FEDERAL UNIVERSITY OF OURO PRETO

INSTITUTE OF EXACT AND BIOLOGICAL SCIENCES

COMPUTER DEPARTMENT

DIEGO HENRIQUE MARQUES MATOS

Supervisor: Prof. Tiago Garcia de Senna Carneiro

Co-supervisor: Dr. Alber Hamersson Sánchez Ipia

AUTOMATED WORKFLOW FOR CLOUD SEGMENTATION TASKS USING DEEP LEARNING

Ouro Preto, MG

2022

FEDERAL UNIVERSITY OF OURO PRETO
INSTITUTE OF EXACT AND BIOLOGICAL SCIENCES
COMPUTER DEPARTMENT

DIEGO HENRIQUE MARQUES MATOS

AUTOMATED WORKFLOW FOR CLOUD SEGMENTATION TASKS USING DEEP LEARNING

Monography presented to the Course of Computer science from the Federal University of Ouro Preto as part of the necessary requirements to obtain the degree of Bachelor of Computer Science.

Supervisor: Prof. Tiago Garcia de Senna Carneiro
Co-supervisor: Dr. Alber Hamersson Sánchez Ipia

Ouro Preto, MG
2022

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E BIOLOGICAS
DEPARTAMENTO DE COMPUTACAO

FOLHA DE APROVAÇÃO

Diego Henrique Marques Matos

AUTOMATED WORKFLOW FOR CLOUD SEGMENTATION TASKS USING DEEP LEARNING

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal
de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Ciência da Computação

Aprovada em 13 de Junho de 2022.

Membros da banca

Tiago Garcia de Senna Carneiro (Orientador) - Doutor - Universidade Federal de Ouro Preto
Alber Hamersson Sánchez Ipia (Coorientador) - Doutor - Instituto Nacional de Pesquisas Espaciais
Rodrigo César Pedrosa Silva (Examinador) - Doutor - Universidade Federal de Ouro Preto
Eduardo José da Silva Luz (Examinador) - Doutor - Universidade Federal de Ouro Preto

Tiago Garcia de Senna Carneiro, Orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 13/06/2022.

Documento assinado eletronicamente por **Tiago Garcia de Senna Carneiro**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 13/06/2022, às 13:26, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0343413** e o código CRC **EB37D630**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35400-000
Telefone: 3135591692 - www.ufop.br

Acknowledgment

I thank Professor Tiago Garcia de Senna Carneiro and Dr. Alber Sanchez Ipia, who embraced the project and dedicated much of their time and effort to guide me during the execution of this work. I also thank professors Eduardo José da Silva Luz and Rodrigo Cesar Pedrosa Silva, who were responsible for introducing me to the area of artificial intelligence and showing everything it can created, as well as inspiring me to this work to be created and contributing to tips and solutions for solving the challenges encountered. I want to thank all my friends, especially those who somehow supported me during my graduation. I thank my girlfriend Andressa, for the years of teaching, affection and countless advices for the implementation of this work. Finally, and most importantly, I want to thank my parents who never measured efforts to make my dreams come true, we know it was never easy, but their support has always been my greatest strength.

Abstract

This work develops an approach that uses deep learning to segment clouds in images from WFI remote sensor. This work is justified by the fact that the remote sensor WFI does not have a tailor-made cloud segmentation algorithm, and this diminishes the quality and amount of data available for analysis. We developed a workflow and an algorithm based on the U-net neural network to fulfill this gap. The collected results presented are promising according to our evaluation. Issues like, false positives on segmentation, loss of edges in cropped images and biased dataset were visually evaluated and managed to segment more clouds, but with some misclassification. These problems motivated proposals for improvements to be made in future work. Therefore, the open source and automated workflow for semantic segmentation of remote sensing imagery remains as the main technological contribution of this work. The proposed workflow was evaluated in segmenting clouds from several training images larger than 2 gigabytes each, on a computer with only 8 gigabytes of RAM.

Keywords: Deep learning, Cloud segmentation, Remote sensing, imagery segmentation.

