POLITECNICO DI TORINO

Master Degree in Mechatronic Engineering

Master Thesis

Convolutional Neural Network for cloud segmentation of satellite imagery

Candidate:
Marta Trapero Bernabé

Supervisors:
Tatiana Tommasi
Jacopo Federici

April 2023

Abstract

Satellites provide enormous amounts of information, some in the form of images. These high-altitude shots are extremely valuable for a wide variety of sectors such as forestry, railway industry, urban planning or agriculture, among others. Nowadays, spacecraft capture images and transmit them to ground without prior filtering. However, most of these images are filled with clouds, being useless for most applications. An on-board tool capable of discarding such images would save resources and free up bandwidth during transmission. To achieve this goal, this thesis studies the use of Artificial Intelligence to carry out the task.

This work presents a Deep Learning algorithm that segments clouds in Natural False Color satellite images. In order to decide which images are to be discarded, cloud coverage is calculated by classifying each pixel as cloud or background. Thus, the system receives the image formed from the combination of the spectral bands and outputs its cloud coverage index. This implementation is thought to be deployed in a COTS such as Intel Movidius Myriad 2 to be installed on board satellites.

To this end, the solution developed used a Convolutional Neural Network (CNN) based on the U-Net architecture. Some modifications have been made to the original architecture to better fit the model. The network has been trained and tested using labelled dataset containing Landsat images. In addition, the performance of the model has been studied through quantitative and qualitative analysis.

Acknowledgements

I want to thank my family and friends, specially my mother Susana and Dani. I would like to extend my sincere thanks to Sasan for sharing his knowledge with me. I had the pleasure of collaborating with Jacopo Federici and Tatiana Tommasi.

Contents

List of Figures

List of Tables

Acronyms

AI - Artificial Intelligence

ANN - Artificial Neural Networks

AUC – Area Under the Curve

CNN - Convolutional Neural Network

COTS - Commercial Off-The-Shelf

CPU - Central Processing Unit

FN - False Negative prediction

FP - False Positive prediction

GPU - Graphics Processing Unit

IoU - Intersection over Union

NIR - Near Infrared Band

ROC - Receiver operating characteristic

TN - True Negative prediction

TP - True Positive prediction

USGS - United States Geological Survey

Chapter 1

Introduction

This thesis proposes the implementation of a deep learning algorithm to evaluate the cloud cover on the satellite to transmit only useful images. And so, the aim of this report is to describe and synthesise the work undertaken to accomplish that task.

This chapter is divided into several sections. The first section details the main motivations behind this research. The following one details the scope of the work by listing the objectives set. And lastly, the structure of the project is described, indicating what will be covered in each chapter.

1.1 Motivation

Being able to observe the earth's surface from above, has innumerable advantages in a wide variety of sectors. With applications in agriculture, fire and flood control, maritime and land traffic control, or land use classification, among others.

Advances in technology have enabled the development of more affordable and compact satellites. Spreading its use and opening the doors of space to new agents. Among the limitations of this new technology are processing capacity and bandwidth. One approach to mitigate this problem is to reduce the amount of images to be transmitted.

Study [1] revealed that almost 70% of the Earth's surface is filled with clouds at any given time. Clouds are a hurdle for most applications and diminish the usefulness of satellite data. Therefore a tool able to identify and discard images full of clouds is needed.

Machine Learning stands out as a solution for managing large volumes of data.

In particular, Convolutional Neural Networks outstand as an approach for image processing. This has motivated the study of the applicability of this technology for the detection of clouds in satellite images.

1.2 Objectives

The aim of this thesis is to develop a Deep Learning algorithm that is able to detect and evaluate the cloud cover of satellite images. In addition, it has also been studied some uncertainties of the model and the error rates related to it. For this general aim the following specific objectives have been established:

- Introduction to the field of satellite imagery.
- Understand computer vision applied to deep learning.
- Develop a Convolutional Neural Network for semantic segmentation.
- Study of the metrics evaluating a semantic segmentation algorithm.

1.3 Thesis outline

This study consists of five main blocks that cover the topics mentioned above. A detailed explanation of what is included in each chapter can be found below.

Chapter 1 is the current section and presents an overview of the thesis and its content. It introduces the reasons behind the development of this thesis, the set objectives and describes the steps followed.

Chapter 2 of this project introduces the state of the art in the field of Convolutional Neural Networks, deepening in the study of object recognition. It also contains a section introducing the handling of satellite images.

Chapter 3 details the environment used for the deployment of the model. Including the gathering of the dataset, and the development and set up of the algorithm.

Chapter 4 reports and discusses the results obtained from the analysis of the model. The outcomes are assesed in a quantitative and qualitative way. The analytical approach outlines the performance of the model by measuring the main metrics for evaluating semantic segmentation models. The visual inspection presents a deeper insight to the main errors encountered.

Chapter 5 describes the conclusions resulting from this project and proposes open lines of research for future work.

Chapter 2

State of art

2.1 Satellite imagery

Satellites like Sentinel or Landsat, orbit the earth to explore it from another point of view. They are equipped with sensors on board that collect large amounts of data. The importance of being able to observe the Earth's surface has led to the development of new techniques and infrastructures to obtain more accurate and higher quality data. Information to be used in countless applications such as meteorology, conservation, geology, agriculture, cartography, education, military purposes and applications that provide solutions to commercial activities among others.

Satellite imagery are photographs obtained by sensors on board artificial satellites, which capture the electromagnetic radiation emitted by bodies on Earth. These images are downlinked to ground as a set of arrays, one for each sensor channel, containing values from 0 to 255. Zero indicates that no radiation arrives from that point, and 255 indicates that the highest radiation value is received.

Devices capture data in several spectral bands that vary in wavelength. Each wavelength range constitutes a bands and gathers information about an specific characteristic. The combinations of various bands emphasise different features, provinding information that otherwise would be undetectable to human eye. This combination of bands is known as "Natural false color."

As an example, the combination of Red Green and Blue bands results in a pancromatic image. It achieves a high spatial resolution as it covers the range 400 to 700 nanometers. Merging this image with another spectral band, such as NIR, would result in a clearer image that highlights water masses. Other bands can measure the radiant heat emitted by the earth or detect fine particles on the atmosphere [2].

Depending on the sensors installed on board satellites, different wavelenghts can be detected and therefore different bands will be obtained. Table 2.1 below shows a sample of the bands analysed by Landsat 8 satellite [3].

Table 2.1: Landsat 8-9 spectral bands

Band	**Name**	**Wavelength (μm)**	**Resolution (m)**
Band 1	Coastal aerosol	0.43-0.45	30
Band 2	Blue	0.45-0.51	30
Band 3	Green	0.53-0.59	30
Band 4	Red	0.64-0.67	30
Band 5	Near Infrared (NIR)	0.85-0.88	30
Band 6	Short Wave Infrared (SWIR) 1	1.57-1.65	30
Band 7	Short Wave Infrared (SWIR) 2	2.11-2.29	30
Band 8	Panchromatic	0.50-0.68	15
Band 9	Cirrus	1.36-1.38	30
Band 10	Thermal Infrared (TIRS) 1	10.6-11.19	100
Band 11	Thermal Infrared (TIRS) 2	11.50-12.51	100

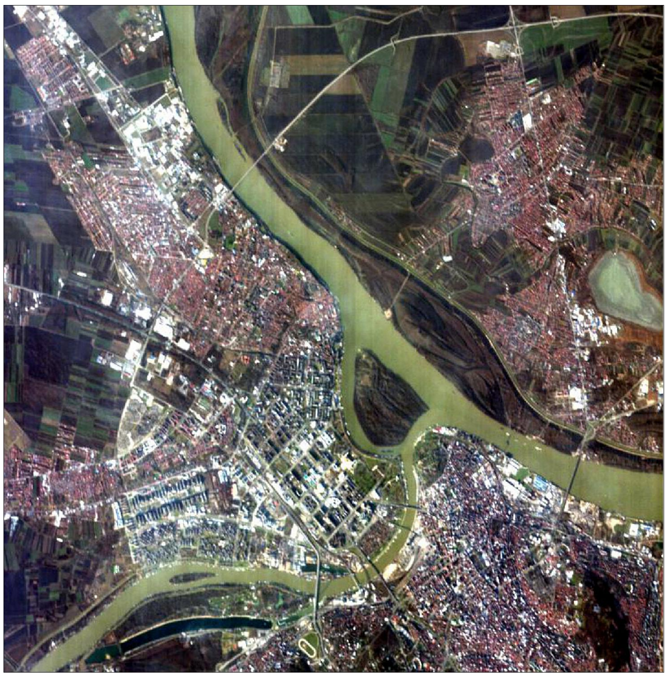
There are available free data sources provided by either government or private agencies, from which a rich dataset can be generated. Some of this resources are USGS Earth Explorer or Copernicus Open Access Hub. They offer a large amount of satellite images from their different satellites. Providing for each geographical area studied, the different bands obtained. This is raw data which, for most applications, must be pre-processed before being used.

2.2 Computer vision

This thesis deals with the analysis of images, so it has been intended to know the state of the art in image processing. In this sub-section, the main concepts to be taken into account for this purpose are included.

Computer vision is the discipline that studies the development of techniques for obtaining, processing and analysing images of the real world. Its aim is to automate the analysis of images or videos and from this to be able to make decisions or understand the environment [4]. In other words, to replicate the way humans perform visual tasks. The use of computer vision generates great benefits in a wide variety of fields such as autonomous driving in the transport sector, faster and more accurate diagnoses in healthcare or better surveillance in the field of security, among others. The most common operations in object recognition are:

Classification: Categorises what is in the image. The output of the code is a prediction of the possible classes of the identified object. An example of this can be seen in Figure 2.1.



River, Building, Road

Figure 2.1: Classification

Detection: Identifies where objects are located in the image. Detected bodies are framed in boxes in the image. They can also be further evaluated by a classifier to categorise them. An example of this can be seen in 2.2.

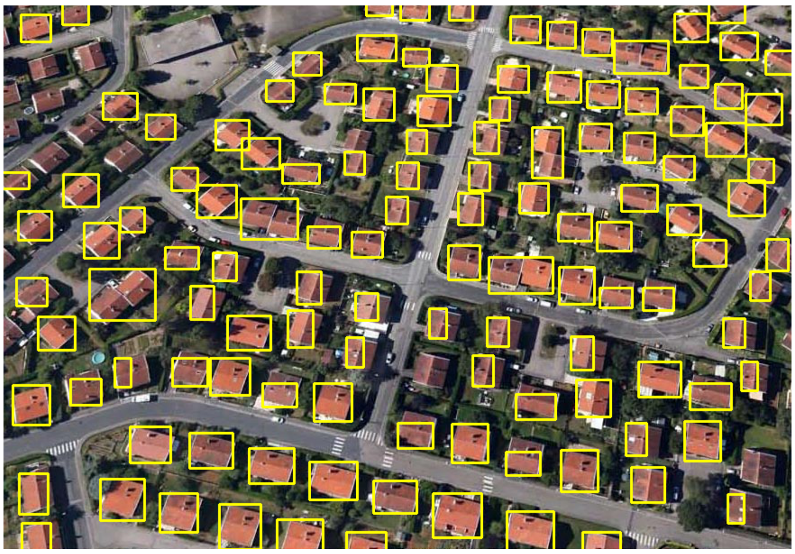


Figure 2.2: Detection

Segmentation: Analyses each pixel of the image and assigns it to a class. Either by grouping different objects under the same class (Semantic segmentation) or by identifying different objects of the same category. Fig. 2.3 shows an example of segmentation.

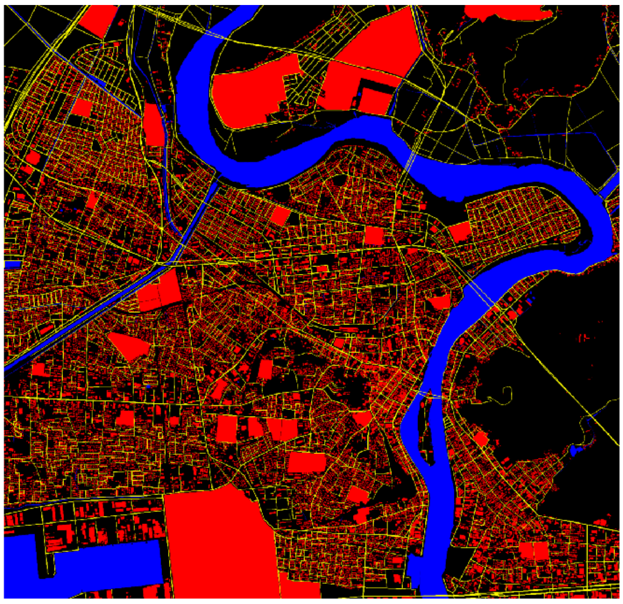


Figure 2.3: Segmentation

Although it is trivial for humans to recognise objects, for a computer vision algorithm it is necessary to take into account the following challenges [5]:

- Change of point of view: identifying the same element from different perspectives.
- Intra-class variation: the same element can have different aspects.
- Disorder of the background: the element is camouflaged with the background.
- Lighting: changes in lighting such as backlighting.
- Deformation: the object may not be rigid and alter its shape.
- Occlusion: the object to be recognised is partially hidden.

A good object recognition algorithm must be robust enough to be invariant to all these challenges, as well as fast. To achieve such a result, developers are leaning towards mimicking the human brain through techniques such as deep learning.

2.3 Deep Neural Networks

The exploitation of the vast amount of data that satellite imagery can provide, leads to a considerable increase in the effort required to analyse and interpret it. In order to develop a tool that overcomes these drawbacks, we wanted to investigate the effectiveness of the use of neural networks. In order to do so, first, the basic concepts of deep learning will be explored in depth. This section begins by discussing where deep learning fits into the field of artificial intelligence. It then introduces artificial neural networks and delves into convolutional networks. Finally, it shows the procedure followed for the development and end-use of this tool.

