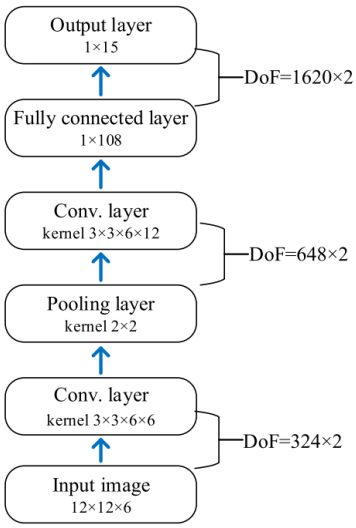


Figure 2.1: Complex CNN layout.

Observing the above figure it is possible to dig deeper into its internal layout. It consists of two convolutional layers, one pooling layer and one fully connected layer.
The input layer has a size of 12 × 12 × 6, and so 12 × 12 is the size of the patch and 6 the number of channels. The size of feature maps will be shrunk due to convolution and pooling operations.
In order to have certain depth of network, it needs zero padding on each dimension of the input layer if the size of input data is less than 12 × 12. The procedure of padding is referred to the amount of pixels added to an image when it is being processed by the kernel of a CNN. If the padding in a CNN is set to zero, as in this case, then every pixel value that is added will be of value zero. Basically it extends the area of which kernels of a neural network scan an image.
In convolution layers the hidden units are complex values and are connected to local patches within the feature maps, complex as well, of the previous layer through kernels. The units in a local patch are convolved by the weight matrix, and then passed through the non linearity activation function, in this case a sigmoid. The main reason of implying a sigmoid in this context is that it exists between (0 to 1) and therefore it is particularly suited for models where the output is a probability. They receive inputs from a set of units located in a small neighborhood from the previous layer, which is called local receptive field. The input image is filtered by six convolution filters of size 3 × 3 × 6 with stride 1 in the first convolutional layer, resulting in six feature maps of size 10 × 10.
Complex convolution works the same way as normal convolution, so the complex output feature maps $O_k^{(l+1)} \in \mathbb{C}^{(W_2 H_2 I)}$ are computed by the convolution between all the previous layer's input feature maps $O_k^l \in \mathbb{C}^{(W_1 H_1 I)}$ and a bank of filters $w_i k^{(l+1)} \in \mathbb{C}^{(FxFxKxI)}$ and add a bias $b_i^{(l+1)} \in \mathbb{C}^I$.
Then the formula of complex convolution appears as:

$$O_i^{(l+1)} = f(\Re(V_i^{l+1})) + jf(\Im(V_i^{(l+1)}) = \frac{1}{1+e^{-\Re(V_i^{(l+1)})}} + j\frac{1}{1+e^{-\Im(V_i^{(l+1)})}}$$

$$V_i =^{(l+1)}= \sum w_i k^{(l+1)} * O_k^(l) + b_i^{(l+1)} =$$

$$= \sum(\Re(w_i k^{(l+1)}) x \Re(O_k^{(l)}) - \Im(w_i k^{(l+1)}) x \Im(O_k^{(l)}) + j \sum(\Re(w_i k^{(l+1)} x \Im(O_k^{(l)}) + \Im(w_i k^{(l+1)}) x \Re(O_k^{(l)}) + b_i^{(l+1)}$$

$O_k^{(l)}$ is the unit of the kth input feature map in layer l and $V_i^{(l+1)}$ denotes the weighted sum of inputs to the ith output feature map in layer $(l+1)$.
I is the number of feature maps, $FxFxK$ is the filter size, S the stride and P the zero-padding size. The zero-padding process is used in order to shrink the size. Here the filter size and stride are equal respectively to 3×3 and 1.
On the other hand the pooling layer aims at merging semantically similar features that are detected by the convolutional layer. Essentially these layers can be considered as subsampling layers which reduce the dimension of features as well as make the representation invariant to small shifts and distortions of the input. The pooling size is equal to 2×2 and as a stride of 1, consequently their size becomes 5×5. Here we use the average pooling and we extend it to the complex domain:

$$O_i^{(l+1)}(x,y) = aveO_i^{(l)}(xs+u, ys+v)$$

g denotes pooling size and s the stride. $O_i^{(l+1)}(x,y)$ is the unit of the ith input feature map at position (x,y).
In fully connected layers each neuron is connected to all the neurons of the previous layer. They are made of 108 neurons and are located at the top of the CV-CNN.
The output of these layers translated in the complex domain can be written as:

$$O_i^{(l+1)} = f(\Re(V_i^{l+1})) + jf(\Im(V_i^{(l+1)})$$

$$V_i =^{(l+1)}= \sum w_i k^{(l+1)} xO_k^(l) + b_i^{(l+1)}$$

As regards the output layer it is a $1 * c$ softmax classifier and it has the purpose of predicting the classification of the input sample, with c equal to the number of classes, two in this case. The label of the class is a one-hot encoding vector.
In order to prevent the problem of overfitting some regularization three techniques have been implemented: L1 regularization, batch normalization and early stopping.
First of all, L1-regularization, or sparse weights, was applied to convolutional layers. This technique aims at limiting the model capacity by adding a parameter norm penalty to the objective function: $\hat{L}(\theta) = L(\theta) + \lambda\Omega(\theta)$ that in L1 assumes the following form:
$\hat{L}(\theta) = \sum L(f(x_i,\theta), y_i) + \frac{L}{2}\sum ||w_l||$.
It passes inside the gradient descent update rule: $w_l^{(t+1)} = w_l^(t) - \epsilon_t\lambda\frac{w_l^{(t)}}{|w_l^{(t)}} - \epsilon_t\Delta_w L$, meaning that as λ grows more weights become 0.
Secondly, the batch normalization is a method to reparameterize a deep network. In fact, as learning progresses, the distribution of layer inputs changes due to parameter updates. This phenomenon is known as internal covariate shift and can result in most inputs being in a nonlinear regime and slow down learning. Batch normalization, by calculating the mean and the standard deviation of all the activations of a layer, can solve this.
Lastly, early stopping means starting with small weights and stopping the learning before it overfits. As stopping criterion a monitor on the accuracy has been set.
The dataset was split into training, validation and test set with the following respective proportions: $90\%, 2\% and 8\%$.
To better operate with the dataset, a batch size of 100 was taken into account.
The two tricks mentioned in the previous section are then applied to try to compensate for memory problems.
PolSAR image classification is actually a dense prediction problem, and so Fully Connected Neural Networks do have great potential in PolSAR image classification because of their network architecture. However, they cannot be applied directly to the classification problem, yet, based on it, it is possible to utilize a sliding window operation to tackle the problem.
In this way repeated calculations are avoided resulting in a better way of dealing with memory occupation.

Later, through sparse coding, the PolSAR image can be down-sampled to a smaller size.
In the figure below is reported how a CV-CNN with these two tricks should look like:

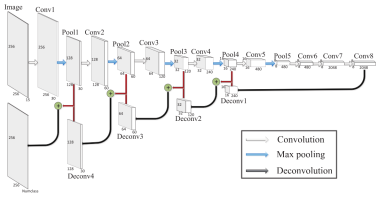


Figure 2.2: CV-CNN with sliding windows and sparse coding.

2.5 Training

It is now possible to train this neural network.
The optimizer chosen is the stochastic gradient descent and it is used in combination with the backpropagation algorithm translated in the complex domain. It is a method for computing gradients, the vector of partial derivatives with reference to all the coordinates of the weights. Each partial derivative measures how fast the loss changes in one direction. Gradient descent aims at finding the model parameters that correspond to the best fit between predicted and actual outputs. Weights are updated following this formula: $w = w - learning_rate * g$, where the learning rate has been set to 0.01.
SGD refers to estimating the gradient using only a subset of training samples, in this case we use a mini-batch made of 100 samples. Therefore we are dealing with a stochastic sampling process.
As loss function the least-squares loss function is the one implied. The loss function is the function that is minimized in the process of fitting a model, and it represents a selected measure of the discrepancy between the observed data and data predicted by the fitted function. Here it is developed as the sum of squared deviations from the fitted line or plane. One of the properties is that it is very sensitive to the outliers.
The total classification error is given by:

$$E = \frac{1}{2}\frac{1}{N}\sum\sum[\Re(T_K[n]) - \Re(O_k[n])^2 + (\Im(T_k[n]) - \Im(O_k[n]))^2]$$

The minimum of the above loss function is searched by iteratively adjusting the weights according to:

$$w_i k^{(l+1)}[t+1] = w_i k^{(l+1)}[t] + \Delta w_i k^{(l+1)}[t] =$$

$$= w_i k^{(l+1)}[t] - \eta \frac{\partial E[t]}{\Delta w_i k^{(l+1)}[t]}$$

$$b_i^{(l+1)}[t+1] = b_i^{(l+1)}[t] + \Delta b_i^{(l+1)}[t] =$$

$$= b_i^{(l+1)}[t] - \eta \frac{\partial E[t]}{\Delta b_i^{(l+1)}[t]}$$

The derivatives of complex functions are obtained according to the complex chain rule. The key point is computing the error gradient of weights:

$$\frac{\partial E}{\partial w_i k^{(l+1)}} = \frac{\partial E}{\partial \Re(w_i k^{(l+1)}} + \frac{\partial E}{\partial \Im(w_i k^{(l+1)})}$$

$$= \left(\frac{\partial E}{\partial \Re(V_i^{(l+1)})} * \frac{\Re(V_i^{(l+1)})}{\partial \Re(w_i k^{(l+1)})} + \frac{\partial E}{\partial \Im(V_i^{(l+1)})} * \frac{\partial \Im(V_i^{(l+1)})}{\partial \Re(w_i k^{(l+1)})}\right)+$$

$$\mathrm{j}\left(\frac{\partial E}{\partial \Re(V_i^{(l+1)})} \frac{\partial \Re(V_i^{(l+1)})}{\partial \Im(w_i k^{(l+1)})} + \frac{\partial E}{\partial \Im(V_i^{(l+1)})} \frac{\partial \Im(V_i^{(l+1)})}{\partial \Im(w_i k^{l+1})}\right)$$

By continuously reducing the above error, the parameters are updated until the error reaches a minimum.