List of Figures

Figure 1.1 – Example of semantic segmentation 2
Figure 3.1 – Demonstration of the image cropping step. On the left the raw scene, in the center the raw scene with the subscenes in red, on the right the resulting subscene. 8
Figure 4.1 – Developed workflow 9
Figure 4.2 – Screenshot of the image generated by a remote sensor overlapped on a map on the INPE website 10
Figure 4.3 – U-net Architecture: The fist half of its architecture is the encoder, the second half is the decoder. 12
Figure 5.1 – Evolution of model along the training step 17
Figure 5.2 – Some input data with the respective predicted mask for the U-net model . . 18
Figure 5.3 – Some input data with the respective predicted mask that present a case of false positive cloudy pixels. 19
Figure 5.4 – Raw Scene. 19
Figure 5.5 – Mosaic mask. 19
Figure 5.6 – Mosaic mask. 21
Figure 5.7 – Mosaic mask with threshold. 21

List of Abbreviations and Acronyms

ABNT	Brazilian Association of Technical Standards
DECOM	Computer Department
UFOP	Federal University of Ouro Preto
INPE	National Institute for Space Research
WFI	Wide Field Camera

Contents

1 INTRODUCTION 1
1.1 Motivation 1
1.2 Problem Statement 1
1.3 Goals of This Work 2
1.4 Work Structure 4

2 RELATED WORK 5

3 DATASET 8

4 METHODOLOGY 9
4.1 Model Calibration 10
4.1.1 Handling Dataset 10
4.1.2 Training Step 11
4.1.2.1 Deep Learning Model 11
4.1.2.2 Metrics 12
4.1.3 Test Step 13
4.2 Production Model 13
4.2.1 Handling Data 14
4.2.2 Segmentation Step 14

5 RESULTS 15
5.1 Materials 15
5.2 Hyperparameters 15
5.2.1 Performance metrics 16

6 FINAL CONSIDERATIONS 22

References 23

1 Introduction

For living in a country that holds most of the largest forests in the world, it is a great desire and privilege to be able to create technologies that help prevent the continuation of deforestation in the Amazon rainforest. The tool proposed in this work should be able to help quantify and identify deforestation in the Amazon rainforest. Thus, the code of this project will be kept open for better replicability and for community contributions. The code can be found at the following: https://gitlab.com/ufopterralab/projeto-segmentacao-semanticas-cbers-4a-wfi.

Finally, there is a desire to create useful tools using the most advanced technology concepts. Applying this ideas to geoscience means identifying and quantifying environmental changes as deforestation and urban growth without need of manual processing. This way, this work proposes an automate and decoupled workflow for cloud segmentation in remote sensor images using a deep learning algorithm.

1.1 Motivation

A step to quantify and to identify deforestation in images from remote sensors is identifying clouds. On average, the cloud cover on earth surface is between 58% (ROSSOW; SCHIFFER, 1999) and 66% (ZHANG et al., 2004). When the object of analysis of a remote sensor image is on earth surface, like forests and cities, then present clouds become obstacles between the remote sensor and the object, which hinder its observation. Thus, detecting and removing clouds are essential steps to remote sensor imagery analysis (ZHU; WOODCOCK, 2012).

1.2 Problem Statement

The process of handling data for segmenting remote sensing images is a challenge. When the algorithm handles remote sensing images some metadata must be preserved. Each scene that is captured by the remote sensor is geographically referenced. This allows that when the scene is analysed with a geographic information system (GIS) the subscene will be rendered at the correct coordinates. There are images from several remote sensors with different characteristics in the captured scenes. The workflow to build a deep learning model to segment tasks has many processes in common, even if the input data comes from different sensors. This way, the main issue that must be resolved in this work is to create a workflow that handles geolocalized images for a neural network and, finally, to obtain the output images of the neural network to build a segmented scene. Lastly, the proposed workflow needs to be able to run on a personal machine, with less computational power, the scenes must be cropped to be loaded in memory in batches at the training step. This way, even when cropping the scene it is possible to re-built the same scene

with the subscenes. Thus, the main problem of this case study is to automatically handle the data to perform a semantic segmentation task.

Semantic segmentation consists of labeling each pixel in the image. This is exemplified in the Figure 1.1.