2.3.1 Artificial Intelligence, Machine Learning and Deep Learning

Artificial Intelligence (AI) is the intelligence performed by machines. In computer science, the field of AI research is defined as the study of "intelligent agents": Any device that analyses its environment and performs actions that maximise its performance given the evidence and knowledge provided [6]. The term "artificial intelligence" is generally associated with the ability of machines to mimic "cognitive" functions that humans relate to other human abilities, such as "learning" and "problem solving". Capabilities currently classified as AI include successful understanding of human speech, high-level competence in strategic gaming systems (such as chess), autonomous car driving, military simulations and interpretation of complex data [7].

The branch of Artificial Intelligence that tries to learn by itself from input data is **Machine Learning**. According to Arthur Samuel in 1959, ”It is the field of study that gives computers the ability to learn without being explicitly programmed”. Nowadays, machine learning is a combination of several disciplines such as statistics, probability, or functional analysis [8], in order to build algorithms that can learn and make predictions about data [9].

Deep Learning is one of the techniques of machine learning, and is based on the structure of neural networks. It mimics the architecture and functioning of the human brain by creating networks with connections between modules. These models consist of layers of non-linear information processing, both labelled (supervised learning) and unlabelled (unsupervised learning) data for pattern analysis and classification [9] Its main applications are computer vision, automatic speech recognition, automatic audio recognition, natural language audio recognition, natural language processing or bioinformatics [10].

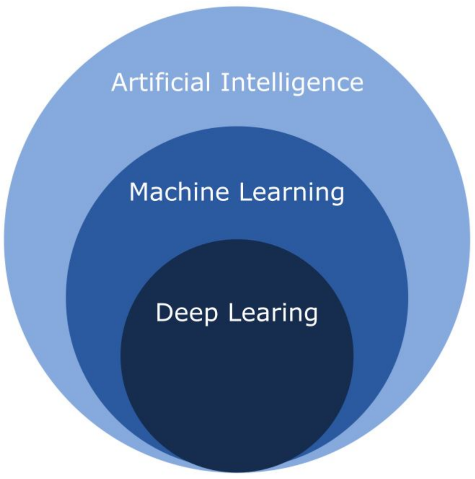


Figure 2.4: Venn Diagram, Deep Learning

2.3.2 Artificial Neural Networks

Artificial neural networks (ANNs) are mathematical models inspired by the structure and functioning of the human brain, to be used in computers to mimic human learning behaviour [11]. The basic unit of computation is the artificial neuron.

This processing unit consists of a series of inputs ”xi” (which are equivalent to dendrites), where the data to be analysed is received. The inputs are balanced by weights ”wi”, which control the strength of the impulse at that synapse. The signals are then linearly combined together with the bias parameter ”b, and are evaluated by an activation function. This results in the output of the processing unit [12]. Figure 2.5 shows a comparative illustration between a biological and an artificial neuron.

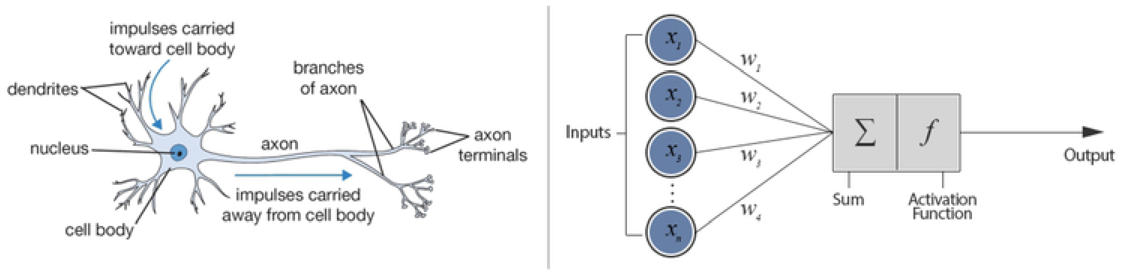


Figure 2.5: Representation of Biological neuron versus Artificial Neuron

In general, Neural Networks are organised in layers of neurons, but the way in which the nodes are connected determines how the network will perform its computations. Therefore it is a highly important decision that the developer has to take [13].

The Fully Connected Network is the most general structure, any other one is a special type of this kind of network. In this type of network, each neuron is linked to all upstream and downstream units of the previous and subsequent layers. These levels can be divided into three areas, the "Input layer", which feeds the input values into the network, the "Output layer", which acts as an interface, and the "Hidden layers", which contain the neurons in between. The number of hidden layers is considered as a measure of the "depth" of an ANN [14].
The connections between the different neurons are balanced by the weights, and it is in the training stage that these parameters are adjusted to achieve the desired result. One of the most commonly used methods to achieve this is the backpropagation method [10].

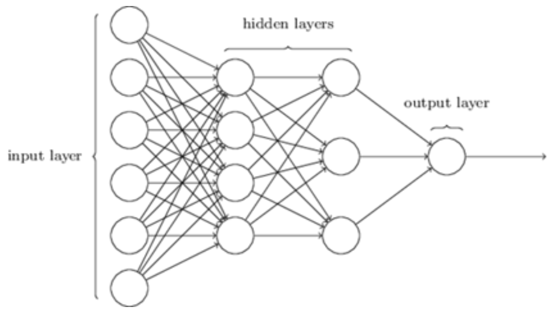


Figure 2.6: Structure of a Deep Neural Network

2.3.3 Convolutional Neural Networks

These networks use a special architecture that is particularly suitable for dealing with images. Moreover, the use of this architecture makes convolutional networks fast to train. Like any other neural network, each neuron receives inputs, performs a series of calculations with weights and biases, and finally returns a score for each proposed class. However, the difference with this type of network is that it assumes that the input data are images. This implies that these networks are designed to evaluate each pixel as input data to the network. Convolutional neural networks include the following layers in their structure:

Convolutional Layer: Its purpose is to extract the most important characteristics of the image in order to identify its features. This is achieved by applying a filter or Kernel to a small portion of the input image and the scalar product is calculated, obtaining a value for that region. The filter is then moved along the image by performing the same operation until finally a ”feature map” or also called ”activation map” is obtained. Seen in Figure 2.7

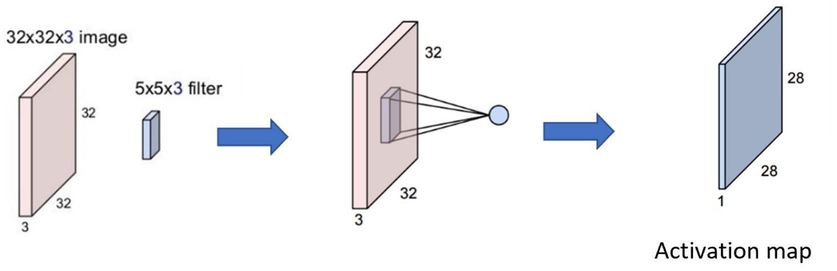


Figure 2.7: Representation of convolution to create an activation map

The filters applied are called activation functions and depending on which one is applied, a different feature is activated. The set of several activation maps forms a layer. This is illustrated in Figure 2.8

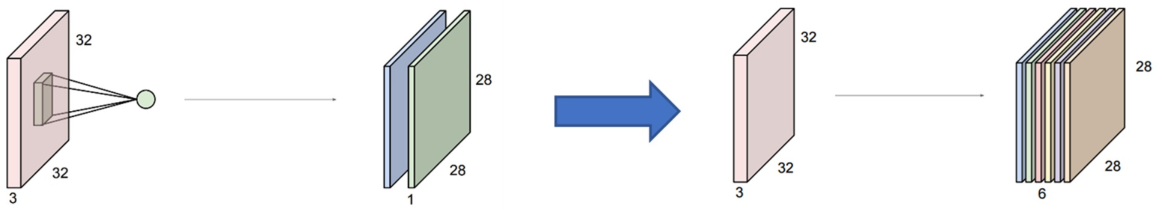


Figure 2.8: Creation of a layer from several activation maps

Some of the most commonly used activation functions are:

Sigmoid function: Output values between 0 and 1, therefore it is used for probability applications.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.1}$$

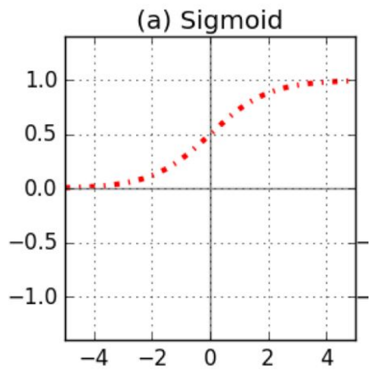


Figure 2.9: Activation function type Sigmoid

Tanh function: Output values between -1 and 1. As it is zero-centered it can strongly assign the output values as hard -1, 0 or +1.

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{2.2}$$

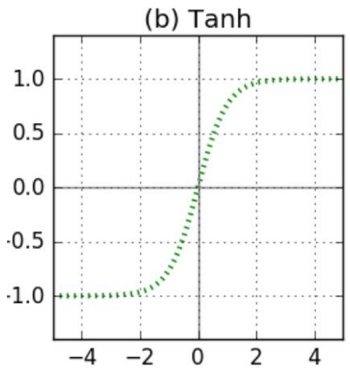


Figure 2.10: Activation function type Tanh

ReLu function: Ranges from 0 to infinity. It is highly used in convolutional neural networks.

$$Relu(z) = max(0, z) \tag{2.3}$$

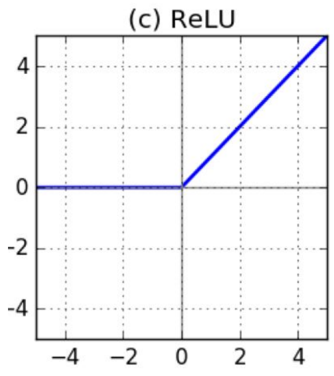


Figure 2.11: Activation function type ReLu

Pooling layer: Its main function is to decrease the number of parameters in the input layer to make it more manageable and thus reduce the amount of computation in the network. This layer is usually placed behind the convolutional layer and operates on each activation map individually. The most commonly used method is "Max Pooling". This approach divides each activation map into quadrants and selects the largest value from each quadrant [15].

Fully connected convolutional layer: It is situated in the last level of Convolutional Neural Networks, in order to connect each resulting activation with the neurons of the output layer [16].

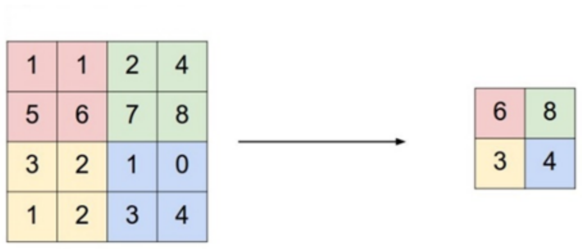


Figure 2.12: Representation of Max Pooling operation

The above mentioned are the most basic layers found in any basic network. However, as the complexity of the architecture increases, other layers can be found. Described below are other layers used in the proposed model.

Dropout: This layer is added to the network to avoid overfitting, explained in detail on 3.3.4 . It forces the network to forget features, to prevent it from learning too much from the training data and be able to generalize. As a result we are able to obtain a model that is able to predict accurately also from unseen data.

Concatenate: It is added to the model with the purpose of adding several layers to obtain just one combined tensor.

2.4 Pipeline of Neural Network models

The development process of supervised deep learning algorithms consists mainly of five phases:

(1) Preparation of the dataset:
Neural networks require a large volume of input data to be trained. Some of the commonly used databases for image processing are ImageNet, Coco, Mnist, Cifar-10.

(2) Model design:
In this phase the topology of the network is defined. The way in which the nodes are connected determines its efficiency. Some of the parameters that need to be defined are, the number of layers of the network, its structure, the order of operations or the number of nodes.

(3) Model training:
The importance of training the networks lies in obtaining features that determine

a prediction. The learning of the neural networks consists in calculating the error and adjusting the weights that weight the input data. This training is made up of forward and backward runs in the neural network layers. In principle, the more data they are trained on, the higher their accuracy will be.

(4) Deployment of the model:
The preparation of the network in the execution environment constitutes the deployment. This requires the implementation of applications that are capable of reading the information provided by the network and adapt it to the format of the desired execution device.

(5) Execution of inference:
Inference is the prediction stage in a trained network. Unlike training, it involves only the forward step. This is usually the last phase of the network pipeline, where real-world data is predicted.

Chapter 3

Development

The scope of this project is to develop a deep learning algorithm that can detect the amount of clouds on images. To this end, it has been chosen to develop a Convolutional Neural Network for semantic segmentation. Since this is a supervised algorithm, the use of labelled data is necessary. Therefore, the development is not only about creating a network and training it, but also about collecting labelled images. Further details on the procedure will be explained below.

3.1 Dataset

Training a supervised deep learning algorithm requires to have a dataset from where the algorithm can extract the most important features and learn from them. The bigger the dataset, the most accurate the algorithm will be. (Up to a point, where uncertainties of deep learning models take into account).

In this case, the input data are images, therefore a search of the available satellite image databases has been made. The repository requires to include a broad sample of satellite pictures of the earth, exhibiting different types of clouds and clear skies. Under this criterion the following sources have been found: Copernicus Open Access Hub [17], USGS Earth Explorer [3] or Google Earth [18]. They are all free source repositories offered by governmental bodies or private agencies, and provide raw free data for non commercial purposes.