2.6 Classification

After having outlined the layout of the neural network, it is possible to move on to a description of how the classification actually takes place.
During the training phase the CV-CNN receives as input the labelled image constructed with the aim of PolSAR. For simplicity only two classes have been selected manually: water (blue) and terrain (and so all the different classes that were not water were forced into the brown color). Using a special python function, the labels were transformed into RGB format. It makes its predictions by calculating the distance between each element in the output vector and the location number of the element with the smallest distance is the required category.

Chapter 3

Data description

An image depicting an area of San Francisco was used for the purpose of this project. Its size is 900×1024 pixels and the spatial resolution is about 10×10 m.
This image was collected by Airborne Synthetic Aperture Radar. AIRSAR served NASA radar technology testbed as an imaging tool which is able to collect data both during days and nights and to penetrate clouds and if with sufficient wavelengths also forest canopy, sand cover and dry snow pack.
For simplicity, with the aim of PolSAR, only two classes have been selected but of course this could be extended to a more precise classification. So, everything was divided between water and terrain.
The main features of this SAR image are:

Frequency/wavelenght	1.26 GHz/23 cm
Polarization	Full
Range Resolution	3.75 m
Swath Width (nominal)	10 km
Off-Nadir Angle (normal)	20-60

In order to deal with this dataset I have first of all imported the image into PolSARpro and extracted it in full resolution. After this conversation a Pauli RGB figure is created Figure3.1:

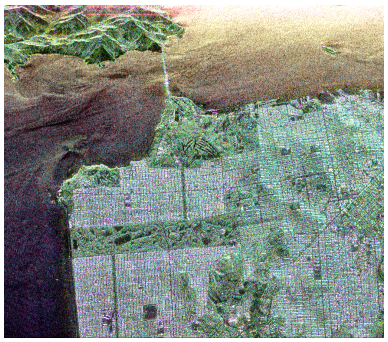


Figure 3.1: PauliRGB.

The Pauli color coding is based on a vector representation of linear combinations of scattering matrix elements. Colors are then associated to the various channels, specifically HH+VV to blue, HH-VV to red and HV to green.
In built up areas the dominant colors are white and red. White pixels correspond to equal amplitude over all polarimetric channels, whereas red indicates that the phase argument of HHVV* is close to π, and denote a wave double bounce reflection.

Chapter 4

Results

The model was trained for 1000 epochs, following which, the accuracy values shown in the table below were obtained. A confusion matrix was used for this scope. It is essentially a tool for analyzing the errors made by a machine learning model and is useful for assessing the quality of the classification model's predictions. Specifically, the matrix highlights where the model errs, in which instances it responds worse and which instances it responds better. It is made of four elements: the output "TN" stands for True Negative which shows the number of negative examples classified accurately. Similarly, "TP" stands for True Positive which indicates the number of positive examples classified accurately. The term "FP" shows False Positive value that is the number of actual negative examples classified as positive; and finally "FN" means a False Negative value which is the number of actual positive examples classified as negative. The accuracy of a model is then calculated as: $Accuracy = \frac{TN+TP}{TN+FP+FN+TP}$.
Remember that by accuracy one intends the best one achieved among the various epochs, by average accuracy the average value of this in the thousand iterations, while loss is defined as the values indicating the difference from the desired target states.

	Train	Validation	Test
Loss	0.027	0.058	0.058
Accuracy	0.962	0.961	0.960
Average Accuracy	0.880	0.901	0.896

Chapter 5

Conclusions

CNN has achieved great success in computer vision areas. This study aims to apply CNN to SAR image processing. In order to take advantages of phase information, we use here a complex version of this neural network in which both neurons and weights are represented by complex numbers.
A complex version of backpropagation is also proposed, and through it we can effectively train deep CV-CNN with massive complex SAR images.
A novel POLSAR image classification scheme based on CV-CNN is then proposed and evaluated with real SAR data. Results show that a significant improvement in terms of classification accuracy can be actually achieved.

Chapter 6

References

Bibliography

@article [1] , author = Zhimian Zhang and Haipeng Wang and Senior Member and IEEE and Feng, title=Complex-Valued Convolutional Neural Network and Its Application in Polarimetric SAR Image Classification journal = Journal of Sketchy Physics, year = 2003