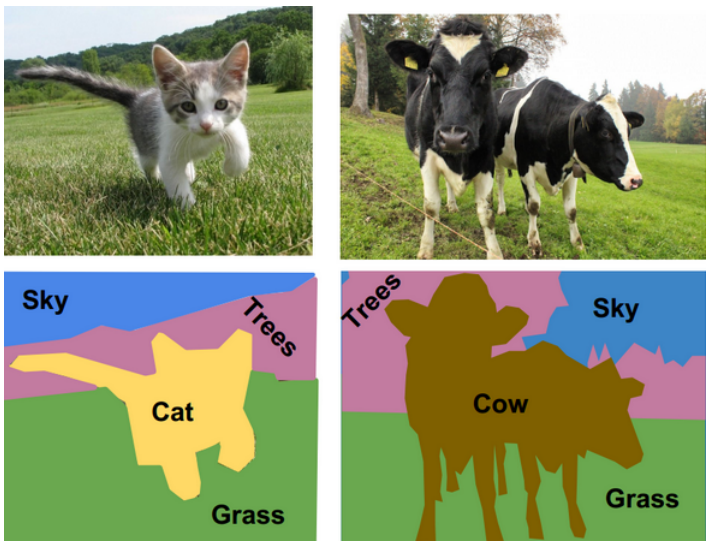

Figure 1.1 – Example of semantic segmentation

The original image is inputted into a machine learning model, then the machine learning model output other images of the same dimensions with the labeled pixels, called "masks".

Therefore, the semantic segmentation problem can be defined as the following: given an image, for each pixel present in the image, determine to which class the the pixel belongs. Where i is the quantity of rows of the image, j is the quantity of columns of the image, and k is the quantity of band of the image. Formally, let I be a remote sensor image so that I = {X ijk | i=1,...N, j=1,...M, k=1,...L}, the problem is to determine which pixels belong to each class.

1.3 Goals of This Work

The general goal of this work is to develop and evaluate an open source workflow that receives remote sensor images and their respective masks to build a deep learning model that can segment images even by using a computer that does not have so powerful hardware.

The specific goals of this work are:

- Build an automated workflow of segmentation tasks that can run on a personal computer.

  An automated workflow for deep learning tasks consists of treating the data, in this case study they are remote sensor images, defining some hyperparameters and training the deep learning model. Finally, with the model trained, it is sufficient to apply new data to collect results.

Machine learning tasks, mainly deep learning tasks, require powerful graphics processing units (GPU), and when remote sensor images are used, they also require a lot of random access memory (RAM) (GÉRON, 2019). Powerful machines that handle big data are inaccessible to everyone, so the proposed workflow configures some parameters so that it can be implemented on less powerful personal machines.

- Applying concepts of software engineering to build a workflow using component-based architecture with easy deep learning model configuration (CHAUDRON; LARSSON; CRNKOVIC, 2005).

  The proposed workflow uses a component-based architecture to allow exchanges of the deep learning model and input images. Also, the workflow should be an easy-to-use framework. The framework must have default settings and it must be possible to change these settings through the UI. Finally, the workflow must be decoupled so that more experienced users can manipulate any section of it and contribute to the developed code.

- Studying deep learning and its applications to remote sensing data.

  Machine learning is the science of programming computers so they can learn from data (GÉRON, 2019). The applications of machine learning are increasing fast in different areas, like biomedicine (RONNEBERGER; FISCHER; BROX, 2015), agriculture (CHEN et al., 2020) and remote sensor image analysis (JEPPESEN et al., 2019).

  Deep learning is a field of machine learning that learns patterns and uses them to perform described tasks, such as identifying clouds (JEPPESEN et al., 2019) and simulating human behavior (XU et al., 2017) as in chatbots. Since 1943, with the first artificial neuron implemented, (MCCULLOCH; PITTS, 1943) and since 1957, with the Perceptron algorithm (ROSENBLATT, 1958), there were networks built with layer composed by some connected perceptrons. This architecture is called "Multilayer Perceptron"and is a type of neural network. Neural networks are models that try simulating the learning from neurons through artificial neurons. The neural networks are getting better and better, decreasing the computational costs and increasing the accuracy of the results (POUYANFAR et al., 2018).

- Conducting a case study applying the deep learning algorithm called U-Net for segmenting clouds in CBERS 4A WFI remote sensor images.