However, not just raw images are required, labels must be also available for the images. As dealing with a segmentation model, this tags must be in the form of masks, also called ground truths. Masks are images constituted by pixels of value zero for background and non-zero value for object detected. As the goal of this study is to achieve a semantic segmentation model for cloud recognition, binary ground truths are used. Pixels representing background will have value zero and pixels where a cloud is observed will have value one. An example of an image and a mask can be found in Figure 3.1

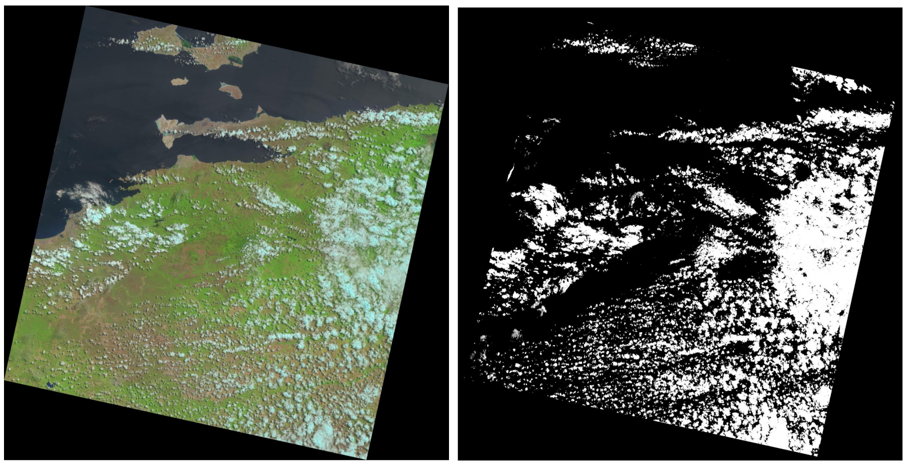

(a) Image (b) Mask

Figure 3.1: Illustration of an image and its corresponding mask

The free repositories discussed above provide just images, but not their ground truths. Therefore, in order to use them as a dataset, it would be necessary to label every image. In fact, here lies one of the biggest problems of using supervised learning, that the data has to be labelled, and as these algorithms need to be trained on huge amounts of data, it is a time-consuming and resource-intensive task.

The shortcut is to use a repository of already tagged images. That is, a database that includes the images and their ground truth masks. The drawback of this option is that a database of labelled images may not be available for the purpose under study, in this case cloud segmentation. Or there may be, but it may be mislabelled. In order to speed up the image collection process, for this study it has been decided to make use of an already tagged repository.

There has been found a labelled dataset for image segmentation available online: *38-Cloud: Cloud Segmentation in Satellite Images from Kaggle [19]*. The dataset includes scenes from Landsat 8 satellite, sampling different types of climate states. It contains 18 images for training, with their corresponding ground truth masks, and 20 images for testing.

Satellite images are extremely large, as an example the size of the image in Figure 3.1 is 1204 x 1231. Direct processing of such images is computationally expensive, therefore it is recommended to crop them into patches and treat them individually.

Kaggle dataset already provides the scenes cropped into patches of size 384x384. Therefore, a total of 8400 images and masks are available for training and 9201 images for testing. Compilation of details of the dataset are described in Table 3.1

Table 3.1: Description of *38-Cloud: Cloud Segmentation in Satellite Images from Kaggle* dataset

Subset	**Bands**	**Scenes**	**Patches**	**Size**
Training	trainBlue	18	8400	384 x 384 x 1
	trainRed		8400	384 x 384 x 1
	trainGreen		8400	384 x 384 x 1
	trainNir		8400	384 x 384 x 1
	trainGt		8400	384 x 384 x 1
Testing	testBlue	20	9201	384 x 384 x 1
	testRed		9201	384 x 384 x 1
	testGreen		9201	384 x 384 x 1
	testNir		9201	384 x 384 x 1

Among the proposed images, many empty patches have been found. This is because the entire images had black borders that should be removed from the dataset. For this purpose, a pre-selection of non-empty patches has been made.

Moreover, as described in 2.1, satellite images are formed by the combination of multiple spectral bands. The dataset offers the images decomposed in channels, separating the collection in various directories according to the type of band. The bands available in this collection of data are: Band 2 (Blue), Band 3 (Green), Band 4 (Red), Band 5 (NIR). A sample of this can be seen in the figure below 3.2.

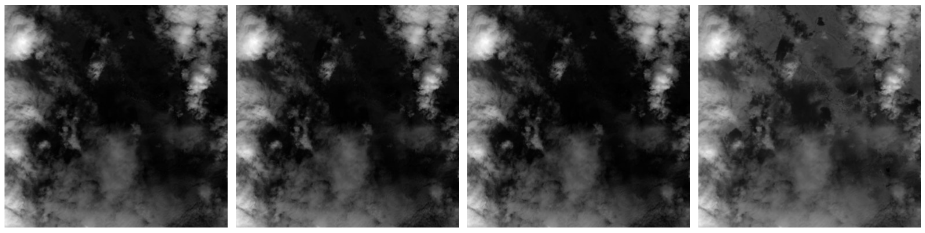

(a) Red patch (b) Green patch (c) Blue patch (d) NIR patch

Figure 3.2: Illustration of the different bands

To prepare the data for training, first the bands have to be merged. The four bands of each patch have been combined to create an array of size 384x384x4 for each of the 8400 training patches and for the 9201 test patches. A sample of a Natural false color resulted from combining the different patches can be observed in Figure 3.3

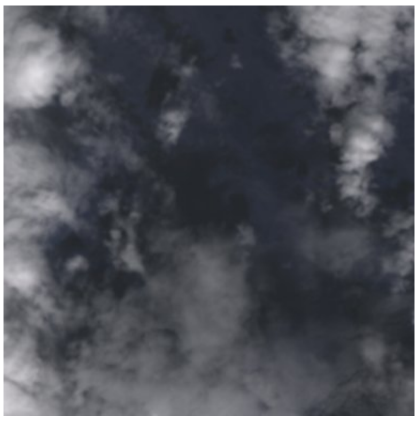


Figure 3.3: Natural false color image from combining Red, Green, Blue and NIR band

Regarding the masks, it is necessary to enlarge the dimensions of the binary images for further processing in the neural network. The aim is to obtain an image of shape 384x384x1. The remodelling has been carried out for each of the 8400 ground truth patches available for training. Figure 3.4 shows the ground truth label of the sample patch presented before.

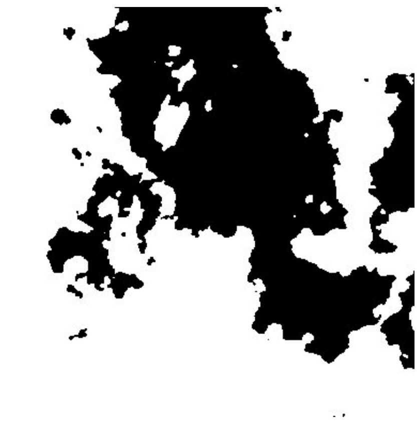


Figure 3.4: Ground truth image

Note that the neural network only works with float data, meaning the values of the variables have to be within the range [0-1]. However, the values of the input images are of type uint8 (unsigned integer 8 bit), which corresponds to 2^8, meaning that they must take values in the range [0-255]. Therefore a format conversion from uint8 to float64 is required. This transformation has been done just before feeding the network with the data set.

Although 8400 patches are available for training, due to limitations of the environment, only 800 images were used to train the network and 200 for validation.

Further details on the training phase are given in section 3.3.

This concludes the processing carried out on the dataset in order to prepare it as input to the neural network.

3.2 Arquitecture

As the main goal is to analyse images to determine their cloud coverage, it has been decided to perform semantic segmentation. In semantic segmentation, Convolutional Neural Networks assign a semantic class to each pixel. In this case classes correspond to cloud or background, therefore the output sought is a binary image. There are a wide variety of CNN architectures that have been successful in image segmentation, including Unet, ResNet, SegNet, among others.

Based on available literature and similar application experiments, the chosen architecture has been Unet. Unet [20] was designed for semantic segmentation of biomedical images, and consists of several Max Pooling, Convolutional and Concatenated layers, arranged in an specific configuration. The structure of the network can be seen in 3.5. However, to better suit it to the project specifications, some modifications on the structure had been performed. These tunnings include the pruning of some levels of the architecture, resulting in a simpler structure of 6 levels instead of 9. In addition, to avoid overfitting, some dropout layers have been introduced. The final architecture for the model is depicted in Figure 3.6. This final model structure is the result of a process of training, analysis of results and re-adjustment of the design.

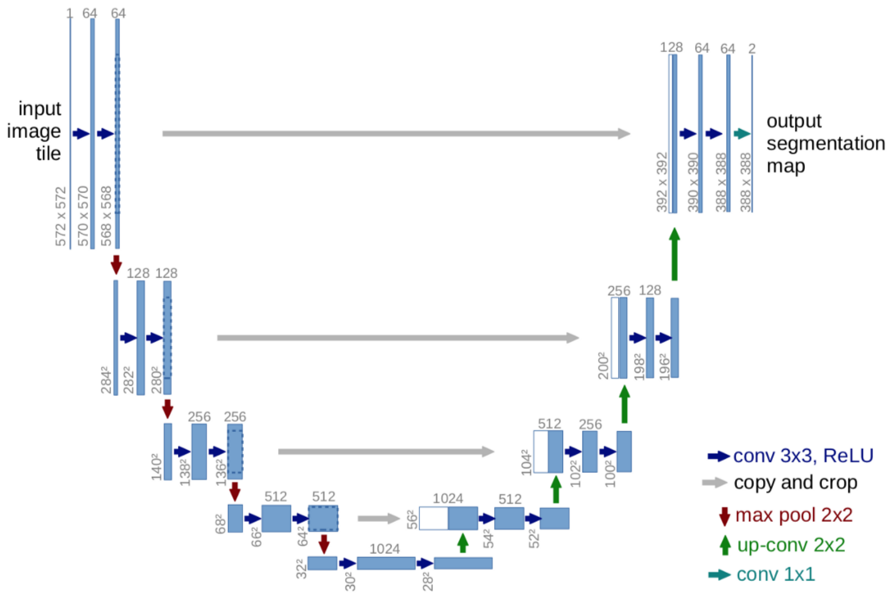


Figure 3.5: Unet architecture

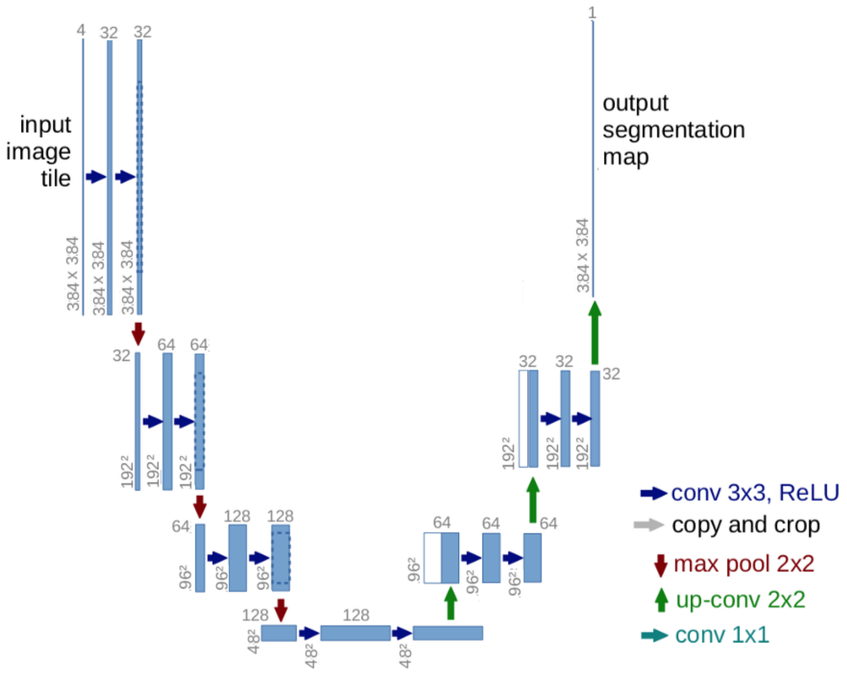


Figure 3.6: Final model architecture

The network consists on a contraction path followed by an expansive path. First, the contraction path acts as a regular CNN. It takes the 384 x 384 x 4 image and performs two 3x3 convolutions with zero padding and a Rectified Linear Unit as an activation function. Next a 2x2 max pooling layer is introduced for downsampling. This combination of layers is repeated several times. At each downsampling the feature parameters are doubled while the size of the image is reduced. In between some of this operations a dropout layer is introduced to prune some data and generalize the model.

Afterwards, in the expansive path the image is upsampled. A 2x2 "Conv2DTranspose" layer reduces the depth of the activation map while expanding its size. Likewise, layers from the expansive path are concatenated with their corresponding feature map of the contraction path. Followed as well by two convolutional 3x3 layers. Some dropout layers are also added in the expansion path.

Finally, the output layer is assembled by a 1x1 convolutional layer with Sigmoid as an activation function. As a result, the model outputs an array with the different probability values for each pixel of each processed image. As the last layer is a sigmoid function, the obtained values range from 0 to 1; meaning 0 been classified as background and 1 been classified as cloud. This array of probabilities is then thresholded to obtain a final binarised image. The threshold chosen has been 0.5, whereby any value higher than the threshold will be automatically identified as a cloud and lower values will correspond to background.

Table 3.2 below shows the backbone structure of the Unet based architecture designed for cloud segmentation of satellite imagery.

Table 3.2: Unet based architecture utilized for cloud segmentation

Layer	Output Shape	Number of parameters
InputLayer	(None, 384, 384, 4)	0
conv2d	(None, 384, 384, 32)	1184
conv2d 1	(None, 384, 384, 32)	9248
max pooling2d	(None, 192, 192, 32)	0
conv2d 2	(None, 192, 192, 64)	18496
conv2d 3	(None, 192, 192, 64)	36928
max pooling2d 1	(None, 96, 96, 64)	0
conv2d 4	(None, 96, 96, 128)	73856
dropout	(None, 96, 96, 128)	0
conv2d 5	(None, 96, 96, 128)	147584
max pooling2d 2	(None, 48, 48, 128)	0
conv2d 6	(None, 48, 48, 128)	147584
dropout 1	(None, 48, 48, 128)	0
conv2d 7	(None, 48, 48, 128)	147584
conv2d transpose	(None, 96, 96, 64)	32832
concatenate	(None, 96, 96, 128)	0
conv2d 8	(None, 96, 96, 64)	73792
dropout 2	(None, 96, 96, 64)	0
conv2d 9	(None, 96, 96, 64)	36928
conv2d transpose 1	(None, 192, 192, 32)	8224
concatenate 1	(None, 192, 192, 64)	0
conv2d 10	(None, 192, 192, 32)	18464
dropout 3	(None, 192, 192, 32)	0
conv2d 11	(None, 192, 192, 32)	9248
conv2d transpose 2	(None, 384, 384, 32)	4128
conv2d 12	(None, 384, 384, 1)	33

Total trainable weights: 766,113

It is during training where the weights of the network are updated. To improve the model and ensure that the final predicted outcome corresponds to the best guess, the model is updated each time to find the smallest possible error between the prediction and the truth. Therefore a function is needed to quantifies how good or bad the approximation is. This is known as Loss function.

