National Institute of Applied Sciences and Technology

CARTHAGE UNIVERSITY

End of year project

Major : **Software Engineering (GL)**

Change Detection in Hyperspectral Images Using Recurrent 3D Fully Convolutional Networks

Realized by

Omar Kamoun

Mohamed Helmi Boudhraa

Mohamed Khalil Ben Salah

Supervisor : **Mrs. Bouzidi Sonia**

Academic year : 2019/2020

Contents

Figures List

Tables list

General Introduction

Background of the study:

Due to the successive breakthroughs in Hyperspectral Remote Sensing, state-of-the-art hyperspectral remote sensors are now capable of producing high resolution spectral images.
A Hyperspectral picture can give more data of picture pixel, in fact, one hyperspectral image at various wavelength bands describes nearly a thousand kilometers of geographical area making HSIs widely used to detect changes on the surface covered by the image due to natural disasters, deforestation, change in coarse of the river, urbanization, etc...

Objective:

The main objective of this project is to exploit several Deep Learning techniques to detect changes on HSIs to be utilized as a part of distinctive applications like Disaster checking of violent winds, snow slides, woodland fires, environment observing, and military target observing and outskirt observation, etc..

Chapter I

Business Understanding

Introduction

This chapter is a general presentation of the project. It poses the detailed problem of the project by introducing the state-of-the-art existing change detection methodologies. It also describes the adopted methodology during the implementation of the project.

1 Project description

This project is a supervised change-detector using several deep learning techniques. The model to be developed will detect the changes on the surface using two HSIs taken at different times. Ground truths of the corresponding HSIs will be used to train, validate and test the model.

2 Detailed problem

If the change detection of the earth's surface is done timely and accurately then the relationship and interaction between natural phenomena and humans can be better analyzed and understood as a result of which better management and use of resources can be done. Change detection involves the application of multi-temporal data sets to quantitatively analyze the changes of land cover classes. The change detection can be done by traditional methods and by using remote sensing technologies. The traditional methods, like using **CNN** for acting mainly as a spatial–spectral feature extractor in a change detection framework, are expensive, time consuming and not so accurate whereas all these problems do not infer with remote sensing technology.

Although deep-learning approaches produce superior results and take the advantage of automatically learning the high-level semantic features from raw data in a hierarchical manner these approaches require training samples to train the network as the methods provide excellent performance when the number of training samples is sufficient. Supervised deep-learning methods, such as **CNN** and **RNN**, require training samples to train the network. These samples have high probabilities of being either changed or unchanged; thus, the accuracy of the samples affects the CD results. To solve these problems, the present study proposes a sample generation method and a novel change detection network known as the recurrent 3D fully convolutional network (**Re3FCN**), which is a fully convolutional network (**FCN**) that includes 3D convolutional layers and a convolutional LSTM (**ConvLSTM**)[3].

3 State of the art

To address these problems, several approaches have been proposed to achieve beneficial hyperspectral CD applications. Image difference and ratio are widely approaches used to solve such problems; however, these approaches are characterized by limited applications . Generally, traditional CD methods for HSIs can be categorized as: Image algebra, transformation-based methods, spectral analysis, and post-classification.

3.1 Image algebra

A classical method in image algebra is a spectral change vector analysis (CVA), which calculates the magnitudes and directions of changes. Using the two variables (i.e., magnitude and direction), different types of change can be detected. Sequential spectral CVA (S2CVA) was proposed for the hyperspectral CD to overcome the problems associated with the original CVA. S2CVA is implemented iteratively, with multiple change information models and it is identified hierarchically in each iteration. However, CVA-based methods have some disadvantages, such as difficulties in identifying multi-class changes and in selecting an appropriate threshold.

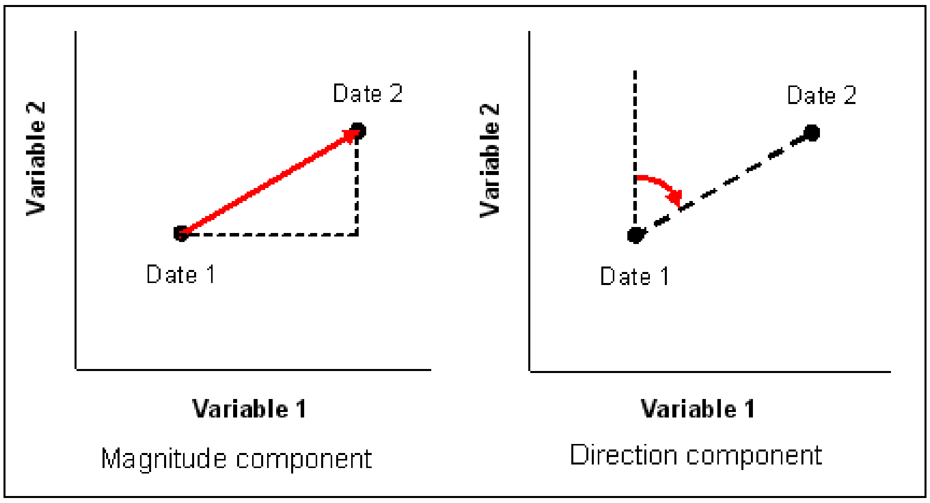


Figure I.1 – Illustration for a two-dimensional variables space describing the change of a feature between two time limits.

3.2 Transformation-based methods

Transformation-based methods can transform HSIs into other feature spaces to distinguish changes from non-changes, these methods assist in producing multivariate components based on the first few components. Principal component analysis (PCA) exploits the variance in the principal components (PCs) of the combined multi-temporal HSI . Multivariate alteration detection (MAD) is based on the canonical correlation analysis, which investigates the linear combinations of the original variables. By assigning higher weights to non-change features, iterative reweighted (IR) MAD generates a non-change background to detect changes. Transformation-based methods are advantageous in reducing both dimensionality and noise, with the ability to emphasize the changed or unchanged features related to specific changes. However, if the changes comprise a large portion of the images, it will be relatively time-consuming to generate new components .

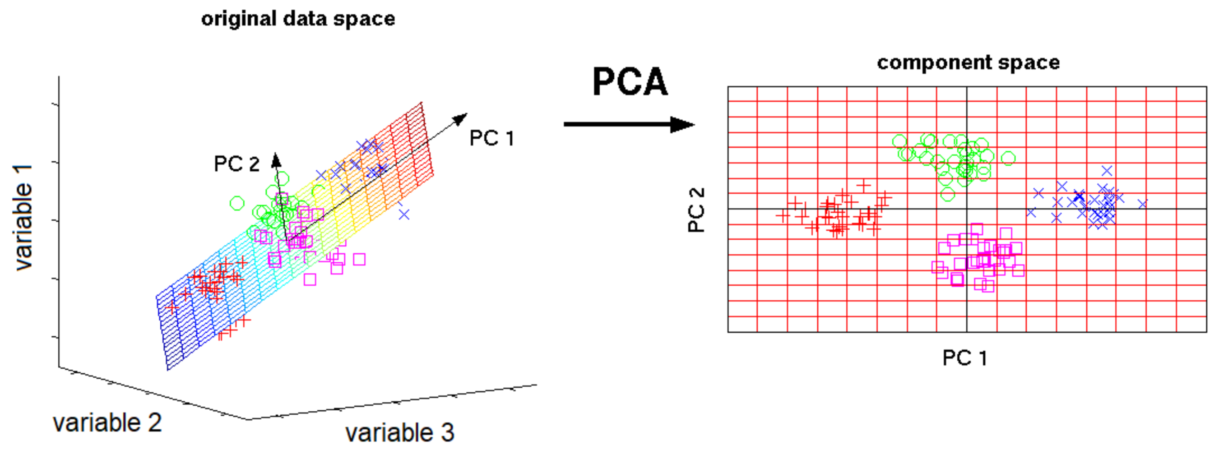


Figure I.2 – Reducing the data space from 3 variables to 2 using PCA[1].

3.3 Spectral analysis

Spectral analysis detects differences in spectral distance or shape in all bands between two pixels acquired at two different times. This analysis is used to construct anomaly-detection algorithms to distinguish unusual pixels from background pixels. Generally, Euclidean Distance (ED), Spectral Angle Mapper (SAM), Spectral Correlation Measure (SCM), and Spectral Information Divergence (SID) are widely used to measure differences between spectral signatures. Moreover, the subspace distances between temporal HSIs were calculated using orthogonal subspace analysis, which assisted in detecting the background anomalous pixels. These models are effective in solving shadow problems and noise effects; however, they are relatively complex and are difficult in determining an appropriate threshold for distinguishing between changes and non-changes.