  In this case study, a collection of results obtained from an U-net architecture applied to CBERS-4A WFI remote images is discussed with the goal of evaluate the proposed workflow in cloud segmentation tasks. Images from CBERS-4 Wide Field Camera (WFI) sensor have been chosen because they were designed to monitor Amazon rain forest (SOUZA et al., 2019) and track back deforestation and forest fires. CBERS-4A is capable of making revisits in five days to a certain area (EPIPHANIO, 2011). This feature makes it a good choice for deforestation monitoring tasks.

Open policies turned it easier to obtain imagery from remote sensing programs. Since 2004, about 1.600.000 images from CBERS-4 were free distributed to academic institutions, government agencies and companies. This open data policy was originated by National Institute for Space Research (INPE) with CBERS program. This fact influenced the USGS (United States Geological Survey) and ESA (European Space Agency) to provide images from Landsat and Sentinel programs, respectively (INPE, 2020). Thus, agricultural (MORAN et al., 2002) and environmental studies (SOUZA et al., 2019) were able to use free remote sensor images in their activities and encourage researchers on these fields.

There are some systems to quantify and identify deforestation using WFI images, like PRODES and DETER (SOUZA et al., 2019). However, even being one of the first to disclose its data, CBERS-4A, a satellite designed by Brazil and China, does not have its own algorithm for cloud semantic segmentation. Currently, INPE uses an algorithm called Cmask(QIU; ZHU; WOODCOCK, 2020) to segment clouds. Nevertheless, Cmask has been originatly develop for processing imagens from Landsat remote sensors and it is not implemented to any remote sensor present in CBERS-4. For this reason, it deliveries few clouds segmented and make some mistakes. This fact may imply that a large part of images are being discarded in the processes of forest monitoring.

Cloud segmentation algorithms needs to account for the fact that there are some remote sensors with more spectral bands than others. The Landsat 8 and Sentinel-2 have 11 and 13 spectral bands in their images, respectively ((MARKHAM; STOREY; MORFITT, 2015) (LOUIS et al., 2016)).Differently, CBERS-4 WFI have just four spectral bands (Red, Blue, Green and near infra-red).

The neural network called U-net (RONNEBERGER; FISCHER; BROX, 2015), that was initially created to segment cell organelles is taking the state of art from CNN (Convolutional Neural Network) in segmenting tasks. There are some works in cloud segmentation field that have used this architecture and obtained satisfactory results (ZHENG; LIU; WANG, 2020)(FRANCIS; SIDIROPOULOS; MULLER, 2019)(JEPPESEN et al., 2019).

1.4 Work Structure

The rest of this work is organized as follows. Section 2 describes two different branches of the state-of-the-art semantic cloud segmentation and related work. Section 3 presents the entire methodology, including each step of the proposed workflow, justifying the way they were prepared. Section 4 shows the results of the presented tool applied to a cloud segmentation task. Finally, section 5 presents the conclusion and future work to complement the tool.

2 Related Work

With the advancement of machine learning, more and more tasks reliant on "human intelligence"are being replaced or aided by an intelligent computational model. On work (ALYAFEAI; GHOUTI, 2020) was proposed an automated workflow to classify images of cervical cancer. The proposed workflow was divided into four steps: acquire the images; pre-process the images; extract image features; and classify the images. The workflow presented in this article can be generically used for segmentation tasks in remote sensor images. However, using it in for remote sensing images requires to adapt it for dealing with large images (> 2 GB) since it has been originally developed for processing images from microscope.

The work we are conducting is in the interface of two fields of science: Artificial Intelligence and Geoinformatics. For this reason, in the reviewed literature some terms are written in different ways but they have the same meaning. Semantic segmentation is a well-known term used in the field of artificial intelligence. It's equivalent in the field of Geoinformatics is the term "pixel classification"(ZHU; WOODCOCK, 2012; FOGA et al., 2017; LOUIS et al., 2016). Thus, some works presented in this section may use one of these two terms.

Cloud segmentation could be divided into two big categories: physical - based algorithms and machine learning algorithms. The physical based algorithms use rules based on physical properties, like, for example, the relative angle of sensors to clouds and the angle from the sun, to extract a potential cloud layer and a potential cloud shadow layer (FOGA et al., 2017). This approach also has some disadvantages. First, there are several different remote sensors, with their own spectral bands and resolutions. As a consequence, their physical characteristics are inherent to each sensor and the algorithms can rarely be generalized to more than one sensor. Second, the physical algorithms execute the same calculus analysis for each image, therefore, there are associated computational costs, which turns the semantic segmentation task longer to execute.