As dealing with binary classification application, the loss function chosen has been "Binary cross entropy". It is defined as seen in Equation 3.1, where "y" are the labels, "s" the score for each class and "C" indicates the number of classes.

$$CE = -\sum_{i=1}^{C=2} y_i \log(s_i) = -(y_1 \log(s_1) + (1 - y_1)\log(1 - s_1)) \tag{3.1}$$

Once the performance of the model is quantified, a procedure to steer the weights in the correct direction is needed. This is done in according to the values obtained from the loss function. To this end, "Adam optimizer [21]" has been chosen. It combines the advantages of two classical stochastic gradient descent algorithms: AdaGrad and RMSProp. Achieving a model that performs well on problems with sparse gradients and on non-stationary problems by maintaining the learning rate.

Below it is shown the code for programming the model. As it will be explained in more detail later, it has been used Python and Tensorflow to develop this task.

Algorithm 1: Build the model

```
def unet_v1_model_standard_v3 (width, height, channels):
    inputs = tf.keras.layers.Input((width, height, channels))
    s = tf.keras.layers.Lambda(lambda x: x/ 255)(inputs)

    # Contraction path
    c1 = tf.keras.layers.Conv2D(32, (3,3), activation='relu',
        kernel_initializer='he_normal', padding='same')(s)
    c1 = tf.keras.layers.Conv2D(32, (3,3), activation='relu',
        kernel_initializer='he_normal', padding='same')(c1)
    p1 = tf.keras.layers.MaxPooling2D((2,2))(c1)

    c2 = tf.keras.layers.Conv2D(64, (3,3), activation='relu',
        kernel_initializer='he_normal', padding='same')(p1)
    c2 = tf.keras.layers.Conv2D(64, (3,3), activation='relu',
        kernel_initializer='he_normal', padding='same')(c2)
    p2 = tf.keras.layers.MaxPooling2D((2,2))(c2)

    c3 = tf.keras.layers.Conv2D(128, (3,3), activation='relu',
        kernel_initializer='he_normal', padding='same')(p2)
    c3 = tf.keras.layers.Dropout(0.1)(c3)
    c3 = tf.keras.layers.Conv2D(128, (3,3), activation='relu',
        kernel_initializer='he_normal', padding='same')(c3)
    p3 = tf.keras.layers.MaxPooling2D((2,2))(c3)

    c4 = tf.keras.layers.Conv2D(128, (3,3), activation='relu',
        kernel_initializer='he_normal', padding='same')(p3)
    c4 = tf.keras.layers.Dropout(0.1)(c4)
    c4 = tf.keras.layers.Conv2D(128, (3,3), activation='relu',
        kernel_initializer='he_normal', padding='same')(c4)

# Expansive path

    u5 = tf.keras.layers.Conv2DTranspose(64, (2, 2),
        strides=(2, 2), padding='same')(c4)

    u5 = tf.keras.layers.concatenate([u5, p2])
    c5 = tf.keras.layers.Conv2D(64, (3, 3), activation='relu',
        kernel_initializer='he_normal', padding='same')(u5)
    c5 = tf.keras.layers.Dropout(0.1)(c5)
    c5 = tf.keras.layers.Conv2D(64, (3, 3), activation='relu',
        kernel_initializer='he_normal', padding='same')(c5)

    u6 = tf.keras.layers.Conv2DTranspose(32, (2, 2),
        strides=(2, 2), padding='same')(c5)
    u6= tf.keras.layers.concatenate([u6, p1])
    c6 = tf.keras.layers.Conv2D(32, (3, 3), activation='relu',
        kernel_initializer='he_normal', padding='same')(u6)
    c6 = tf.keras.layers.Dropout(0.1)(c6)
    c6 = tf.keras.layers.Conv2D(32, (3, 3), activation='relu',
        kernel_initializer='he_normal', padding='same')(c6)

    u7 = tf.keras.layers.Conv2DTranspose(32, (2, 2),
        strides=(2, 2), padding='same')(c6)

    outputs = tf.keras.layers.Conv2D(1, (1, 1),
        activation='sigmoid')(u7)

    model = tf.keras.Model(inputs=[inputs], outputs=[outputs])

    model.compile(optimizer='adam',
                  loss='binary_crossentropy',
                  metrics=['accuracy',
                  tf.keras.metrics.Recall(name="recall"),
                  tf.keras.metrics.Precision(name="precision")])
    model.summary()
    return model
```

3.3 Environment Setup

This section describes the resources used for the development of the model and its training. This processes consisted of, first designing the structure and then compiling it, following a supervised process through which the network adapts its parameters according to the corresponding right label.

3.3.1 Software Environment

A wide range of development software platforms for machine learning models are available, such as Tensorflow, Microsoft CNTK, Theano, Caffe, Pytorch, among others. These open source libraries facilitate the process of acquiring data, training networks, making predictions and refining future results.

Among all the frameworks, Tensorflow stands out as the leading tool in the Machine Learning sector. It offers an end-to-end platform available in different programming languages that can process your data, build and train Machine Learning models and serve the trained models across different devices. Therefore, **Tensorflow 2.3.1** has been the tool chosen for developing the model. The programming language used for this regard has been **Python 3.8**.

3.3.2 Hardware Environment

The technical specifications of the hardware environment are provided below. The environment used has been Intel^{TM} Core^{TM} i7-6500U. Further details can be found in Table 3.3

Deep learning training requires a large amount of computational power to run. This is because it is a process in which multiple matrix multiplications have to be performed. These operations can be performed either through the CPU (Central Processing Unit) by executing one operation after the other, or by making use of a GPU (Graphics Processing Unit), which executes the operations at the same time. The processor that gives the best results when training neural networks is the GPU, as it reduces the training time considerably.

Although the environment is GPU-enabled, the Tensorflow library only supports the use of GPUs for NVIDIA devices. Therefore this network has been trained on **CPU @ 2.50 GH**. Due to this inconvenient, the training procedure lasted several hours.

Overall, the study has been carried out in Python 3.8 through Spyder making use of Tensorflow in a device with a processor Intel Core i7-6500U CPU @ 2.50GHz. Table 3.3 gathers all the technical specifications of the environment used in this study.

Table 3.3: Technical specifications of the environment employed

CPU	Intel^{TM} Core^{TM} i7-6500U CPU @ 2.50 GHz, 8GB of RAM
Framework	Tensorflow 2.3.1
Language	Python 3.8

3.3.3 Final Training and test dataset

The structure of the network used is described in section 3.2. As mentioned in previous paragraphs, the network had to be trained on a CPU processor. This is not only detrimental to the training time, but also limits the workload that can be supported by this process. For this reason, it has not been possible to train the network with all the available images, making use of only 1000 images. These images have been randomly selected from the 8400 patches available. As the patches will be the input data to the network, they will be referred to as images.

Moreover, 800 of these images are used for training and 200 for validating the model. This partitioning has been done through the "train test split" attribute which randomly splits the dataset into two subsets, one consisting of 80% of the dataset and the other of the remaining 20%. Table 3.4 summarises the dataset finally used.

In addition, as it will be described in more depth in chapter 4, another set of 200 different images has been gathered to be used for performance measuring purposes. A summary of the final dataset used has been

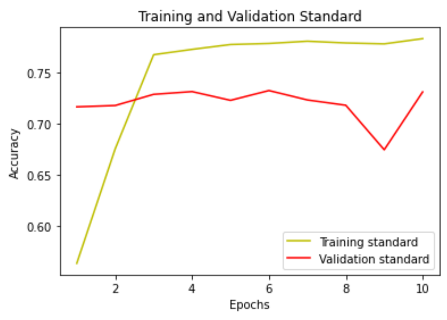
Table 3.4: Description of dataset finally used

Subset	Type of image	Images	Size
Training	Natural false color	800	384 x 384 x 4
	Ground truth	800	384 x 384 x 1
Validation	Natural false color	200	384 x 384 x 4
	Ground truth	200	384 x 384 x 1
Metrics evaluation	Natural false color	200	384 x 384 x 4
	Ground truth	200	384 x 384 x 1

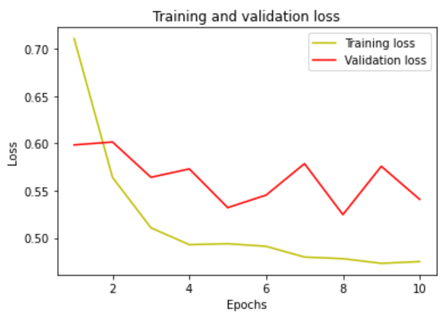
3.3.4 Training features

When training the model it is necessary to choose training features such as batch size and number of epochs. Epochs are the number of times the data will be processed through the network to learn from its main characteristics. That is, the algorithm will perform the forward and backward loop as many times as the number of epochs defined. Likewise, batch size is the number of examples that are introduced into the network each time it is executed.

At first, the algorithm was trained for 10 epochs and 16 batch sizes. The accuracy and loss results during the training procedure of the preliminary model can be seen in Figure 3.7. Analysing the graphs it was observed that the model was still very unstable and did not converge. This suggested that the model had to be trained for more epochs. Moreover, it was noted a significant difference between validation and training results. It was observed that the training was throwing better results than the validation, this is called overfitting. It means that the model is learning too specific features from the training dataset and it is not able to generalize when processing unseen data.



(a) Accuracy



(b) Loss

Figure 3.7: Evolution of Accuracy and Loss during training the first version of the model

To overcome this issues, it was decided to add some dropout layers to the network structure and to train it for more epochs. Thus the model finally used has been **trained for 50 epochs with a batch size of 32 units**.

The performance metrics have been tracked throughout the training phase using Tensorflow's build in visualizer: *"Tensorboard"*. The results of the latest version of the model are shown below.

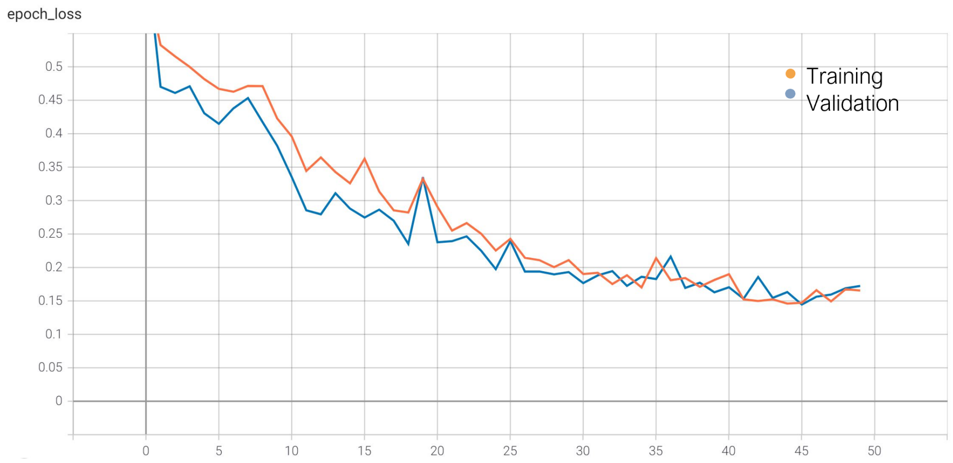


Figure 3.8: Loss

Figure 3.8 shows the decreasing loss over time. This indicates that the model is on the right track being able to minimize its loss as epochs are been executed. Moreover, it can be observed that towards the execution of the last epochs, training and validation losses are quite similar. This implies that the model has learned the features and evaluates in the same way already seen images and new ones.

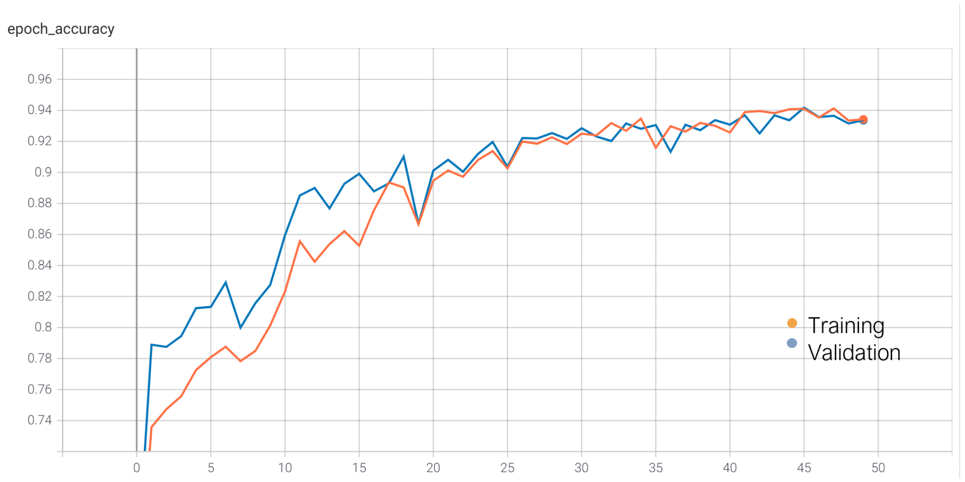


Figure 3.9: Accuracy

In Figure 3.9 it can be observed that at the end of the set number of epochs, the validation and training data shows similar results reaching 93,7% of accuracy. This indicates that the model was trained enough and that there is no overfitting or underfitting.