3.4 Post-classification methods

Post-classification methods are used to compare the land-cover classification results after the multi-temporal images have been classified independently using a specific classifier. These methods are widely used and provide “from-to” CD information. A fuzzy c-means (FCM) classifier is generally used in a clustering algorithm that calculates the degree of uncertainty in each class and expresses the property of the class membership . The advantage of that particular method is its ability to generate a change map obtained from two different sensors. Nevertheless, the CD accuracy depends on the classification accuracy .

4 Methodology used

This project used the CROSS Industry Standard Process for Data-Mining (CRISP-DM) methodological approach.

This is the methodology used in the majority of Data-Science projects, created in the 1960s by IBM to organize mainly its Data-Mining projects. And until today it presents the only effective method to manage and carry out these types of projects. This approach breaks down the completion of any Data-Science project into 6 stages in an iterative way, passing through each stage several times.

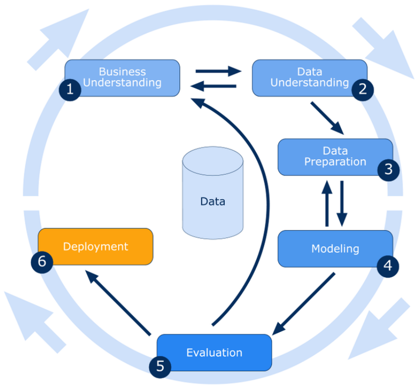


Figure I.3 – The different stages of the CRISP-DM methodology

The steps of this methodology are :

- **Business Model** This step consists of studying the business area of the project as well as the context of the problem to be solved. The objective of this study is to clarify the exact need of the project, and to define a plan for the project, taking into account the costs, risks and benefits of carrying out this project.
- **Data Preparation** This phase consists of extracting and constructing the precise set of data to be used from the brutes. It begins by cleaning raw data, defining useful information and classifying it according to its importance by pre-selected criteria. And finally do a recoding so that they can be used by the appropriate tools and algorithms.
- **Modeling** This is the phase where our change detection model will be well defined. During this phase we choose algorithms, mathematical models and relationships between them. After choosing the techniques to use, the configuration of these models is achieved by choosing the parameters that will lead to optimal results. It is during this phase that all data is divided into two subsets: learning data and evaluation data.
- **Evaluation** During this phase, the model is evaluated in order to ensure its performance, precision and robustness before deploying it.
- **Deployment** This is the final stage of the project, it is the production of the models built for the end users of the solution.

Conclusion

In this first chapter, the project was set in context. The working methodology was also presented. The next chapter will focus on understanding and preparing data.

Chapter II

Data Presentation and Preparation

Introduction

In this chapter we study the raw data by analyzing its structure before starting to process, clean and format it in order to extract the training and evaluation data.

1 General Description of Data

This project is a Hyperspectral images change detector, meaning that our data will consist of a set of pairs of HSIs. Each pair will contain two images of the same surface at different times.
A 3D HSI contains data captured by a hyperspectral sensor.
image is coded as a three-dimensional matrix, the first dimension is associated to the frequencies, each band is associated with an acquisition frequency.
The other two dimensions are spatials for the positions (X, Y). Each value of the matrix is the frequentiel response of the portion of the surface covered by the image at (X, Y) to a frequency F.

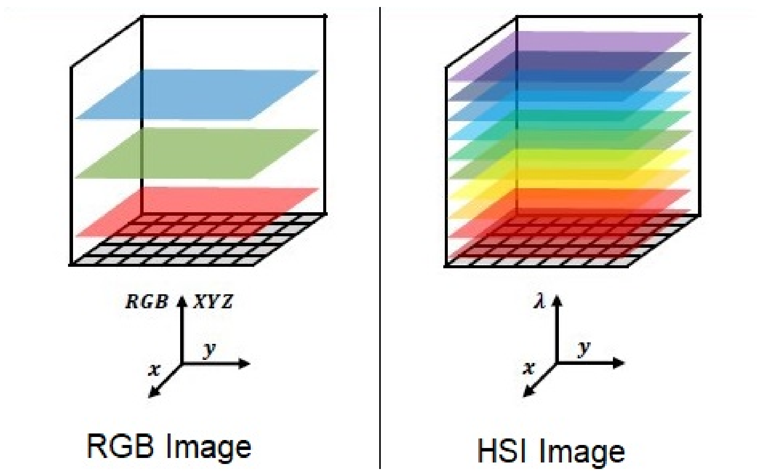


Figure II.1 – RGB vs HSI images.

2 Data Presentation

We are going to work on well-known HSIs widely used by the scientific community interested in Hyperspectral images studies. These images were used in previous change detection models and can be considered as a benchmark to train and evaluate the accuracy of our model.

2.1 Indian Pines' Hyperspectral Images

[4] This scene was gathered by AVIRIS sensor over the Indian Pines test site in North-western Indiana and consists of 145x145 pixels and 224 spectral reflectance bands in the wavelength range 0.4–2.5 10^{-6} meters. This scene is a subset of a larger one. The Indian Pines scene contains two-thirds agriculture, and one-third forest or other natural perennial vegetation. There are two major dual lane highways, a rail line, as well as some low density housing, other built structures, and smaller roads. Since the scene is taken in June some of the crops present, corn, soybeans, are in early stages of growth with less than 5% coverage.
The ground truth available is designated into 13 classes and is not all mutually exclusive.

	Class	Samples
1	Alfalfa	46
2	Corn-notill	1428
3	Corn-mintill	830
4	Corn	237
5	Grass-pasture	483
6	Grass-trees	730
7	Grass-pasture-mowed	28
8	Hay-windrowed	478
9	Oats	20
10	Soybean-notill	972
11	Soybean-mintill	2455
12	Soybean-clean	593
13	Woods	1265

Tableau II.1 – Ground truth classes for the Indian Pines scene and their respective samples number.

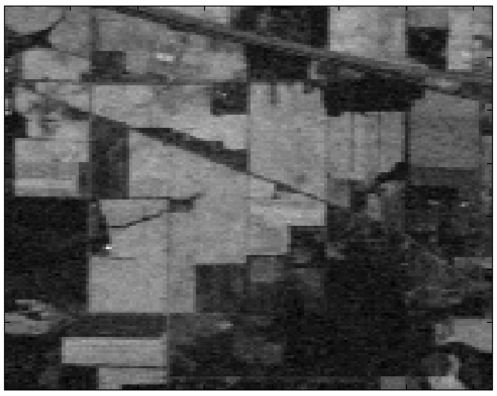


Figure II.2 – Sample band of Indian Pines dataset

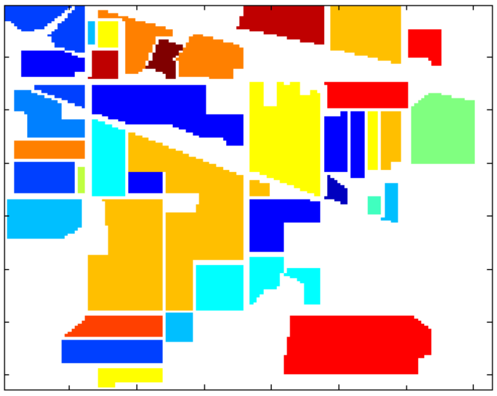


Figure II.3 – Ground truth of Indian Pines dataset

2.2 Salinas' Hyperspectral Images

This scene was collected by the 224-band AVIRIS sensor over Salinas Valley, California, and is characterized by high spatial resolution (3.7-meter pixels). The area covered comprises 512x217. This image was available only as at-sensor radiance data. It includes vegetables, bare soils, and vineyard fields. Salinas ground truth contains 13 classes.

	Class	Samples
1	Brocoli_green_weeds_1	2009
2	Brocoli_green_weeds_2	3726
3	Fallow	1976
4	Fallow_rough_plow	1394
5	Fallow_smooth	2678
6	Stubble	3959
7	Celery	3579
8	Grapes_untrained	11271
9	Soil_vinyard_develop	6203
10	Corn_senesced_green_weeds	3278
11	Vinyard_untrained	7268
12	Vinyard_vertical_trellis	1807
13	Lettuce_romaine	4981

Tableau II.2 – Ground truth classes for the Salinas scene and their respective samples number.