The next paragraphs have the purpose of present the state-of-the-art papers on deep learning and physical based algorithms for cloud segmentation.

Starting with physical based algorithms, Fmask is one of the most used algorithms to segment clouds in Landsat TM images (ZHU; WOODCOCK, 2012). Currently, it is common to use it for benchmarking other algorithms. The evidence of this is that it was employed to evaluate another implementation of Fmask, called CFMask, comparing its performance with "Automated Cloud Cover Assessment"(ACCA) system (FOGA et al., 2017). Other physical based algorithm is the sen2cor, that can classify clouds using just physical data presented in each spectral band (LOUIS et al., 2016). This algorithm was created for Sentinel-2 Level-1C products. These Cmask, Fmask, and sen2cor algorithms evidence the relationship between physical based algorithms and remote sensor, in a way that the major part of algorithms difficultly will be adapt to different

sensors.

The success of deep learning applications in object recognition and semantic segmentation has excited scientists and increased the adherence of these techniques to problems with remote sensing images. Although the segmentation problem, or pixel classification, is an well studied problem, it remains very challenging. Considering the huge amount of remote sensor data and the complexity of the data, the machine learning methods are of great benefit to researchers and professionals actuating in Geosciences (YUAN; SHI; GU, 2021).

The machine learning area is wide, and comprises a series of different algorithms capable of solving various problems, like object detection, patterns on time series, natural language processing and more (JEPPESEN et al., 2019) (FRANCIS; SIDIROPOULOS; MULLER, 2019).

Machine learning models could be developed in a supervised, unsupervised, reinforcement, or semi-supervised manner. Fully Convolutional Neural Networks (FCNN) are supervised models (LONG; SHELHAMER; DARRELL, 2015) , that is, it is necessary to provide a ground-truth input to the algorithm in order to teach it the desired outputs. After training the model with the provided inputs, it should be capable of correctly classifying each new given input on the test step. With the results of those classifications, it would be possible to compute user defined metrics to determine how good the algorithm performs in the classification problem, allowing the comparison with the results of other works (LI et al., 2019).

The Fmask (ZHU; WOODCOCK, 2012) and MSCFF (LI et al., 2019) algorithms were used to remove thick clouds and cloud shadows from sensor images present in Sentinel-2 (ZHANG et al., 2020). The first step was making the cloud masks using the cited algorithms in multi-temporal imagery. Then, the imagery is combined to analyze the similarity to recover the cloud-free areas in one patched image. The biggest problem found in state-of-the-art cloud removal is the missing values on the reconstructed images, like spectral reflectance from differences between multi-temporal images, but the proposed algorithm rounds this problem. Usually the algorithms use images from only one remote sensor because each remote sensor has its own band with different spectral information. Nevertheless, the MSCFF architecture can handle both Landsat 8 and Sentinel-2 satellites multi-temporal/single imagery of small/large scenes. This shows that besides the difficulties some algorithms can be generic and can be adapt to different sensors.

In the literature, a neural network architecture was proposed that allows the input of remote sensing images of different dimensions, the model is called CloudFCN (FRANCIS; SIDIROPOULOS; MULLER, 2019). The mechanism of receive different input images is very useful to allow entering data from more than one remote sensor, thus making the model more generic. Using the Biome dataset (FOGA et al., 2017) the model showed an accuracy of 82.81% for RGB scenes and 91.00% with all spectral information versus CFmask (FOGA et al., 2017) with 65.69% using all spectral bands available. The lack of spectral bands, using only RGB bands, made difficult to the model to predict some cases of snow, sand (when it is used remote sensor of high resolution) and other objects that present high albedo (a metric that measures the quantity

of solar radiation reflected from the object versus the total incident energy). The mechanism that allows the entry of different dimensions is very interesting and expected in a workflow that aims to be generic and modular.