Other metrics were measured during the training phase such as precision and recall. The evolution of this metrics is represented in Figures 3.10 and 3.11

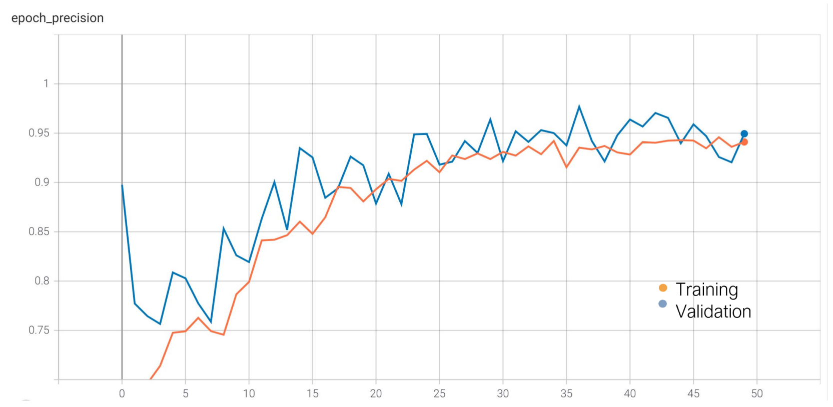


Figure 3.10: Precision during training

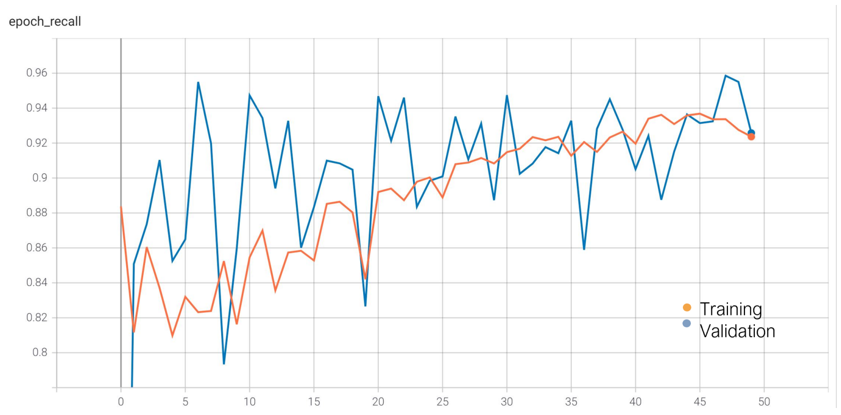


Figure 3.11: Recall during training

In the above images, it can be seen that towards the last trained epochs, precision and recall metrics reaches values up to 95% and 93% respectively. This reveals promising results showing that the model is capable of correctly identifying positive predictions, and has the ability to identify all the relevant cases within the dataset.

The process of training, analysing the results and adjusting the network has been carried out several times in order to arrive at the optimal model. Once satisfied with the results obtained, the calculated weights are saved. This concludes in a model ready to be deployed on any device compatible with machine learning applications.

Below is shown the code that executes the network training function and sets the parameters finally used.

Algorithm 2: Train the model

```
checkpointer = tf.keras.callbacks.ModelCheckpoint(
    'Unet_for_clouds_standard_v3.h5',
    verbose=1,
    save_best_only=True)
file_name = 'standard_model_v3'
callbacks = [tf.keras.callbacks.TensorBoard(
    log_dir='logs_standard_v3\\{}'.format(file_name))]

results_standard = model_standard.fit(
    X_train, Y_train,
    batch_size = 32,
    verbose=1,
    epochs=50,
    validation_data=(X_val, Y_val),
    callbacks = callbacks)
model_standard.save('Unet_cloud_standard_v3.hdf5')
```

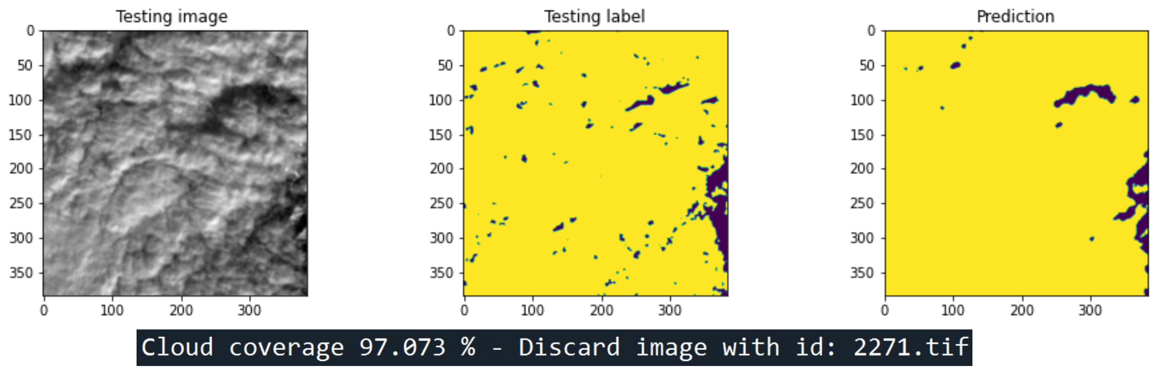
3.4 Cloud coverage assessment

In addition, as the ultimate goal was to determine which images should be discarded and which were useful, a final function has been added after the neural network. A cloud cover detector for the prediction images. To this end, the program will output a binary decision array that detects images with cloud cover above a set threshold. This decision selection works by analysing each pixel in the image and calculating an overall percentage of clouds in that patch. If the total number of positive pixels is greater than the threshold, the image is discarded. Otherwise, the image is considered useful and therefore should not be rejected.

To this end the cloud coverage metric was computed evaluating for each image, the total number of pixels classified as clouds, divided between the total number of pixels found in a 384 x 384 image. This equation was implemented as follows Eq. 3.2.

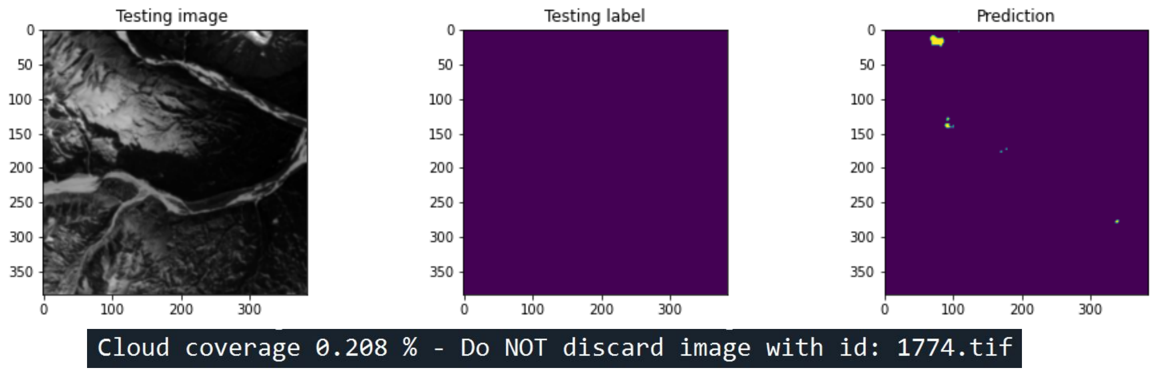
$$Cloudcoverage = \frac{Numberofpixelsclassifiedascloud}{Totalnumberofpixelsinanimage} \tag{3.2}$$

As a sample of the above, some images ranging from no clouds to full cloud cover are processed through the developed algorithm. The natural false colours and ground truths of the selected images can be seen in Figure 3.12 and Figure 3.13; as well as the result of the binary prediction and the final discard decision.



```
Cloud coverage 97.073 % - Discard image with id: 2271.tif
```

Figure 3.12: Example image 2271.tif: Discard



```
Cloud coverage 0.208 % - Do NOT discard image with id: 1774.tif
```

Figure 3.13: Example image 1774.tif: Do not discard

For the images seen above, the decision threshold was set to cloud coverage of 70%. However it can be modified to follow a more or less conservative decision. On one hand, the final goal is to discard useless images, so a low threshold should be selected. Nevertheless, the lower the threshold, the more possible useful images are discarded. This threshold should then be determined taking into account the final application of the image.

For a reference of what the different ranges of cloud cover would depict, Figure 3.16 displays some examples.

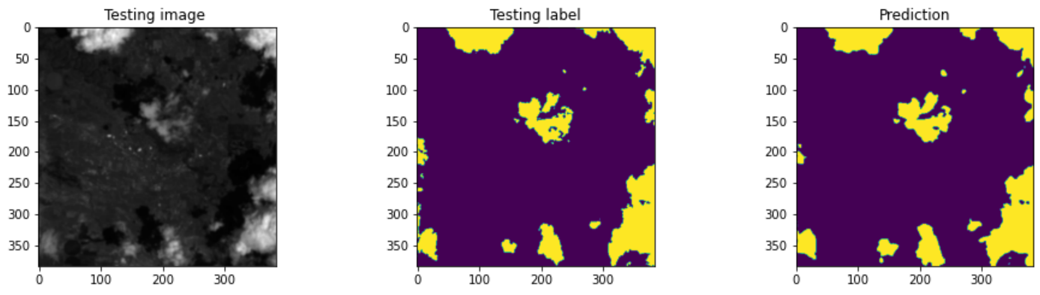


Figure 3.14: Cloud coverage: 15% example image: 1829.tif

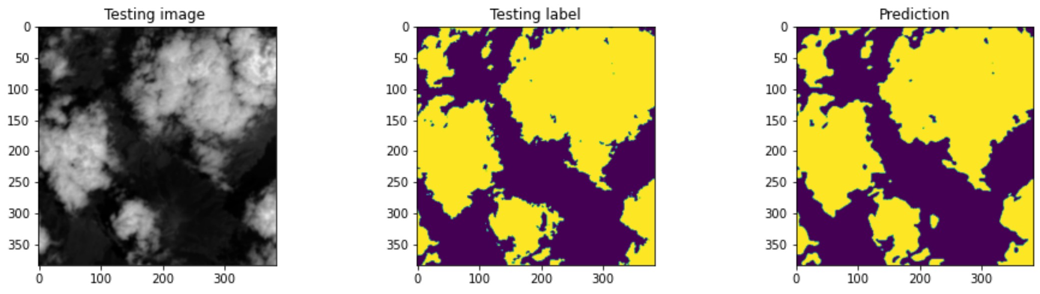


Figure 3.15: Cloud coverage: 55% example image: 1829.tif

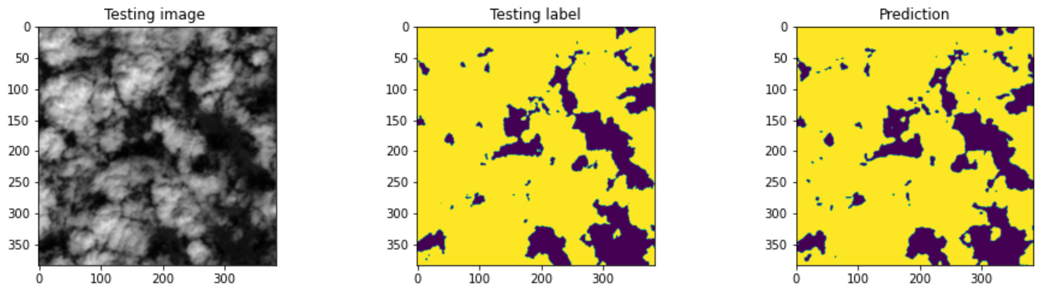


Figure 3.16: Cloud coverage: 80% example image: 1829.tif

Chapter 4

Results

Although some metrics are calculated during the training phase, a more in-depth analysis has been carried out to evaluate the performance of the model. For this purpose, 200 images not seen by the model have been processed through the network. These unseen scenes have been collected from the original dataset, as the images and their respective masks are available. Hence, some metrics have been calculated by comparing the obtained predictions with their ground truths. Quantitative and qualitative results are presented in this section.

The dataset used to test the model follows the cloud coverage distribution represented in Figure 4.1. As a sample, images with various types of possible situations have been used, from the near absence of clouds to situations of total cloud cover.

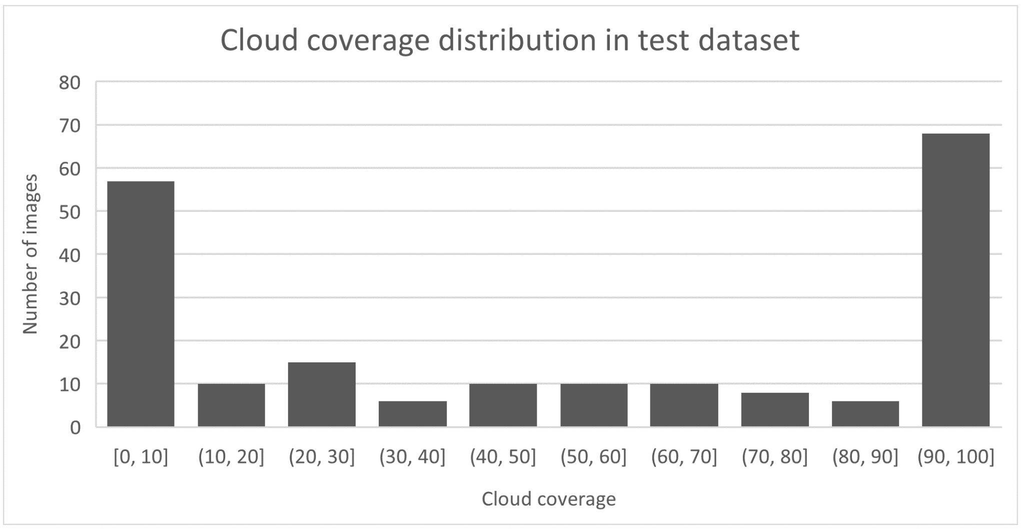


Figure 4.1: Cloud coverage distribution in test dataset

4.1 Quantitative results

To determine the amount of pixels correctly classified and the ones miss-classified, the model has been tested on 200 images that it has never seen before. A prediction was obtained for each image of the test dataset and it is then compared to their respective ground truth. This assessment is calculated pixel by pixel for each image. Based on the results obtained, some metrics can be computed to benchmark the model. It has been decided to study the Accuracy, Recall, Precision, Intersection over Union and Dice coefficient

For the sake of understanding, classified as *"Positive or 1"* stands for cloud, and *"Negative or 0"* stands for background. To identify the number of predictions correctly classified as either True or False, it is used *TP* and *TN* respectively. On the other hand, for the miss-classified predictions, *FP* stands for erroneously classified as positive and *FN* stands for erroneously classified as negative. Figure 4.2 grahically represents this concepts in a more understandable way. This table is known as confusion matrix.