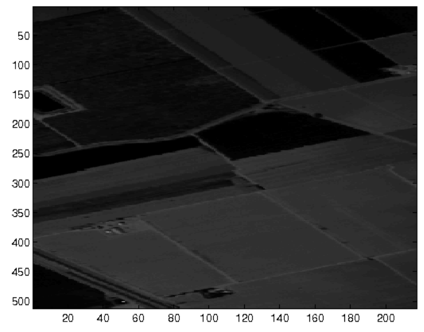


Figure II.4 – Sample band of Salinas dataset

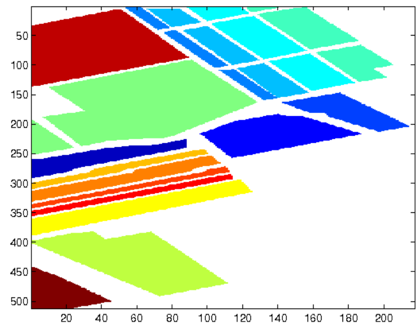


Figure II.5 – Ground truth of Salinas dataset

3 Data Preparation

3.1 Image Synthesis

To be able to obtain two HSIs of the same scene it is possible to synthesise a new altered image from the original one.
This is done by creating a pixel mask denoting the pixels that kept the same frequential response and the ones that changed their frequential response (changed their class).
Later on we use this mask to generate a new image and ground truth based on the old one.

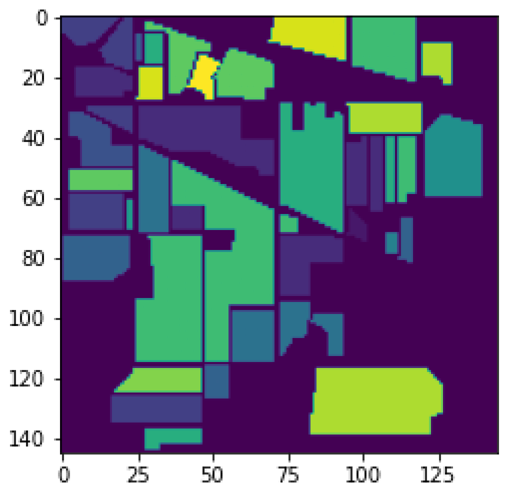


Figure II.6 – Original ground truth of Indian Pines dataset

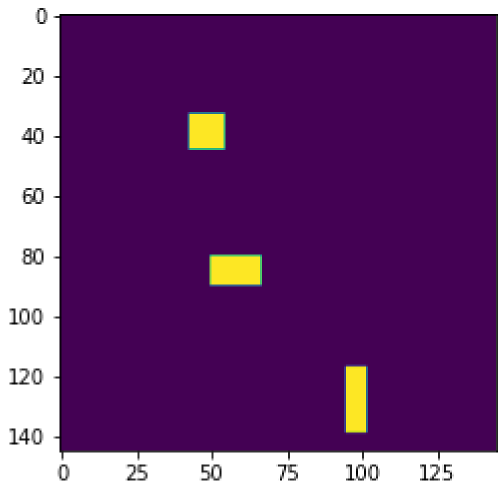


Figure II.7 – Replacement Mask.

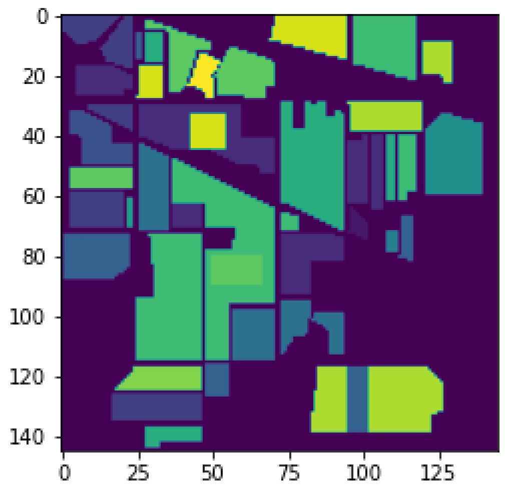


Figure II.8 – Ground truth of Synthesized Indian Pines image

3.2 Image Normalization

Image Normalization is the change of the range of values of pixels. It's the application of the same arithmetic operation on all pixels. For a 3D HSI each band is associated to its own arithmetic operation. There is two wiedly-used methods of image normalization:

- Histogram equalization; a method in image processing of contrast adjustment using the image's histogram.

$$I'(x,y) = \frac{I(x,y) - Imin}{Imax - Imin} \tag{II.1}$$

  But this method may give erroneous values due to noise, for example, if the maximal value is a noisy pixel.

- Standardization; is a data scaling technique that assumes that the distribution of the data is Gaussian and shifts the distribution of the data to have a mean of zero and a standard deviation of one.

$$I'(x,y) = \frac{I(x,y) - Avg}{StandardDeviation} \tag{II.2}$$

We are going to choose the second method. For each band, we will calculate the Average and the Standard Deviation of its values then we will apply the transformation on every pixel in the band.

3.3 Data Augmentation

3.3.1 Patches Generation

Operating on small size images is easier than operating on the entire image itself. We are going to split each image to several smaller sized rectangular patches, operate individually on each of these patches, and finally tile all these patches at their respective locations.
Working on small patches have two main benefits:

- Reduce Memory Usage (much smaller matrix sizes, less parameters, less computations).
- Improving model capability to identify finer details in one image patch.

3.3.2 Random Flips / Rotations

A widely-used method in data augmentation in images-related machine learning problems is to perform random flips/rotations on the data.
But we have to be careful and not generate large number of data using this method to avoid over-fitting problems later on.

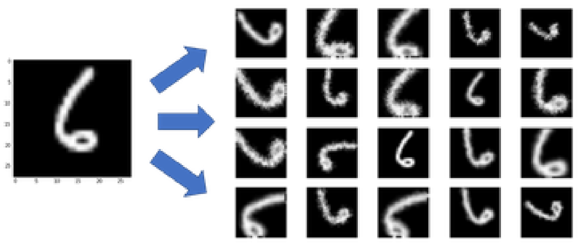


Figure II.9 – Example of data augmentation using random flips / rotations.

3.4 Oversampling weak classes

Indian Pines dataset have large differences in the number of samples of different classes, this may cause our model to perform poorly on new data.
We will use the aforementioned data augmentation techniques to over-sample weak classes by making the labels count difference between classes as small as possible.

3.5 One-Hot Encoding

One-Hot encoding transforms the pixels' classes to binary numbers having only 1 ON bit and the rest arek, 0. For example if we have 5 classes our One-Hot encoding will be the following:

Class	Encoding
Class 1	00001
Class 2	00010
Class 3	00100
Class 4	01000
Class 5	10000

Tableau II.3 – One-Hot Encoding Example.

3.6 Splitting the data into Train, Validation and Test datasets

Training Dataset ($\approx$ 70%)

The actual dataset that we use to train the model. The model sees and learns from this data.

Validation Dataset ($\approx$ 15%)

The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while **tuning model hyper-parameters**. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration.

Test Dataset ($\approx$ 15%)

The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset and optimized on the validation dataset. The Test dataset provides the gold standard used to evaluate the model. It is only used once the model is completely trained

Conclusion

In this chapter, we introduced the structure of raw data that we will process and use in training and evaluating our model. The next chapter will focus on the model conception and training.

Chapter III

Model Conception and Training

Introduction

The different concepts presented make it possible to highlight the context of our work. The goal is to have the ability to detect the change in Hyperspectral Images by exploiting deep learning.
In this chapter, we explain the process we took to achieve this goal.

1 Proposed Solution

Change detection in Hyperspectral Images (HSI) is widely used for the analysis of remote sensing images. The convolutional neuron network (CNN) is one of the most commonly used methods for data processing. The present study proposes a different approach using a sample generation method and a CD network known as the recurrent 3D fully convolutional network (Re3FCN), which is a fully convolutional network (FCN) that includes 3D convolutional layers and a convolutional LSTM (ConvLSTM).

1.1 Convolutional Neural Network (CNN)

Neural Networks is one of the most popular machine learning algorithms at present. It has been decisively proven over time that neural networks outperform other algorithms in accuracy and speed. With various variants like CNN (Convolutional Neural Networks), RNN(Recurrent Neural Networks), AutoEncoders, Deep Learning etc.

1.1.1 What is a Neuron?

As the name suggests, neural networks were inspired by the neural architecture of a human brain, and like in a human brain the basic building block is called a Neuron. Its functionality is similar to a human neuron, i.e. it takes in some inputs and fires an output. In purely mathematical terms, a neuron in the machine learning world is a placeholder for a mathematical function, and its only job is to provide an output by applying the function on the inputs provided.