It is common to find architectures based on U-Net when the field of machine learning problem is semantic segmentation. When the goal of case study is cloud segmentation in remote sensor imagery, then U-Net architectures are also commonly used. A new approach to detecting clouds even with hardly distinguishable scenery was created (JEPPESEN et al., 2019). The model is a neural network built from U-Net and got meaningful results. The "Biome"and "Sparcs"datasets (FOGA et al., 2017) were used to evaluate the model divided into three situations. First, the model was trained with the Biome dataset and evaluated with Sparcs, and vice versa. Second, the groundtruth masks were generated by the Fmask algorithm. Finally, the model was trained and tested on the same dataset using image-based cross-validation, where the training data are the groundtruth cloud masks. The model improves the FMask even when using just RGB bands in all the three cited cases of evaluation. This case study shows that, even with images extracted from a remote sensor with small spectral bands, like CBERS-4 WFI, a deep learning model can produce trustworthy cloud masks.

Currently, the algorithms for semantic segmentation on remote sensing images mainly use deep learning strategies (YUAN; SHI; GU, 2021). This occurs because in the last decade, more and more deep learning algorithms have demonstrated much superior performance in tasks as object identification and semantic segmentation.The major part of the deep learning models presented in this section were based on U-Net, therefore, it seems to be the direction that the state-of-the-art is going.

3 Dataset

Building a dataset for cloud semantic segmentation is a challenge. When groundtruth masks are used to segment a cat, for example, it will be observed whether each pixel of image belongs or not to the cat. Clouds are different, the center of a cloud can be dense, with high albedo, and borders of cloud can have transparent gradient that makes it difficult to classify whether that pixel is a cloud or not (DOWLING; RADKE, 1990).

Just to facilitate the work, in this proof of concept,we used the Cmask algorithm to automatically produce a dataset of segmented images. This strategy allow us to focus on the deep learning algorithm development. Later, a larger dataset will be segmented manually. The current dataset is composed by 4 images from WFI remote sensor, that were divided into smaller images, as shown in the figure 3.1, of 256x256x4 pixels, totalizing more than 10.760 images with its respective masks. As masks were obtained by applying the Cmask algorithm (QIU; ZHU; WOODCOCK, 2020) to each image, these masks have errors associated to them and do not correctly represent the ground truth.

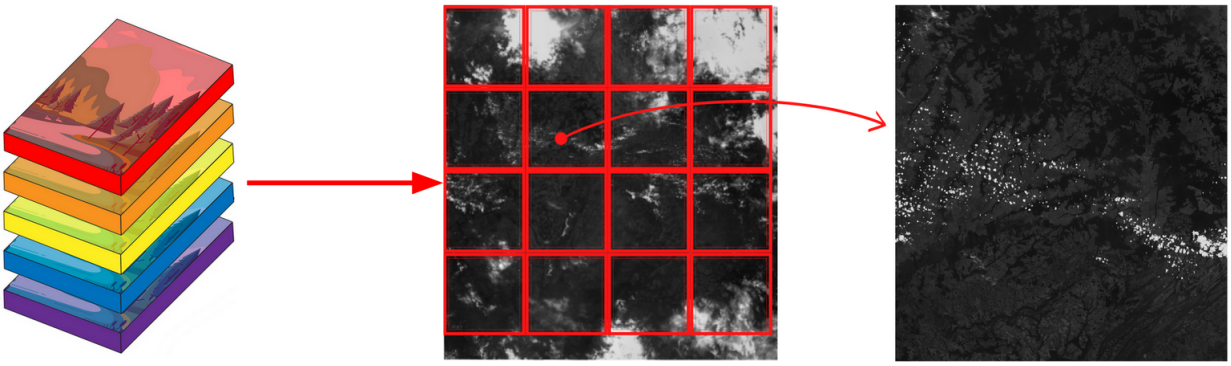

Figure 3.1 – Demonstration of the image cropping step. On the left the raw scene, in the center the raw scene with the subscenes in red, on the right the resulting subscene.

Each image from CBERS-4 WFI remote sensor has four spectral bands, Red, Blue, Green and near infra-red. The red, blue and green (RGB) bands, in cloud semantic segmenting task, are responsible to detect the albedo level, and the infrared is known by being absorbed by vegetation (ZHANG; XIAO, 2014).