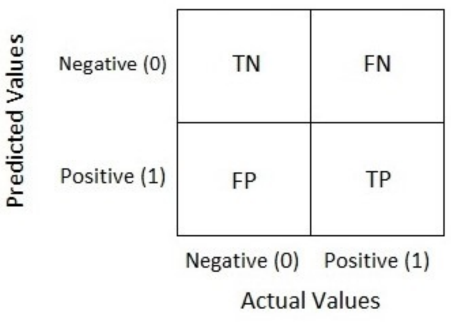


Figure 4.2: Graphical representation of TP, TN, FP and FN

As dealing with just a random sample of data, there will always be an instrinsical error of the mean. This is due to the fact that a sample represent a small portion of the entire population and therefore contains variability, lack of information and might not represent accurately the population as they might be biased. Moreover the final goal is to minimize test error, however the algorithm has been developed by reducing the training error loss. Therefore it is inevitable to encounter a training and test error.

An approach to analyze the error of the model through the acquisition of several data samples could have been: K-Means method. It randomly divides the dataset in K clusters and performs training including all of the clusters except one. Once the performance metrics are computed, the model is trained from scratch once again excluding one of the other clusters. In summary, it would be necessary to train the model K times. It has been decided to not use this procedure as it is highly computationally expensive and it would take several days to perform.

Hence, to estimate the error of the different performance metrics of a model, a bootstrapping[22] operation has been performed. Bootstrapping is a technique used to obtain a broader approximation of the error of a sample of data, by using random sampling with replacement. Once the performance metrics of each image of the trained model have been measured for each image in the validation dataset, the bootstraping procedure is applied. It takes the resulting values from the metrics and generates new sample measures. This will give more results without needing to execute the model several times. Therefore the values presented on the following subsection correspond to a result of applying bootstraping method.

4.1.1 Accuracy

Accuracy: Represents the amount of correctly identified pixels over the total amount of pixels of an image.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{Correct\ predictions}{Total\ predictions} \tag{4.1}$$

Computing the mean among all the 200 evaluated images, the accuracy reaches a mean value of 93,70% with standard deviation of 10,93%.

Although this metric is widely used in machine learning applications, it is not the best metric for highly imbalanced applications. Such as semantic segmentation, where a large part of the image in which the number of pixels showing the background may outnumber those containing clouds, and vice versa. In such cases, the model can achieve high accuracy just by classifying correctly the dominant class. Intersection over Union and Dice coefficient describes better the performance of such models, these metrics will be further explained below.

4.1.2 Precision

Precision: Shows the probability of being True Positive, given classified as Positive. In other words, gives a measure of over all the clouds detected, which ones were truly clouds.

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

The metric has been calculated for all predictions that contain some TP or FN pixel value. Due to the fact that, if the precision is calculated for images where the ground truth just contains background, this metric would yield value 0. Therefore, if the precision result of these images is part of the overall average, these results would lower the average result of the metric, showing a worse performance than it actually has.

Averaging all the pixels examined, it has been concluded to reach a mean of 88,46%, with a mean standard deviation of 23,15%.

4.1.3 Recall

Recall: Also known as *Sensitivity*, shows the probability of being classified Positive, given it is True Positive.

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

As is the case for precision, recall is just calculated for the images that contains at least one cloud pixel. Once again this is done to show the actual performance of the model.

After averaging all the measures obtained for each prediction, the mean value obtain is 83,35% with a standard deviation of 24,88%.

As briefly introduced above, for semantic segmentation applications, the following metrics help to more accurately identify the performance of the model. This is because they target pixels of the positive class.

4.1.4 Intersection over Union

Intersection over Union: Also known as Jaccard Coefficient, it is the metric that measures the area of overlap over the area of union between ground truth and prediction divided by the area of union. The output is a value between 0 and 1. The higher the score, the better, as it implies a higher match between the actual truth and the prediction within the total area covered. It is therefore a more precise measure than accuracy for semantic segmentation.

$$Jaccard(U,V) = \frac{|U \cap V|}{|U \cup V|} = \frac{TP}{TP + FP + FN} \tag{4.4}$$

The mean value obtained from averaging the results thrown by the 200 images, is a 63,50% score with a 41,64% standard deviation.

4.1.5 Dice Coefficient

Dice coefficient: Being mathematically the same as F1 score, is the harmonic mean of precision and recall. It corresponds to twice the area of overlap divided by the total number of pixels. It is similar to IoU, however as it gives more relevance to FP errors than FN errors, it better estimates the influence of FP within the network. Moreover, unlike IoU, it is easily differentiable.

$$Dice\ Coefficient = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} = \frac{2 * |U \cap V|}{|U \cup V| + |U \cap V|} \tag{4.5}$$

The mean value obtained averaging all the F1 scores of the different predictions is 66,99% with standard deviation of 42,43%.

4.1.6 Confusion matrix

To evaluate the pixel classification performance of the model on the background and cloud segmentation, a confusion matrix has been determined. This tool is commonly used to assess classification accuracy by compering predictions with labels. Figure 4.3 below shows the obtained results.

It has been calculated by comparing each pixel of the predictions with the ground truths. An average of all TP, TN, FP and FN predictions has been obtained for every pixel in each image. The following equation shows the calculation used.

$$Overall\ Confusion\ Matrix = \frac{Value}{200 * 384 * 384} \tag{4.6}$$

where Value = TP, TN, FP or FN

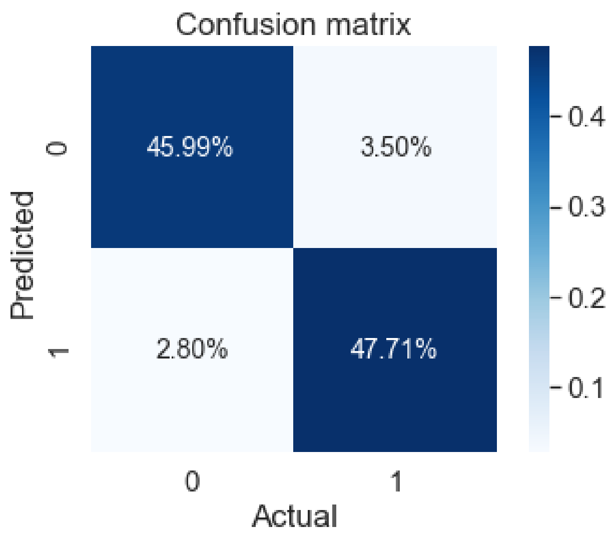


Figure 4.3: Confusion matrix

Depicted in the image, overall the model classifies each pixel as cloud and background with a high enough accuracy. Being able to classify TP and TN with almost equal ease. Regarding the miss classifications, the model shows a higher FN rate with a value of 3.50%, meaning that it tends to not identify clouds.

4.1.7 ROC curve

ROC curve Receiver operating characteristic; Represents a graphic comparison between Sensitivity(or false positive rate) and Specifity (or true positive rate). The area under this curve is known as AUC and ranges between 0.5 to 1, where 1 would represent a perfect model. The result corresponds to the probability of a model to distinguish correctly between the classes. As depicted in Figure 4.4, the model has obtained an AUC value of 0.94.

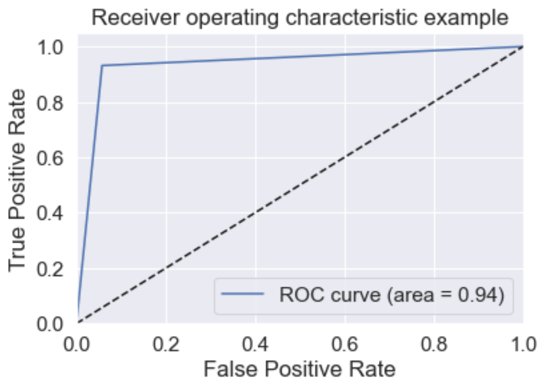


Figure 4.4: ROC Curve

To gather the measured results a summary of the average values is shown below.

Table 4.1: Recopilation average measurements

Metric	Mean(%)	Standard deviation(%)
Pixel Accuracy	93,70	10,93
Precision	88,46	23,15
Recall	83,35	24,88
Iou	63,50	41,61
Dice coefficient	66,99	42,43

4.2 Qualitative results

From the confusion matrix in Figure 4.3, it has been observed that taking into account both FP and FN results, around a 5% of the pixels are miss-classified. In order to asses the reason of the error, the erroneous predictions have been identified.

To this end, a confusion matrix for each image has been computed, spotting the images with the highest error rates. It must be highlighted that the confusion matrix has been computed by comparing each pixel of the predicted image with each pixel of the label.

For each scene analyzed it is given:

-Testing image: It is the natural false color image fed into the network.

-Testing label: It is the ground truth provided by the dataset. Yellow represents actual cloud and purple stands for actual background. During training, it is provided to the network and acts as the solution used to recalculate the weights.

Whereas, in the evaluation phase it doesn't interfere with the network. It is just used to evaluate the prediction's similarity with reality.

-Prediction: It is the result of the model displayed as an image. Yellow represents predicted as cloud and purple stands for predicted background.

4.2.1 Example FP case

Several False Positive samples have been found in between the results obtained. Meaning that the model predicted as positive truly negative pixels. An example with a high false positive rate is depicted below.

Image 4.6 stands out with a FP rate of 37.93%. Meaning that 55931 out of the 147456 pixels of an image where classified as positive when they are actually negative. The confusion matrix provided for this image can be seen in Figure 4.5.

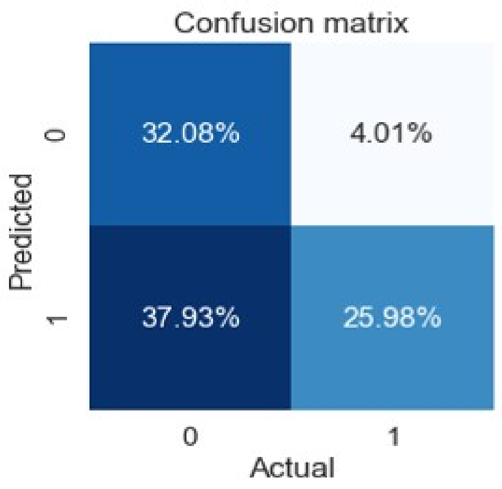


Figure 4.5: Confusion matrix of image 1910.tif

Note that this confusion matrix has been calculated for every pixel of just one image. By comparing the prediction with its corresponding ground truth. An average of all TP, TN, FP and FN predictions has been computed for each image. The following equation 4.7 shows the calculation used.

$$Individual\ Confusion\ Matrix = \frac{Value}{384 * 384} \tag{4.7}$$

where Value = TP, TN, FP or FN

A comparison of the false-colour image, its labelled mask and the prediction obtained can be seen in Figure 4.6. It can be seen that the model failed to classify the image predicting clouds when it is actual background.

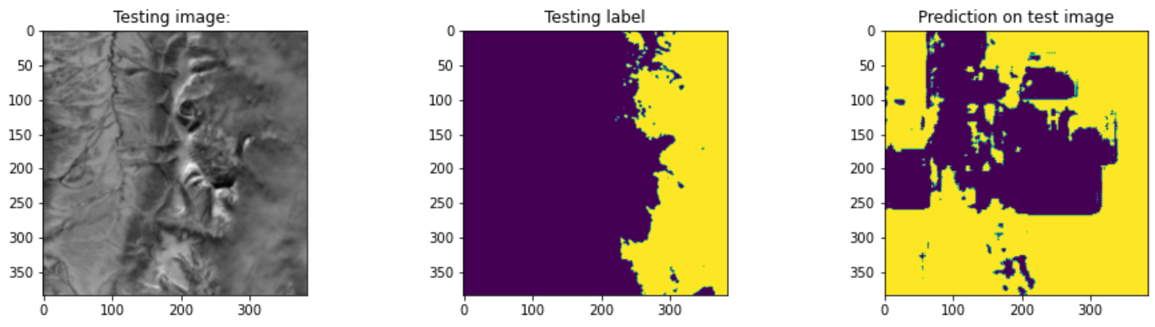


Figure 4.6: Comparison of image 1910.tif

4.2.2 Example FN case

Another flaw that can occur in binary classification models is to label data as negative when it is actually positive. This is known as False Negative. Some cases of FN have been detected among the assessments.

An image that stands out due to its high percentage of erroneously classified pixels is Figure 4.8. As observed, the algorithm has erroneously miss predicted background, not identifying all the clouds present. It is clearly depicted in its confusion matrix represented in Figure 4.7. Reaching a percentage of 32,40% of FN pixels. Once again this individual confusion matrix has been computed following equation 4.7.