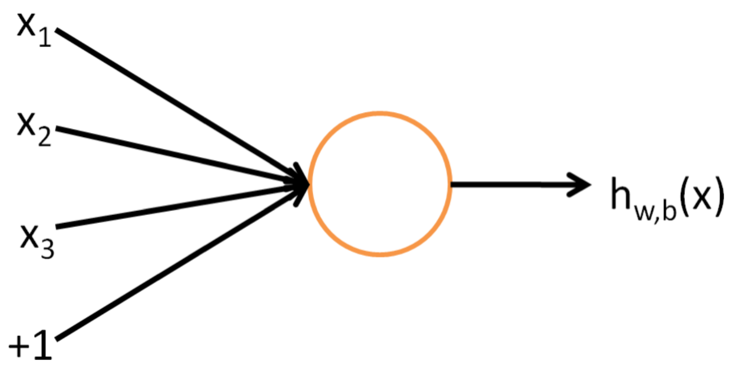


Figure III.1 – Illustration of a Neuron.

The function used in a neuron is generally termed as an **activation function**. There have been 5 major activation functions tried to date, step, sigmoid, tanh, ReLU and leaky ReLU. In this project, we are only interested in ReLU function.

ReLU Function

The Rectified Linear Unit is the most commonly used activation function in deep learning models. The function returns 0 if it receives any negative input, but for any positive value x, it returns that value back. So, it can be written as **f(x)= max(0, x)**.

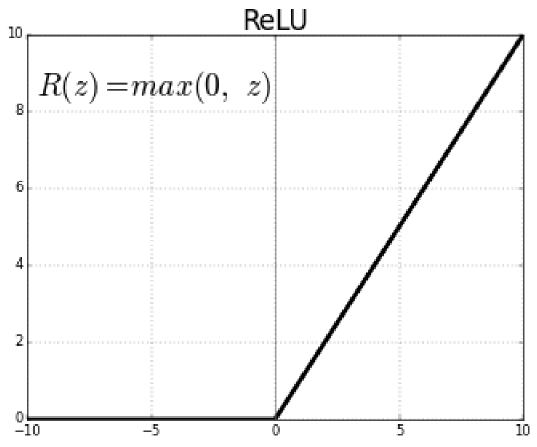


Figure III.2 – ReLU Function.

The **Leaky ReLU** (a variation of classic ReLU) is one of the most well-known. It is the same as ReLU for positive numbers. But instead of being 0 for all negative values, it has a constant slope (less than 1).

1.1.2 What is a Neural Network?

A neural network is consisted of multiple **layers** stacked sequentially. A layer is nothing but a collection of neurons which take in an input and provide an output. Inputs to each of these neurons are processed through the activation functions assigned to the neurons. For example, here is a small neural network.

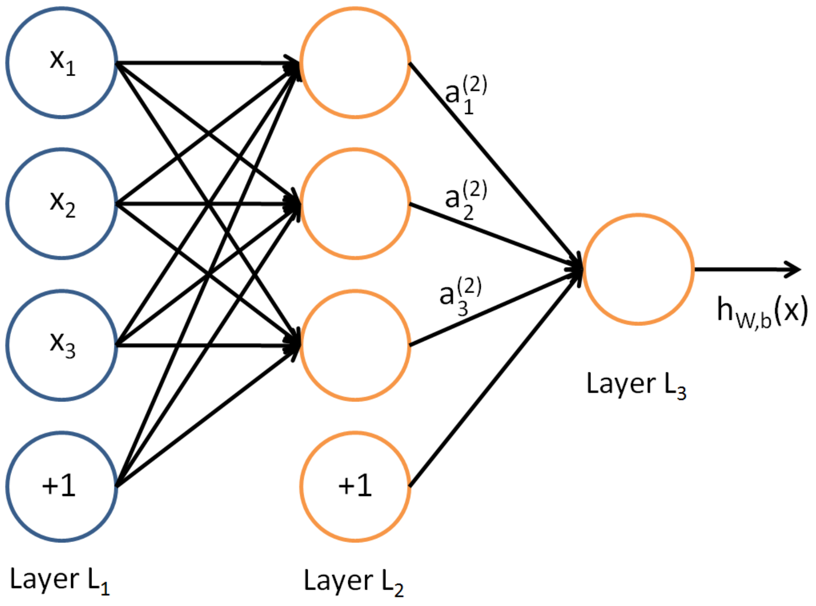


Figure III.3 – Small Neural Netowrk Example.

Any neural network has 1 input and 1 output layer. The number of hidden layers, for instance, differ between different networks depending upon the complexity of the problem to be solved.
A neural network having more than one hidden layer is generally referred to as a **Deep Neural Network**.

1.1.3 What is a Convolution?

A convolution operates on two signals (in 1D) or two images (in 2D or 3D): we can consider one as the "input" signal (or image), and the other (called the kernel) as a "filter" on the input image, producing an output image (so convolution takes two images as input and produces a third as output).

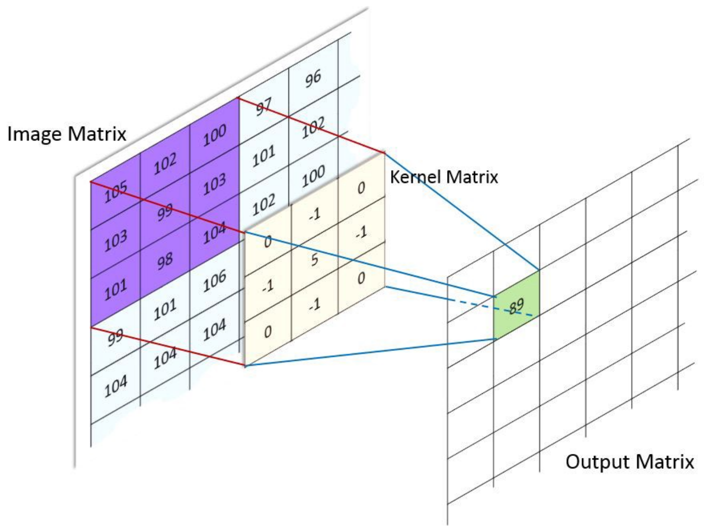


Figure III.4 – Convolution example.

1.1.4 What is a Pooling?

Pooling is a **sample-based discretization process**. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensions and allowing for assumptions to be made about features contained in the sub-regions binned.
There are 2 main types of pooling commonly known as max and min pooling.

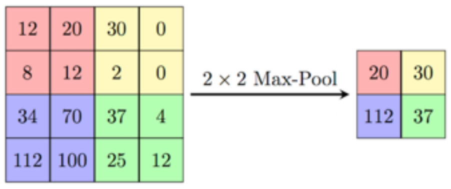


Figure III.5 – Max-Pool example.

1.1.5 What is a Convolutional Neural Network (CNN)?

Convolutional Neural Networks (CNN) [5]is one of the variants of neural networks used heavily in the field of Computer Vision. It derives its name from the type of hidden layers it consists of. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers and normalization layers. Here it simply means that instead of using the normal activation functions defined above, convolution and pooling functions are used as activation functions.

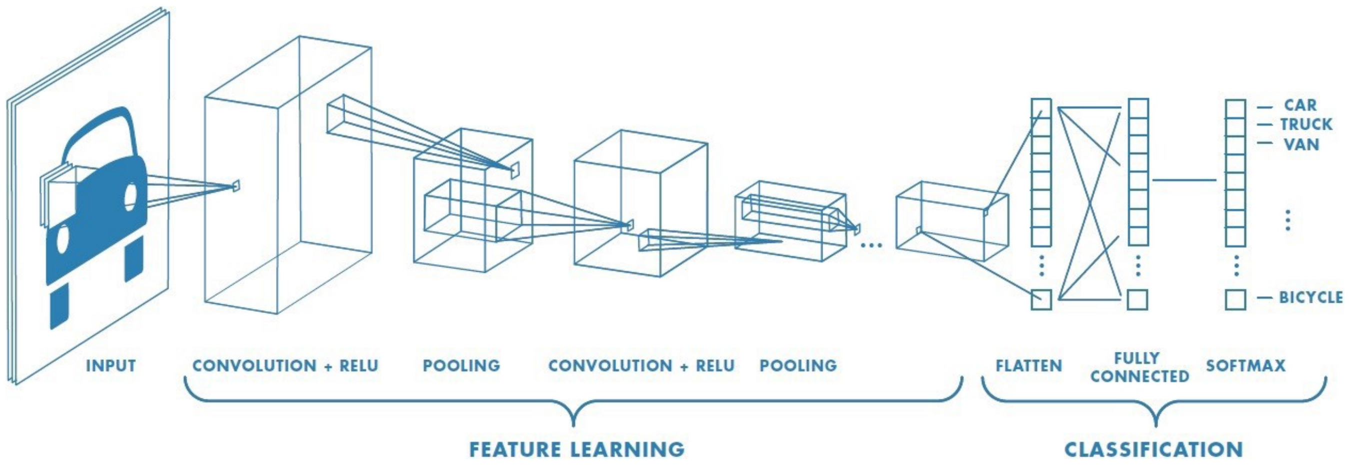


Figure III.6 – CNN example of objects classification.