4 Methodology

This chapter details each step present in the developed workflow, Figure 4.1. The workflow was divided into two main components: model calibration and production model. Workflow steps are decoupled from each other. These steps are called in two Jupyter notebook files. Each notebook implements the main control flow of each component, one for model calibration and another for the production model. The notebook files act as an interface for the user, in which he/she may script the workflow in Python programming language (VANROSSUM, 1995), configuring and connecting steps. Each section of this chapter explains only one workflow step.

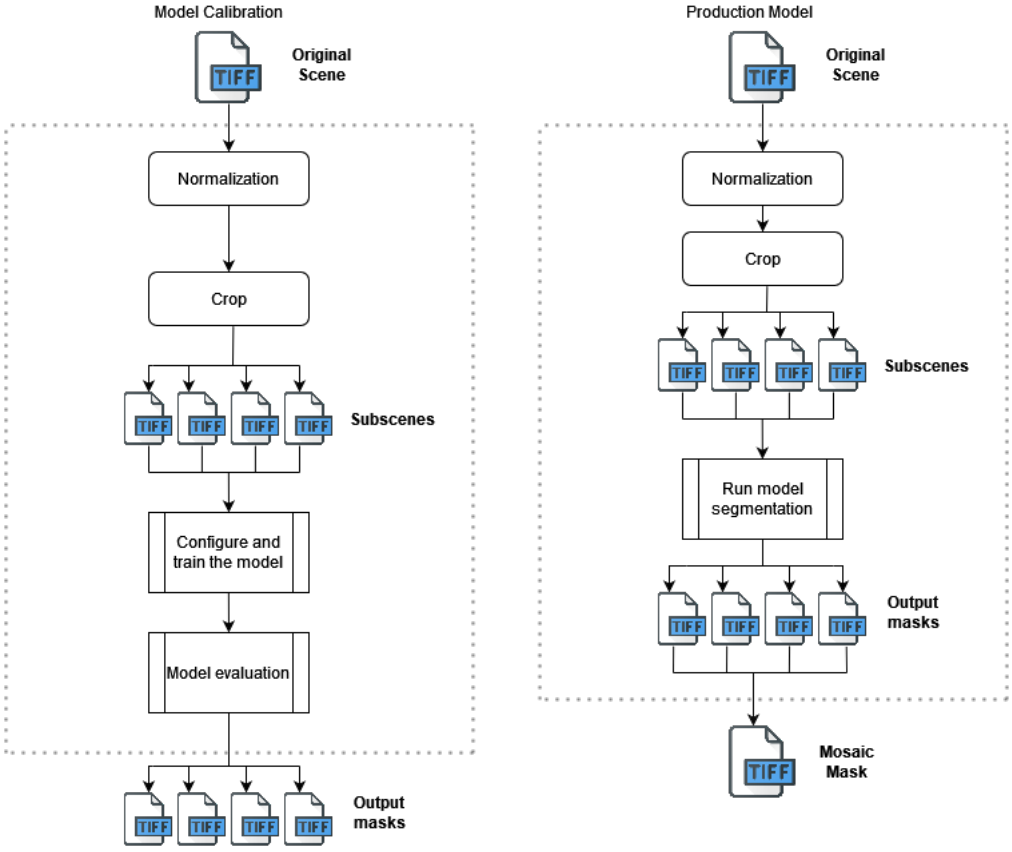

Figure 4.1 – Developed workflow

Each step is implemented as a Python function based on the Keras (KETKAR, 2017) library that provides features to deep learning model development. The GDAL (WARMERDAM, 2008) and NumPy (OLIPHANT, 2006) libraries have been combined to handle remote sensing images without losing the coordinates and "no data values"metadata.

Images from remote sensor have some pixel values called "no data". The "no data value"pixels are a mechanism to fill the matrix representing the images in places where no data have been collected by the remote sensor. When the image is collected by the remote sensor, the position and angle of the remote sensor vary in relation to earth, thus the image seems to

be rotated and the edge of images is filled with empty pixels. This mechanism is crucial, for example, to overlay an image over a map in the correct position without hiding areas where no data have been collected.