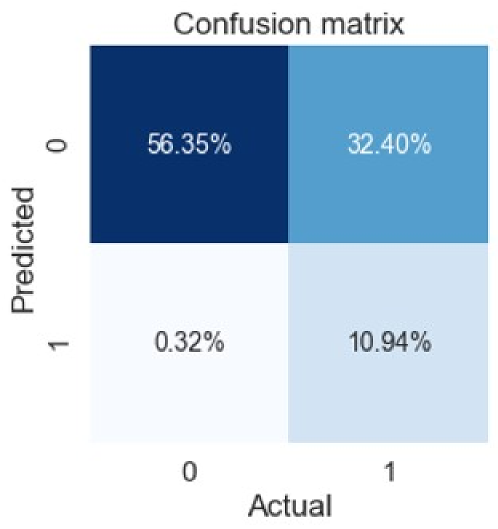


Figure 4.7: Confusion matrix of image 1836.tif

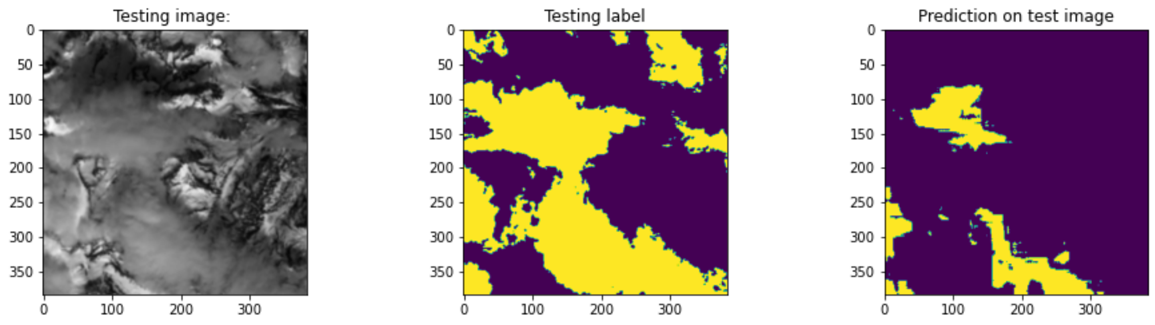
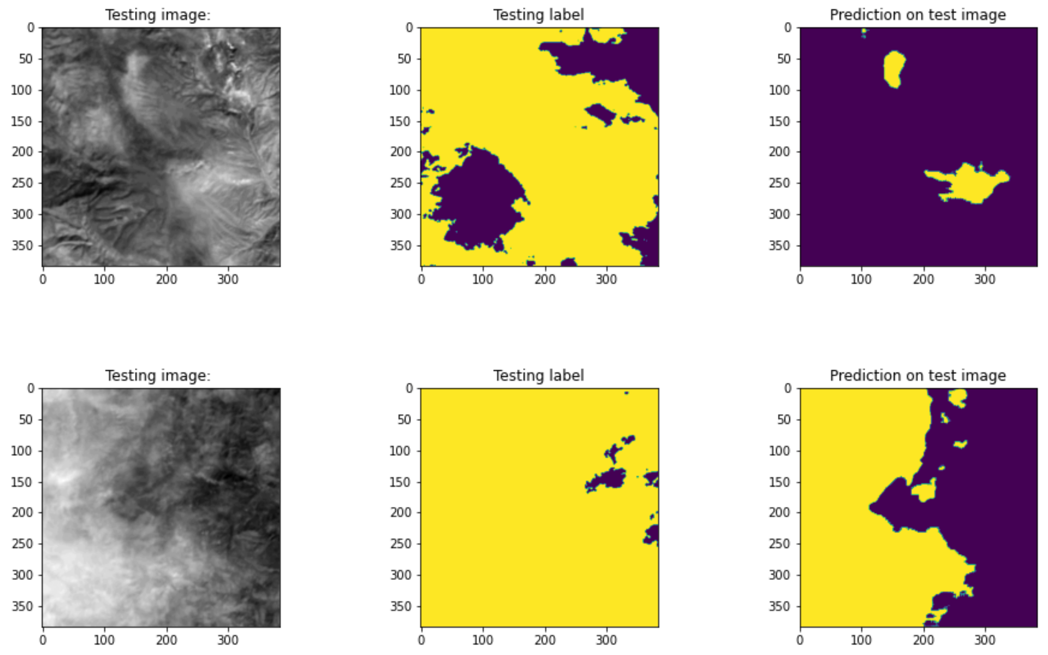


Figure 4.8: Comparison of image 1836.tif

Other False Positive or False negative errors have been identified and are displayed in the figures below. Analysing the results, it can be seen that most of these images contain snow or light clouds. Cases in which it is difficult to distinguish and can cause controversy even when analysed under human eye.

In order to optimise the model and successfully detect such cases, it is proposed to extend the number of classes to be analysed by the network. Broadening the segmentation to light clouds, thick clouds, snow and background.

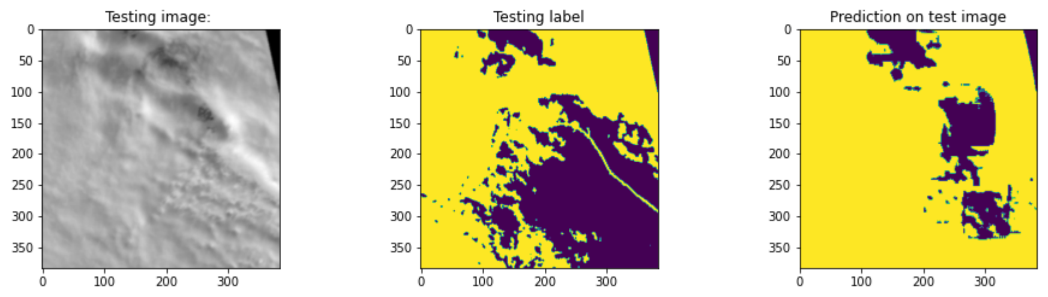


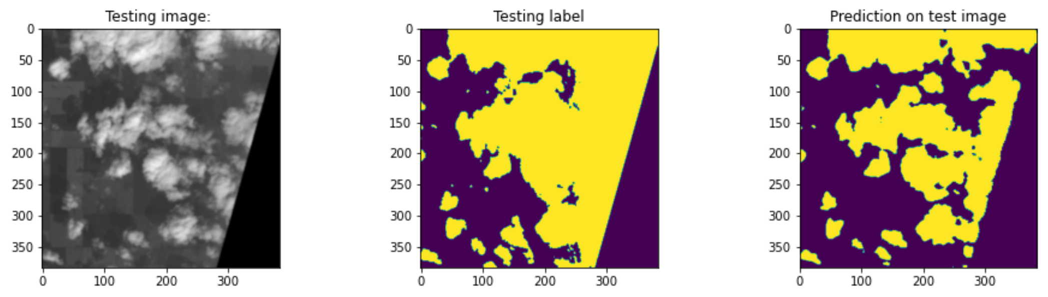
4.2.3 Missclasified dataset

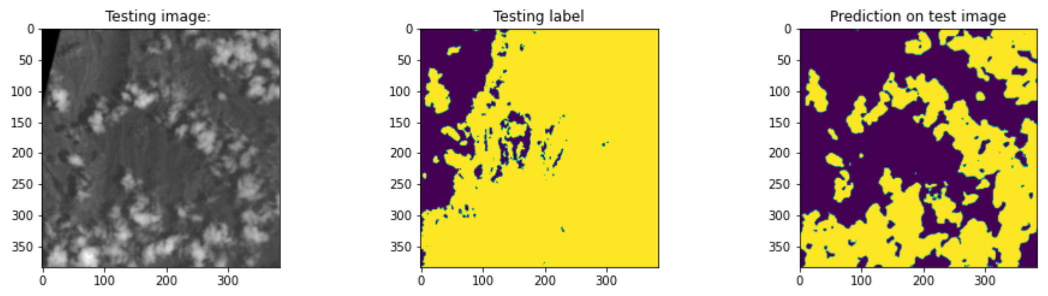
Analyzing the images that returned errors, it was found that many ground truths were mislabelled. An example of this is shown in the figures below. It can be observed how the "Testing labels" mark background pixels as clouds and vice versa.

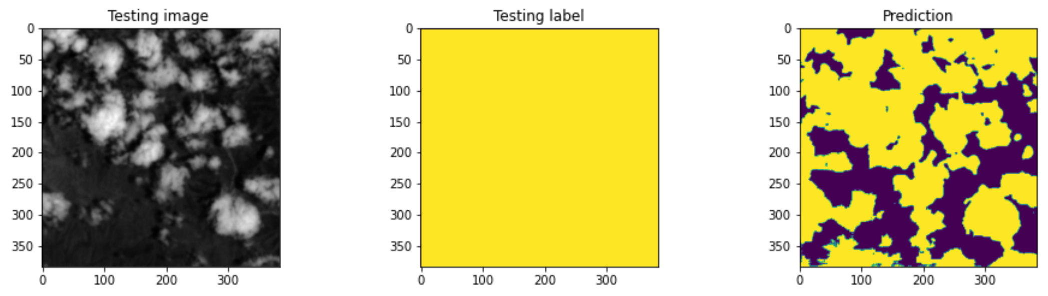
If erroneous labels have appeared in the testing dataset, it can be assumed that the training dataset will also contain errors, as they come from the same source. As a consequence, the model obtained will not be optimal. Furthermore, the metrics analysed have also been calculated with respect to the ground truths, so the results obtained will not be accurate.

Some of the mislabels found are displayed bellow in Figure 4.9. It can be seen the comparison between the studied image, the erroneous label and the prediction obtained from the model. Highlight that in most of these cases the model predicts better results than the provided labels.


Testing image:
Testing label
Prediction on test image
0
50
100
150
200
250
300
350
0
100
200
300


Testing image:
Testing label
Prediction on test image
0
50
100
150
200
250
300
350
0
100
200
300


Testing image:
Testing label
Prediction on test image
0
50
100
150
200
250
300
350
0
100
200
300


Testing image
Testing label
Prediction
0
50
100
150
200
250
300
350
0
100
200
300

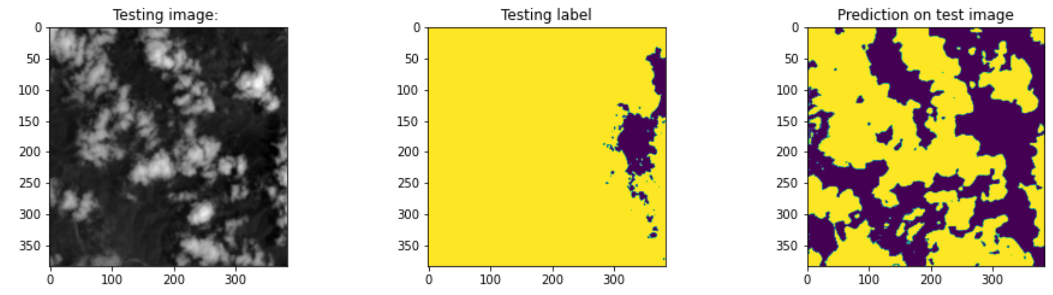


Figure 4.9: Mislabels compilation

Chapter 5

Conclusions and future work

5.1 Conclusions

The main objective has been to develop system that evaluates and determines the percentage of cloud cover in satellite images. The output of the solution was determined to be a divider between useful images that include few clouds, and images that should be discarded for having a high percentage of clouds. This goal has been successfully achieved by developing a deep learning algorithm capable of segmenting clouds in satellite imagery.

The choice to use machine learning to address the challenge, was due to its outstanding performance when classifing images. In order to understand all the components involved in this approach, a general overview of the field of machine learning and in particular the subfield of deep learning has been carried out. Thus the use of Convolutional Neural Networks and their application for object recognition purposes has been studied. These partial objectives have been achieved thanks to focused research in this field, with published conference papers, or articles in popular scientific journals.

Likewise, the singularities of satellite images have made it necessary to carry out a previous study of their processing. For this reason, this work compiles the basic details of multispectral images, detailing how the satellites capture the data according to the reflected frequency, obtaining different bands. This partial objective has been achieved by using the information provided by governmental agencies regarding satellites.

Once all the necessary background knowledge had been studied, the Convolutional Neural Network was developed. The model obtained is based on the UNet architecture. Although some modifications have been made to adapt it better to the final application. The network has been trained and tested using a labelled dataset containing Landsat images. The training was carried out using 1000 of these images.

Once the model was trained, a series of tests have been executed to evaluate its performance. This validation has been carried out on another 200 reserved images from the dataset. To obtain a more accurate result of the performance, the bootstrapping method has been employed. As a result, the model has shown a mean accuracy of 93,70%, showing lower average results in precision and recall, reaching 88,46% and 83,35% respectively. The evaluation of Intersection over Union metric reached a total mean value of 63,50% and a 66,99% of Dice coefficient. The confusion matrix and the ROC curve have also been plotted. This successfully completes the partial objective of evaluating the performance of a semantic segmentation algorithm.

In addition, a visual check has been carried out, comparing the images, their ground truths and the predicted results. It has revealed that miscassifications do not just correspond to model mistakes, but to wrongly labelled patches. Nevertheless, in most of the cases, the model outworks segmenting those images, predicting better results than the ground truths themselves.

Overall a model to segment clouds on satellite images has successfully been developed and its performance has been evaluated.

5.2 Future work

This study has generated interest in new approaches to work. Some of the lines of research that have been opened up are presented below:

- Improve the dataset, either by self-labelling the images or finding more reliable labelled data.
- Analyze how each band of a hyperspectral image influences the cloud detection to determine the bands where a higher amount of important features are identified. With this improvement, a more effective dataset for training would be arranged.
- Evaluate the model by performing K-Fold Cross Validation. This method proposes dividing the data in K different sets to then train the model several times. The accuracy could then be computed for every fold obtaining a more accurate evaluation of the model. This was too computational expensive for the available environment to be performed in this Thesis.

- Develop a more complex algorithm capable of differentiating between snow and clouds. For this purpose ground truths labelling the snow and clouds are needed.
- Self-label unseen images from a different source, such as Sentinel's images, and evaluate the performance of the model on them. This evaluation will allow to test how the model performs with completely new data.

Appendix A

Appendix

Table A.1 shows the compilation of the results obtained after evaluating the network. Accuracy, Intersection over Union, Precision, Recall and Dice coefficient have been measured. This evaluation has been carried out on 200 images previously unseen by the model. As a result, the metrics computed for each pixel of each image have been evaluated.

Note that precision and recall have just been computed for the images with at least one TP pixel. This is because the way this metric is computed assumes that all images contain TP pixels, however there are entire background images that do not include clouds. In those cases, precision and recall would throw values of 0, which would worsen the mean values of this metrics. For this reason, it has been chosen to evaluate this metrics on images with at least one cloud pixel.