1.2 Samples generation

1.2.1 What is Samples Generation?

In many real cases, it is difficult to obtain training samples by applying CD methods. The fusion of PCs obtained from multi-temporal images and the spectral correlation angle (SCA) can produce more-representative samples that have high probabilities of either being changed or unchanged, for obtaining multivariate high accuracy. This improves the training of network efficiency with fewer samples. Generating samples for network training consists of three main parts:

- **Identify multiple changes:** using PCA and similarity measures. To identify multiple changes, a difference image (DI) was produced using PCA and the spectral similarity measure.
- **Endmember extraction:** The PCs and SCA were calculated using multi-temporal images and fused to form the DI. To select training samples for each class, the endmembers were extracted as a reference spectrum of each class.
- **Classification:** The pixels in which the spectral angle was lower than the threshold were assigned to each endmember class.

1.2.2 What is Endmember extraction ?

One of the most popular and important methods for analyzing hyperspectral data is spectral unmixing. A few spectral signatures jointly occupy a single pixel when the spatial resolution of the hyperspectral image is too coarse to distinguish between different materials on the ground.

The measured spectral signature is thus a composite of the individual spectra. Although sub-pixel nonlinear mixing can be important in certain types of analyses, the effects of multiple scattering in the majority of applications are assumed to be negligible when a linear model is used . The key task when using a linear mixture model is to find an appropriate set of pure endmembers (spectral signatures).
The IEA is one of the popular, sequential, linear constrained endmember extraction algorithms based on the linear mixture model. The algorithm identifies endmembers one by one based on previously extracted endmember.

1.3 Recurrent 3D fully convolutional network Re3FCN :

1.3.1 What is Re3FCN?

Segmentation of 3D images is a fundamental problem in Hyperspectral image analysis. Deep learning (DL) approaches have achieved state-of-the-art segmentation performance.
To exploit the 3D contexts using neural networks, known DL segmentation methods, including 3D convolution, 2D convolution on the planes orthogonal to 2D slices, and LSTM in multiple directions, all suffer incompatibility with the highly anisotropic dimensions in common 3D images.
A recurrent 3D fully convolutional network a new DL framework for 3D image segmentation, based on a combination of a fully convolutional network (FCN) and a recurrent neural network (RNN), which are responsible for exploiting the intra-slice and inter-slice contexts, respectively. To our best knowledge, this is the first DL framework for 3D image segmentation that explicitly leverages 3D image anisotropic.

2 Our solution

Our method is divided into two parts, namely:

- **Generating samples for network training**; This allows to identify multiple changes, a difference image (DI) was produced using PCA and the spectral similarity measure. The PCs and SCA were calculated using multi-temporal images and fused to form the DI. To select training samples for each class, the endmembers were extracted as a reference spectrum of each class. Finally, the pixels in which the spectral angle was lower than the threshold were assigned to each endmember class. The samples were then selected randomly, and 3D image patches centered at each selected sample were fed into the Re3FCN network.

- **Training the Re3FCN and producing the CD map**; The 3D patches obtained from each image passed through the 3D convolutional layer to extract spectral and spatial information, whereupon the spectral-spatial feature maps were fed into the ConvLSTM layer. In this phase, the temporal information between the two images was reflected. The output of the ConvLSTM layer was fed into the prediction layer to generate the score map. The number of final feature maps equaled the number of classes. Finally, the pixels were classified into the final classes according to the score map.

2.1 Sample Generation :

We have T_1 and T_2 two HSIs acquired over the same location at different times T_1 and T_2, respectively. The size of each image is C × R × L, where C and R are the columns and rows of the image, respectively, and L is the number of spectral bands. Figure III.7 shows the flow of generating samples for network training.

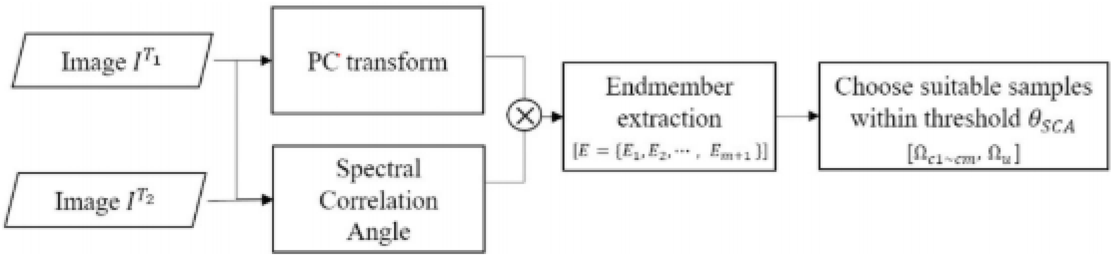


Image I^{T_1}
Image I^{T_2}
PC transform
Spectral Correlation Angle
Endmember extraction
$[E = [E_1, E_2, \cdots, E_{m+1}]]$
Choose suitable samples within threshold θ_{SCA}
$[\Omega_{c1\sim cm}, \Omega_u]$

Figure III.7 – Flowchart of sample generation.

If I_{T_1} and I_{T_2} have the same size, $L_1 = L_2$ and $n_1 = n_2$. Considering each band as a vector, the stacked image I_s can be simply defined as follows :

Principal component analysis PCA has been used as a data compression technique.
Uncorrelated linearly transformed components are derived from the original data such that the first principal component accounts for the maximum possible proportion of the variance of the original data set, and subsequent components account for the maximum proportion of the unexplained residual variance, and so forth.

Although the optimal PCs show the changed and unchanged pixels; some pixels are not distinguished by the PCs. To discriminate between the changed and unchanged pixels, the

$$I_s = \begin{pmatrix} X_1^{T_1} \\ X_2^{T_1} \\ \vdots \\ X_L^{T_1} \\ X_1^{T_2} \\ X_2^{T_2} \\ \vdots \\ X_L^{T_2} \end{pmatrix}$$

optimal PCs and spectral similarity values are combined. Our solution used the SCA, which reflected the level of the linear correlation between two spectral vectors

$$SCM(x,y) = \frac{\sum_{l=1}^{L}\left(I^{T_1}(x,y,l) - \overline{I^{T_1}}(x,y)\right)\left(I^{T_2}(x,y,l) - \overline{I^{T_2}}(x,y)\right)}{\sqrt{\sum_{l=1}^{L}\left(I^{T_1}(x,y,l) - \overline{I^{T_1}}(x,y)\right)^2 \sum_{l=1}^{L}\left(I^{T_2}(x,y,l) - \overline{I^{T_2}}(x,y)\right)^2}}$$

$$SCA(x,y) = cos^{-1}\left(\frac{SCM(x,y)+1}{2}\right),$$

where I T1 (x, y) and I T2 (x, y) are the mean values of the vectors of I T1 (x, y, l) and I T2 (x, y, l), respectively.

Then we generate a difference image DI by :

$$DI_i(x,y) = PC_i(x,y) \times SCA(x,y)$$

where DIi and PCi are the i-th band and the selected PC, respectively.
After generating the DI, we used iterative error analysis (IEA) to extract the endmembers.The IEA is one of the popular, sequential, linear constrained endmember extraction algorithms [6] based on the linear mixture model. The algorithm identifies endmembers one by one based on previously extracted endmembers.

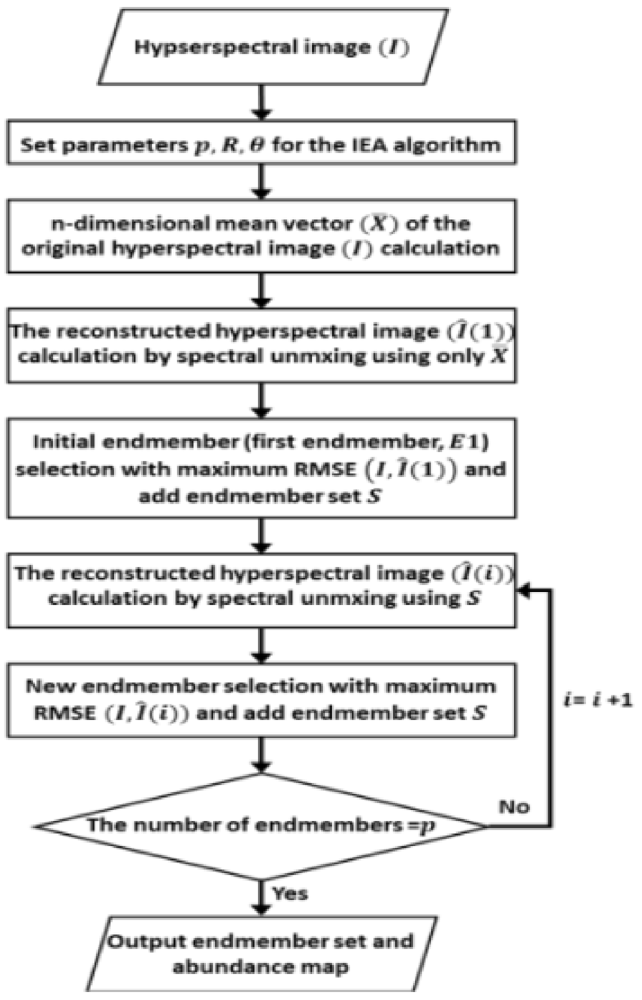

Hypserspectral image (I)
Set parameters p, R, θ for the IEA algorithm
n-dimensional mean vector ($\bar{X}$) of the original hyperspectral image (I) calculation
The reconstructed hyperspectral image ($\hat{I}(1)$) calculation by spectral unmxing using only $\bar{X}$
Initial endmember (first endmember, $E1$) selection with maximum RMSE $(I, \hat{I}(1))$ and add endmember set S
The reconstructed hyperspectral image ($\hat{I}(i)$) calculation by spectral unmxing using S
New endmember selection with maximum RMSE $(I, \hat{I}(i))$ and add endmember set S
$i = i + 1$
The number of endmembers $= p$
No
Yes
Output endmember set and abundance map

Figure III.8 – Logical flow of the IEA algorithm [2]

Finally, the pixels were clustered to each endmember class using the spectral similarity values.Pixels were extracted randomly from Ωc and Ωu as training samples; whilst no pixels were taken from Ωb . The 3D image patches centered at these pixels were fed into the network

2.2 Training Re3FCN and Producing CD Map:

2.2.1 Spectral-Spatial Module with 3D Convolutional Layers

The 3D patches were selected randomly from T1 and equivalently from T2. Let iT1 and iT2 be 3D patches taken from the same location in T1 and T2, respectively. iT1 and iT2 were fed separately into 3D convolutional layers and batch normalization (BatchNorm) layers. the 3D convolution can extract both spatial and spectral features simultaneously using a 3D kernel.

3D convolution is calculated as:

$$o_{lj}^{xyz} = f\left(\sum_{n}\sum_{r=0}^{R-1}\sum_{i=0}^{M-1}\sum_{j=0}^{M-1} w_{n}^{ijr}\, o_{(l-1)n}^{(x+i)(y+j)(z+r)} + b\right)$$

where oxyz j is the pixel value at position (x, y, z) in the j-th 3D feature cube and in the l-th layer. R is the spectral dimension of the 3D kernel, and w ijr n is the weight value at position (i, j,r) connected to the n-th feature in the (l 1)-th layer.

2.2.2 Temporal Module with Convolutional LSTM

The LSTM, which is an RNN architecture that can remember values over arbitrary intervals using a memory cell ct at time step t. The LSTM has three gates, namely the input gate ig, the output gate og, and the forget gate f g, each of which has a learnable weight. f gt is the gate for forgetting the previous information.

The ConvLSTM is a modification of the conventional LSTM by replacing the matrix multiplication operators with the convolution operators. ConvLSTM operates as Equations (1)–(4), which is the modification of Equations (2)–(3), wherein the matrix multiplication operators are replaced by convolution operators. Its structure is shown in Figure III.10.

$$fg_t = \sigma\left(W_{hfg} * h_{t-1} + W_{ffg} * f^{T_t} + b_{fg}\right) \quad (1)$$

$$ig_t = \sigma\left(W_{hig} * h_{t-1} + W_{fig} * f^{T_t} + b_{ig}\right) \quad (2)$$

$$og_t = \sigma\left(W_{hog} * h_{t-1} + W_{fog} * f^{T_t} + b_{og}\right) \quad (3)$$

$$\overline{c_t} = \tanh\left(W_{h\overline{c}} * h_{t-1} + W_{f\overline{c}} * f^{T_t} + b_{\overline{c}}\right) \quad (4)$$

$$c_t = fg_t \odot c_{t-1} + ig_t \odot \overline{c_t} \quad (5)$$

$$h_t = og_t \odot tanh(c_t) \quad (6)$$

Figure III.9 – ConvLSTM Activation Functions.

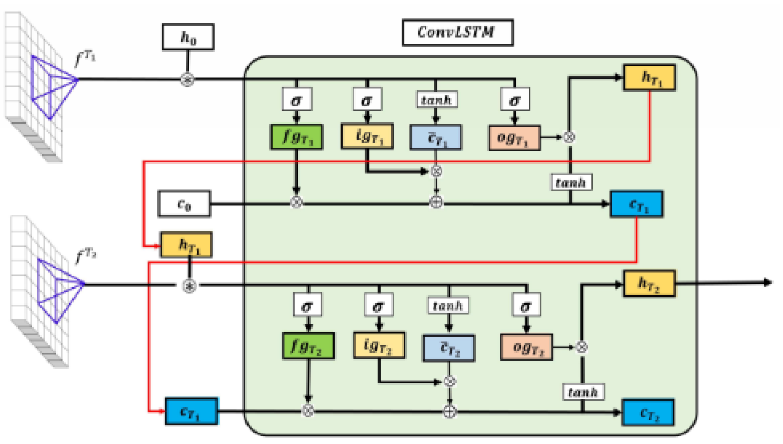


Figure III.10 – Structure of the convolutional long short-term memory (LSTM).

2.2.3 Proposed Model

The Re3FCN is based on FCN architecture that includes 3D convolutional layers and the ConvLSTM. Figure III.11 shows the architecture of the Re3FCN. Rather than using a fully connected layer, the network uses a convolutional layer at its end

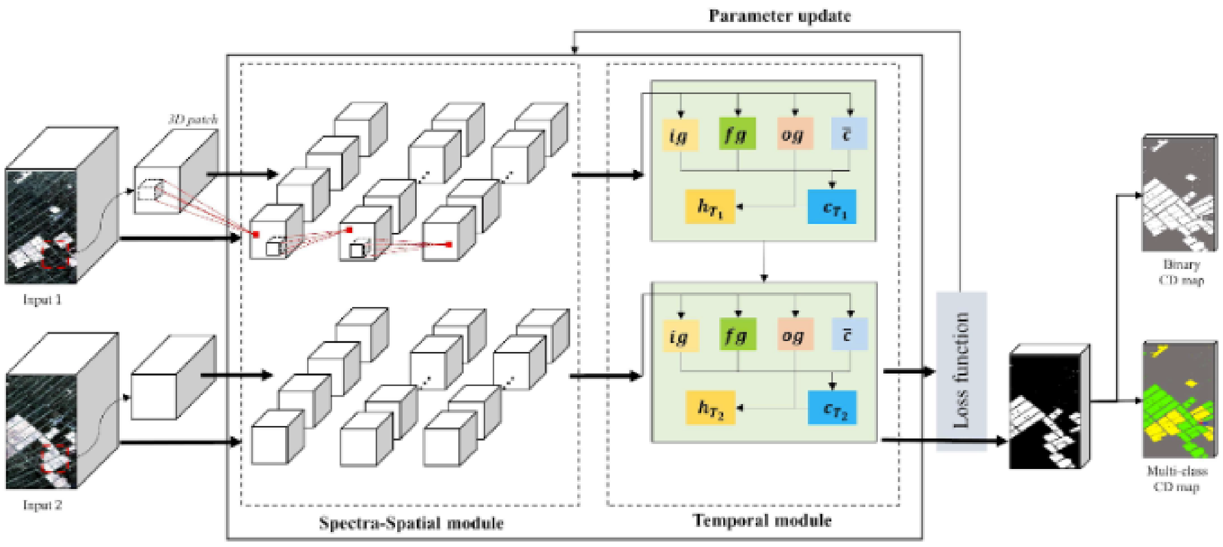
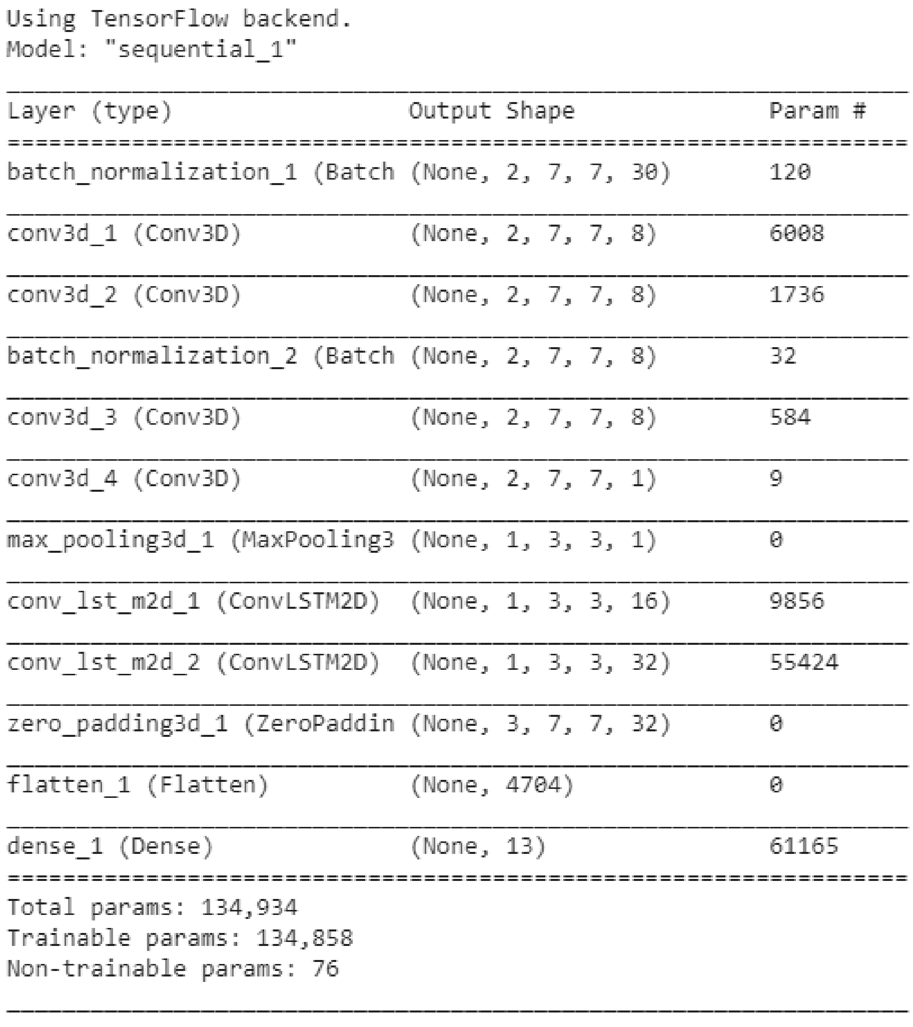