Table A.1: Results

Accuracy	IoU	Dice coefficient	Precision	Recall
0,99883355	0	0	0,924494755	0,861727986
0,949293349	0	0	0,963308806	0,965858165
1	1	1	0,962785934	0,817154448
0,974385579	0,805068126	0,892008578	0,745801778	0,839644124
1	0	0	0,121791782	0,668264463
0,961466471	0,931586679	0,964581801	0,970767324	0,897879339
0,994981554	0,792134831	0,884012539	0,9662858	0,999616086
0,828450521	0,65281838	0,789945693	0,997466875	0,930434479
1	1	1	0,899140068	0,999965537
1	1	1	0,883813713	0,898076923
1	0	0	0,95620644	0,96781637
0,999871148	0	0	0,871229698	0,938202715
0,788682726	0,114848166	0,206033736	0,781848027	0,932061081

0,964850532	0,874241763	0,932901806	0,008841443	0,066404329
0,967610677	0,965927332	0,982668399	0,954304421	0,870658801
0,930223253	0,928241144	0,962785331	0,994617693	0,999828998
0,911682129	0,899112206	0,946876338	0,986937096	0,996314783
1	1	1	0,892921877	0,936406727
1	0	0	0,949067281	0,805264094
0,995985243	0	0	0,943547478	0,970134838
1	1	1	0,950710736	0,956847921
0,942593045	0,803244776	0,890888233	0,907094609	0,974503159
1	0	0	0,419559902	0,693343709
0,952779134	0,926738424	0,961976377	0,947122491	0,95419719
0,967352973	0,823946752	0,903476761	0,999368408	1
0,867004395	0,739693116	0,850371953	0,991450563	0,86157712
0,999898275	0,999898275	0,999949135	0,949844302	0,999692969
1	1	1	0,922878992	0,948704638
0,993231879	0	0	0,885663148	0,914623539
0,997890896	0	0	0,926410976	0,899057583
0,884494358	0,007863925	0,015605132	0,956009743	0,951736353
0,953009711	0,835813469	0,910564698	0,836224005	0,982776704
0,994466146	0,994448526	0,997216537	0,252398316	0,971683335
0,983357747	0,983347358	0,991603769	0,87229911	0,971378022
0,979593913	0,979593913	0,989691781	0,988685879	0,999882395
0,994045681	0	0	0,998400436	0,604076794
1	1	1	1	0,999674479
1	0	0	0,920911381	0,947203569
1	1	1	0,780753531	0,988750256
0,95042589	0,841870728	0,914147464	0,969892877	0,946762062
0,990926107	0,771905898	0,871271888	0,966180371	0,952569231
0,953708225	0,916914163	0,956656465	0,870099067	0,997744573
0,956732856	0,911624557	0,953769456	0,467412065	0,885473838
0,909478082	0,886065469	0,939591423	0,832758621	0,984827586
1	1	1	0,991723085	0,999667104
1	1	1	1	0,910705566
0,993706597	0	0	0,559703868	1
0,967556424	0	0	0,989723288	1
0,713168674	0,353890103	0,522775227	0,95418738	0,685618283
0,972025553	0,90593574	0,950646678	0,951343446	0,954727494
0,999369303	0,999368408	0,999684104	0,722209247	0,996472125
0,86380344	0,855223224	0,921962611	0,972824105	0,9761385
0,949571398	0,949567293	0,974131333	0,934494662	0,915604845
0,997938368	0	0	0,932965152	0,984911381
1	1	1	0,804854706	0,937349191

1	0	0	0,982972508	0,999421292
1	1	1	1	0,721339899
0,964179145	0,879016927	0,935613633	0,054019505	1
0,949239095	0,818033743	0,899910407	0,810681629	0,990153879
0,972961426	0,839130084	0,912529343	0,823498695	0,989956058
0,970391168	0,911805107	0,953868261	0,940342077	0,9938814
0,896952311	0,824146192	0,903596648	0,994800707	0,905235892
1	1	1	0,967652271	0,999913831
1	1	1	0,974546587	0,996348169
0,877407498	0	0	0,77093925	0,99742354
0,993489583	0	0	0,913787627	0,98107265
0,672844781	0,250555392	0,400710585	0,9507703	0,748113311
0,970533583	0,850440589	0,919176324	0,984205804	0,99994433
0,988667806	0,98857092	0,994252617	0,883283821	0,99894409
0,641235352	0,60349273	0,752722751	0,801721736	0,989810564
0,968811035	0,968811035	0,984158477	0,521854808	0,147721771
0,999369303	0	0	0,995720148	1
1	1	1	0,866365076	0,406443329
1	0	0	1	0,999782986
0,999674479	0,999674479	0,999837213	0,84930213	0,997073952
0,959628635	0,875948154	0,933872455	0,860539552	0,990417212
0,894632975	0,773879	0,872527382	0,241091493	0,081870708
0,969455295	0,919731965	0,958187895	0	0
0,964809842	0,921832397	0,959326524	1	0,998440213
0,938164605	0,868391045	0,929560273	0,950349064	0,973924827
1	1	1	0,998199755	0,999992728
1	1	1	0,999304439	0,999985032
0,942030165	0	0	0,825075416	0,991360477
0,984910753	0	0	0,867312066	0,985571541
0,812594944	0,440765775	0,611849313	0,439485628	0,170556289
0,97591824	0,822209983	0,902431653	0,996757674	1
0,991516113	0,991395675	0,995679249	0,550632234	0,999641084
0,910705566	0,910705566	0,953266251	0,976393558	0,980065135
0,568440755	0,559703868	0,717705302	0,949221909	0,952333809
0,971672906	0	0	0,800746506	0,973920603
0,989807129	0,989723288	0,994835105	0,241876577	0,92087156
1	0	0	0,008661417	0,000215648
0,699991862	0,663768336	0,797909567	1	0,995313856
0,974344889	0,910278911	0,953032466	0,98230841	0,99969992
0,845336914	0,72036735	0,837457593	0,934417747	0,972660102
0,972520616	0,950227245	0,974478484	0,936705343	0,996995776
0,961568197	0,860384331	0,924953319	0,894899653	0,992350628

0,950066461	0,919818358	0,958234777
1	1	1
1	1	1
0,994818793	0	0
0,999966092	0	0
0,839477539	0,76376774	0,86606385
0,999613444	0	0
0,982598199	0,982413335	0,991128659
0,727511936	0,721339899	0,838114424
0,282335069	0,054019505	0,102501908
0,981614855	0	0
1	1	1
1	1	1
0,99810791	0	0
1	0	0
0,845411513	0,804198627	0,891474603
0,990397135	0,816675298	0,899087799
0,946024577	0,934929771	0,966370754
0,903876411	0,900973221	0,947907325
0,988098145	0	0
0,752855089	0	0
0,968349881	0,967571586	0,98351856
1	1	1
0,98550076	0	0
0,972866482	0	0
0,971564399	0,971077972	0,985326796
0,933119032	0,769407033	0,869677829
0,924621582	0,89795731	0,946235519
0,783759223	0,720214801	0,837354498
0,983093262	0	0
0,714775933	0	0
0,984307183	0,984151879	0,992012647
0,988254123	0,988254123	0,994092366
0,914883084	0	0
0,998074002	0	0
1	1	1
0,915398492	0,882459909	0,937560375
0,85422092	0,795159139	0,885892645
0,875495063	0,130111348	0,230262882
0,999891493	0	0
0,342115614	0	0
0,995720757	0,995720148	0,997855485

0,919542854	0,750024825
0,970189377	0,890698498
0,970178744	0,874430548
0,85258274	0,981973637
0,845880575	0,98292477
0	0
1	0,99742296
0,901611198	0,994962857
0,944984457	0,999835926
0,878111941	0,990894839
0,935867726	0,897965727
0,586371078	1
0,977870488	0,842738784
0,968567725	0,968431284
0,863070539	0,952708119

0,957227919	0,957227919	0,978146602
0,580600315	0,382465425	0,553309209
0,999938965	0	0
0,999782986	0,999782986	0,999891481
0,960401747	0,8471906	0,917274698
0,905748155	0,85343373	0,920921764
0,926852756	0,065094912	0,122233073
1	0	0
0,961690267	0	0
1	1	1
0,988186306	0,988186306	0,994058055
0,998440213	0,998440213	0,999219498
0,984714084	0	0
0,928365072	0,926768395	0,961992523
0,998311361	0,998192509	0,999095437
0,99935574	0,999289491	0,999644619
0,970113118	0,819185164	0,900606689
0,918931749	0,856437724	0,922667874
0,903252496	0,140084388	0,245743893
1	0	0
0,997823079	0,996757674	0,998376205
0,629889594	0,550523394	0,71011298
0,964748806	0,957379818	0,978225901
0,958794488	0,906169408	0,950775313
0,9434611	0,783937179	0,878884289
0,859707303	0,236951791	0,383122111
0,645609538	0,000210454	0,000420819
0,995313856	0,995313856	0,997651425
0,998345269	0,998345269	0,999171949
0,982211643	0,982018852	0,990927862
0,998114692	0	0
0,914530436	0,910503405	0,95315549
1	1	1
1	1	1
0,948371039	0,934068884	0,965910668
0,929531521	0,88876876	0,941109127
0,827568902	0,703835715	0,82617791
1	0	0
1	1	1
0,629075792	0,629075792	0,772310036
0,924140082	0,866971113	0,928746146
0,975036621	0,85154265	0,919819643

0,964246962	0,839444512	0,912715232
0,900743273	0,833630773	0,909267869
0,793287489	0	0
0,99742296	0,99742296	0,998709818
0,99983724	0,99983724	0,999918613
0,933308919	0,89751446	0,945989587
0,993259006	0	0
1	1	1
1	1	1
1	1	1
0,946919759	0,944837939	0,97163668
0,920701769	0,871083328	0,931100518
0,892740885	0,845912532	0,916525044
1	0	0
1	1	1
0,607625326	0,586371078	0,739260928
0,923502604	0,826967326	0,905289673
0,969672309	0,938922957	0,9684995
0,976494683	0,827613648	0,905676808

Bibliography

[1] M. D. King, S. Platnick, W. P. Menzel, S. A. Ackerman, and P. A. Hubanks, "Spatial and temporal distribution of clouds observed by modis onboard the terra and aqua satellites," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 51, no. 7, pp. 3826–3852, 2013. Accessed: 2023-04-01.

[2] E. D. Alanytics, "Satellite imagery and spectral band combinations." `https://eos.com/es/make-an-analysis/`. Accessed: 2022-10-18.

[3] U. E. Explorer. `https://earthexplorer.usgs.gov/`. Accessed: 2022-08-10.

[4] SAS, "The rise of computer vision." `https://www.sas.com/sas/offers/19/computer0vision.html?utm_source=bing&utm_medium=cpc&utm_campaign=ana-gen0emea_50745&msclkid=fe0278131e23189d463526e2ac6ef536&utm_source=bing&utm_medium=cpc&utm_campaign=EMEA-AI-PATHFINDER-BING0SPAIN-2020-Combined-EN0Bing+Only&utm_term=computer+vision&utm_content=SPECIFIC+-+50745-+Computer+Vision`. Accessed: 2022-10-21.

[5] S. U. S. of Engineering, "Lecture 2 - image classification." `https://www.youtube.com/watch?v=OoUX-nOEjG0&list=PLC1qU-LWwrF64f4QKQT-Vg5Wr4qEE1Zxk&index=2`. Accessed: 2022-10-21.

[6] A. Sabharwal and B. Selman, "S. russell, p. norvig, artificial intelligence: A modern approach, third edition.," *Artif. Intell.*, vol. 175, pp. 935–937, 04 2011.

[7] P. Ongsulee, "Artificial intelligence, machine learning and deep learning," in *2017 15th International Conference on ICT and Knowledge Engineering(ICT&KE)*, pp. 1–6, 2017.

[8] A. Muñoz, "Machine learning and optimization," 2012.

[9] R. Kohavi and F. Provost, "Glossary of terms. special issue of applications of machine learning and the knowledge discovery process," *Mach. Learn.*, vol. 30, 01 1998.

[10] J. Heaton, “Ian goodfellow, yoshua bengio, and aaron courville: Deep learning,” *Genetic Programming and Evolvable Machines*, vol. 19, p. 305–307, 04 2018.

[11] Stanford, “Image classification: Data-driven approach, k-nearest neighbor, train/val/test splits.” `https://cs231n.github.io/classification/`. Accessed: 2022-10-21.

[12] Stanford, “Neural networks part 1: Setting up the architecture.” `https://cs231n.github.io/neural-networks-1/`. Accessed: 2022-10-21.

[13] C. M. y. S. R. K.Mehrotra, “Elements of artificial neural networks,” pp. 1–6, Boston: MIT Press, 1997.

[14] F. Burghardt and R. Garbe, “Introduction of artificial neural networks in emc,” in *2018 IEEE Symposium on Electromagnetic Compatibility, Signal Integrity and Power Integrity (EMC, SI & PI)*, pp. 165–169, 2018.

[15] Stanford, ““convolutional neural networks: Architectures, convolution / pooling layers.” `https://cs231n.github.io/convolutional0networks/`. Accessed: 2022-10-21.

[16] Y. Xu, Y. Chi, and Y. Tian, “Deep convolutional neural networks for feature extraction of images generated from complex networks topologies,” *Wireless Personal Communications*, vol. 103, 11 2018.

[17] C. O. A. Hub. `https://scihub.copernicus.eu/dhus/#/home`. Accessed: 2022-08-10.

[18] G. Earth. `https://earth.google.com/web/data=CiQSIhIgOGQ2YmFjYjU2ZDIzMTFlOThiNTM2YjMzNGRiYmRhYTA`. Accessed: 2022-08-10.

[19] Kaggle, “Satellite imagery and spectral band combinations.” `https://www.kaggle.com/datasets/sorour/38cloud-cloud-segmentation-in-satellite-images`. Accessed: 2022-08-10.

[20] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation.” urlhttps://arxiv.org/abs/1505.04597, 2015.

[21] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization.” `https://arxiv.org/abs/1412.6980`, 2014.

[22] A. Rodríguez, “Introduction to bootstrapping in data science — part 1.” `https://towardsdatascience.com/introduction-to-bootstrapping-in-data-science-part-1-6e3483636f67`. Accessed: 2022-10-21.