Figure III.11 – Architecture of the proposed change detection (CD) model.

Model's Layers



```
Using TensorFlow backend.
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
batch_normalization_1 (Batch (None, 2, 7, 7, 30)       120
_________________________________________________________________
conv3d_1 (Conv3D)            (None, 2, 7, 7, 8)        6008
_________________________________________________________________
conv3d_2 (Conv3D)            (None, 2, 7, 7, 8)        1736
_________________________________________________________________
batch_normalization_2 (Batch (None, 2, 7, 7, 8)        32
_________________________________________________________________
conv3d_3 (Conv3D)            (None, 2, 7, 7, 8)        584
_________________________________________________________________
conv3d_4 (Conv3D)            (None, 2, 7, 7, 1)        9
_________________________________________________________________
max_pooling3d_1 (MaxPooling3 (None, 1, 3, 3, 1)        0
_________________________________________________________________
conv_lst_m2d_1 (ConvLSTM2D)  (None, 1, 3, 3, 16)       9856
_________________________________________________________________
conv_lst_m2d_2 (ConvLSTM2D)  (None, 1, 3, 3, 32)       55424
_________________________________________________________________
zero_padding3d_1 (ZeroPaddin (None, 3, 7, 7, 32)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4704)              0
_________________________________________________________________
dense_1 (Dense)              (None, 13)                61165
=================================================================
Total params: 134,934
Trainable params: 134,858
Non-trainable params: 76
_________________________________________________________________
```

Figure III.12 – Model's Layers.

2.3 Used tools

To carry out this project, we used the Python ecosystem oriented Data Science and for realize this model and build the neural network we used the learning library deep Keras [7] which is a high-level neural network API written in Python and interfaced with TensorFlow. This library was developed with the aim of allowing quick experiments and being able to get from idea to result with the shortest delay possible being the key to effective research

Conclusion

In this chapter, we presented the followed approach to build our model and highlighted its theoretical architecture. In the next chapter, we will discuss and interpret the trained model results on the Hyperspectral data.

Chapter IV

Experimentation and Results discussion

Introduction

In this chapter we will evaluate our approach, it is necessary to apply the proposed model on HSIs to be able to discuss different results. We will present the obtained results, discuss the model parameters and conclude our work with some interpretations.

1 Evaluation Criteria

The evaluation of our Hyperspectral Change-Detection model is divided into 5 parts:

- **Train accuracy**; percentage of correct changes-detection on the training set.
- **Validation accuracy**; percentage of correct changes-detection on the validation set.
- **Test accuracy**; percentage of correct changes-detection on the test set.
- **Learning Time**; Time spent to complete learning phase.
- **Prediction Time**; Time spent to output detected changes of a given input.

2 Discussion of parameters and their impact on the model

In this section we will evaluate different parameters' effect on the performance of the model'. We already trained the model on the training set using **default parameters** and we are going to proceed with the hyper-parameters tuning using the validation set.
For each parameter, we only tune it alone while fixing the other parameters.

	Parameter	Value
1	Epochs	20
2	Batch size	32
3	Learning rate	10^{-4}
4	Patch size	7x7
5	Normalization Type	Standardization
6	Training set percentage	70%
7	Usage of PCA	NO

Tableau IV.1 – Default parameters.

2.1 Number of Epochs

Fixing the epochs number is both an important and tricky task, a low number of epochs will result in under-fitting, on the other side a high number of epochs will lead to over-fitting. In **Table IV.2**, we can observe that the **Epochs** = 20 gives the best performance among other values, we also notice that fixing **Epochs** = 40 will cause the model to **over-fit** the training data, so we are going to fix our model's Epoch's number to this value.

Epochs	5	20	40
Train Accuracy	0.5257	0.9121	0.97412
Validation Accuracy	0.4863	0.8736	0.8237
Learning Time	8 min	14 min	26 min
Prediction Time	1 min	1 min	1 min

Tableau IV.2 – Evaluation metrics while Tuning Number of Epochs parameter.

2.2 Batch size

Tuning the batch size parameter is important in both accuracy improving and learning time reduction, classic deep learning models used to fix a $batchsize = datasize$ meaning that when solving the optimization problem the **Gradient Descent** (or any other used optimizer) will process the whole data before recalculating the gradients and adjusting the directions, this may cause the process to miss the local optima, but when using $batchsize < datasize$, the Gradient descent algorithm will recalculate gradients after processing each batch reducing the probability of missing the local optima and significantly reducing computational operations. In **Table IV.3**, we can observe that the **Batch size** = 32 gives the best performance among other values, so we are going to fix our model's batch size to this value.

Batch Size	16	32	64
Train Accuracy	0.8452	0.9120	0.8966
Validation Accuracy	0.7923	0.8237	0.8123
Learning Time	16 min	14 min	12 min
Prediction Time	2 min	1 min	1 min

Tableau IV.3 – Evaluation metrics while Tuning Batch Size parameter.

2.3 Learning rate

To choose a learning rate value, we will try a set of learning rates starting from $5 * 10^{-5}$ up to 10^{-4}.
For each fixed learning rate, we will calculate the different performance metrics of the model then we are going to choose the value maximising the model's performance. In **Table IV.4**, we can observe that the **learning rate** $= 6 * 10^{-4}$ gives the best performance among other values, so we are going to fix our model's learning rate to this value.

Learning Rate	0.00005	0.00006	0.0001
Train Accuracy	0.8340	0.9034	0.8745
Validation Accuracy	0.7738	0.8682	0.8390
Learning Time	11 min	12 min	13 min
Prediction Time	1 min	1 min	1 min

Tableau IV.4 – Evaluation metrics while Tuning Learning Rate parameter.

2.4 PCA Usage

In this section we are going to evaluate our model performance with and without PCA. In **Table IV.4** we can observe that the model's performance without PCA is slightly better, but the Learning time is longer. Due to comparable performance between the two options **we decided to use PCA**.

PCA	Without PCA	With PCA
Train Accuracy	0.9519	0.9330
Validation Accuracy	0.8930	0.8724
Learning Time	14 min	11 min
Prediction Time	2 min	1 min

Tableau IV.5 – Evaluation metrics with and without PCA.

3 Results

We are going to test our model on the **Indian Pine** HSIs

- Train Accuracy : 87.29
- Validation Accuracy :82.56

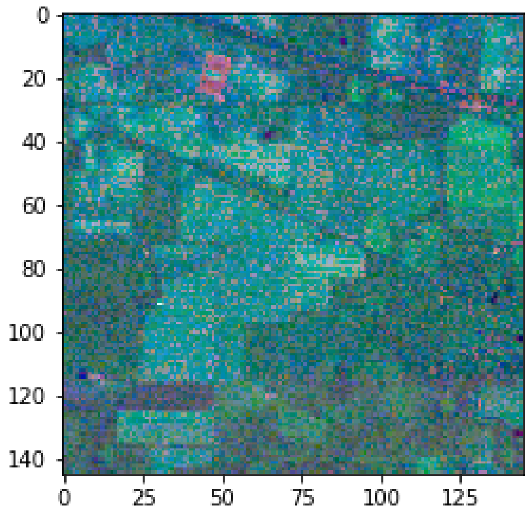


Figure IV.1 – Image at T_1

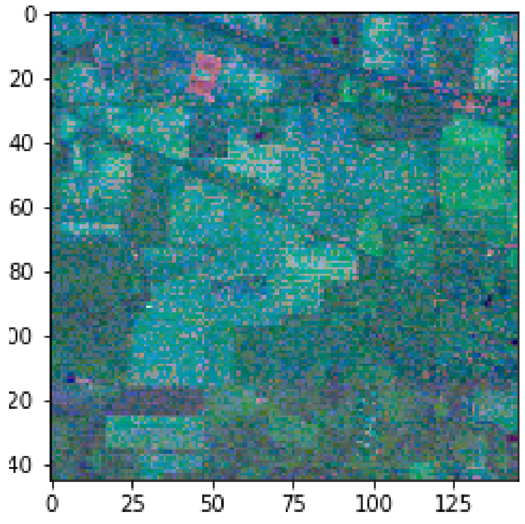


Figure IV.2 – Image at T_2

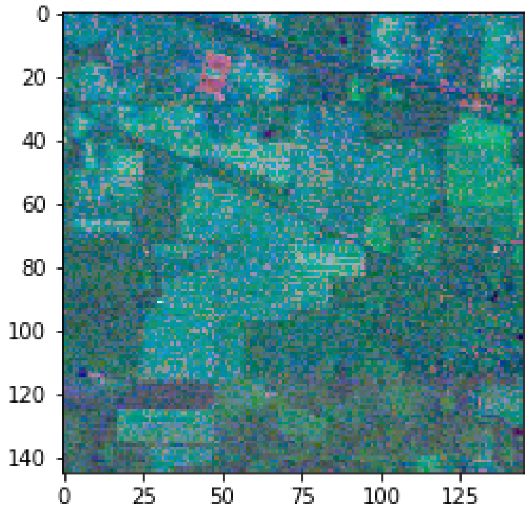


Figure IV.4 – Image at T_1

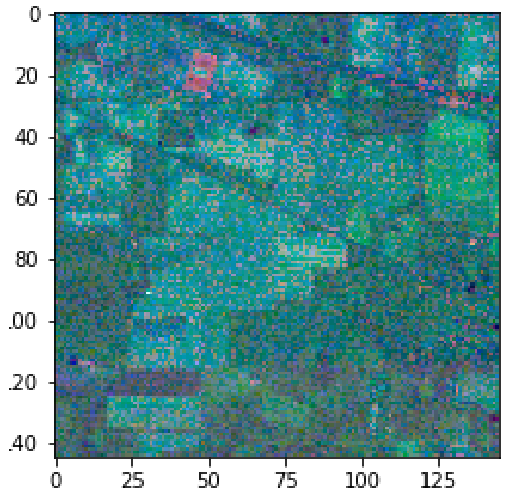


Figure IV.5 – Image at T_3

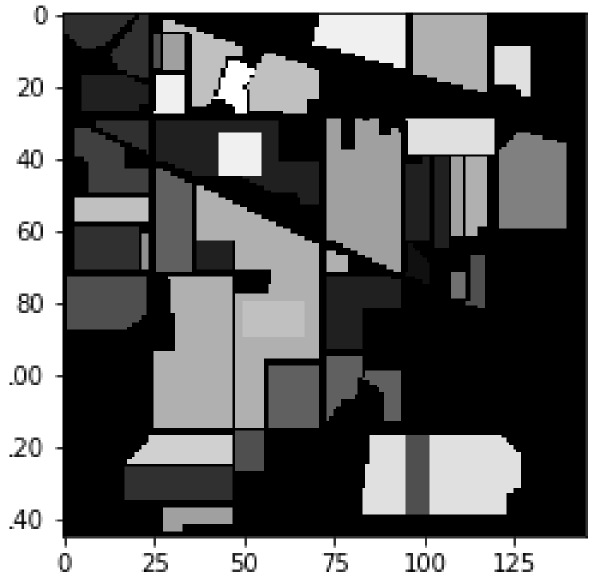


Figure IV.3 – Change detection ground truth betwenT_1 and T_3.

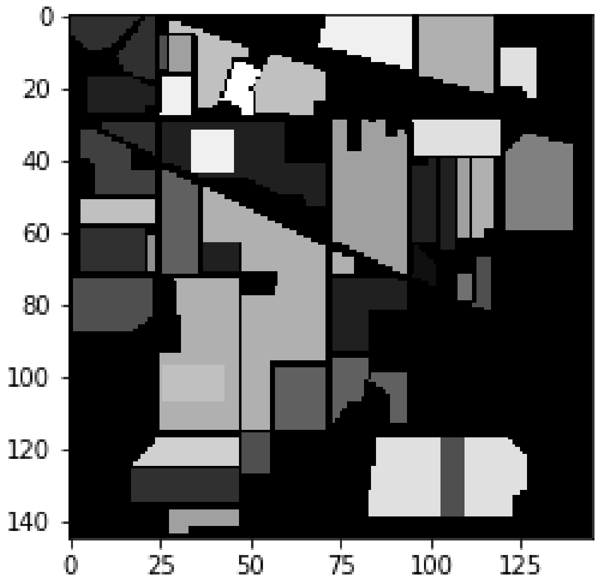


Figure IV.6 – Change detection ground truth between T_1 and T_3.

Conclusion

In this chapter we evaluated our model and made some experiments to improve its hyper-parameters. We later applied the final trained model on the Indian Pines dataset to detect changes.

General Conclusion

In our project, we proposed a novel sample generation and a change detection network, called the recurrent 3D fully convolutional network (Re3FCN) based on published paper. This method merged the advantages of both a 3D FCN and a convolutional LSTM.

The PCA and SCA were combined to generate reliable samples with high accuracy, and the training samples were determined based on the SCA value for each endmember, through the classification of each pixel. This method assists in conducting change detection in cases where there is reference or training data, such as in unsupervised change detection. Furthermore, the Re3FCN can not only extract spectral–spatial-temporal information between multi-temporal images, but also is effective in detecting binary and multi-class changes whilst maintaining the spatial structural inputs by replacing fully connected layers with convolutional layers, and can be trained in an end-to-end manner.

Particularly, the Re3FCN was effective in detecting changes in areas from which no training samples had been extracted. However, several problems in the CD method were identified; for example, the errors associated with sample generation can affect the final CD. To solve these problems, we improved the accuracy of the training-sample generation and conducted additional experiments to confirm the ability to learn changed rules from multi-temporal images obtained from different sensors.

References

[1] Jonathon Shlens. *A tutorial on principal component analysis.* decembre (2005). https://www.cs.cmu.edu/ elaw/papers/pca.pdf. iii, 4

[2] Mahesh M. Solankar By Karbhari V. Kale and Dhananjay B. Nalawade. *Hyperspectral endmember extraction techniques.* July (2019). https://www.intechopen.com/books/processing-and-analysis-of-hyperspectral-data/hyperspectral-endmember-extraction-techniques. iii, 22

[3] Youkyung Han by Ahram Song, Jaewan Choi and Yongil Kim. *Change detection in hyperspectral images using recurrent 3d fully convolutional networks.* decembre (2018). https://www.mdpi.com/2072-4292/10/11/1827. 3

[4] The sentinel-1 sar imagery. https://rslab.ut.ac.ir/data. 8

[5] Andrew Ng. *Deep Learning Specialization.* mars (2020). 17

[6] Jaewan Choi Seokkeun Choi Ahram Song, Anjin Chang and Yongil Kim1. *Automatic extraction of optimal endmembers from airborne hyperspectral imagery using iterative error analysis (iea) and spectral discrimination measurements.* February (2015). https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4367322/. 21

[7] Keras: The python deep learning library. https://www.actuia.com/keras/. 25