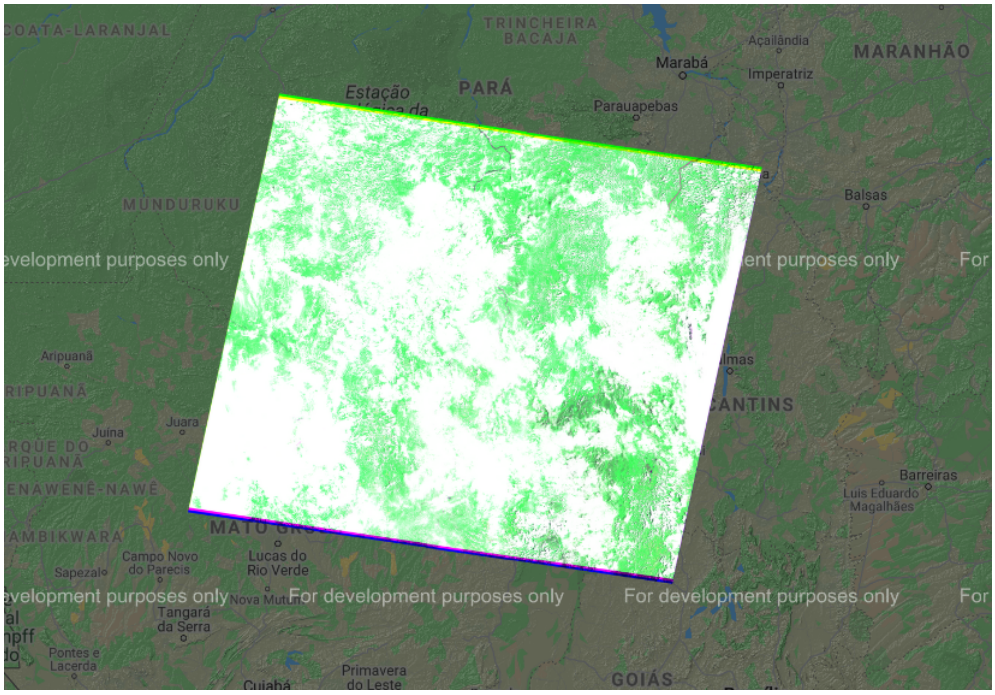

Figure 4.2 – Screenshot of the image generated by a remote sensor overlapped on a map on the INPE website

In Figure 4.2, it can be seen a remote sensor image superimposed on the map. The image rotation results from the satellite position and angle and the image displacement reveals pixels of "no data"value.

4.1 Model Calibration

The model calibration workflow component consists of several steps: pre-processing remote sensor imagery (normalization and crop), configure and train a deep learning model, and compute metrics to evaluate the trained model.

4.1.1 Handling Dataset

The first step in handling the dataset is to build a directory hierarchy. The directory folders are divided into the following: original scene folder, to be filled with the original scenes (with all bands); normalized scene folder, to be filled with the respective normalized scenes; cropped production folder, to be filled with subscenes from the normalized scenes; cropped production mask folder, to be filled with output masks estimated by the model for each subscene.

In sequence, each scene goes through a Min-Max normalization step, which uses a linear operation to transform the pixel's values (SARANYA; MANIKANDAN, 2013). Here, the original

pixels values ranging from 0 to 10.000 are mapped to new values between 0 and 1. The purpose of the normalization step is to transform the image in a way that each image presents a similar distribution of its pixels, and this helps in the conversion of the neural network.

In the next step, the normalized scene will be cropped based on subscene dimensions parameter and separated between the training, validation and test groups. Before starting the training step, the proportion of subscenes in each group must be included. The groups need to be randomly formed to avoid biased data (GÉRON, 2019). Usually the proportion of groups is 70% for the training group, 15% for validation and 15% for the test, but these values may change depending on the dataset size.

4.1.2 Training Step

In the learning stage, the model must receive subscenes as input. The set of subscenes is called batches. To load the batches and not use all the machine's memory, a mechanism called "generator"was built to iterate among the images, loading a batch and, after training, reallocating memory to a new batch. In this way, the generator expects the batch size hyperparameter as the input.

4.1.2.1 Deep Learning Model

Although the workflow is modular and can coupled to different deep learning models, only one neural network model was used in this work to implement this step. The chosen neural network has an U-net architecture (RONNEBERGER; FISCHER; BROX, 2015) due to its performance and presence in the state-of-the-art. Semantic segmentation is a particular field of computer vision, and the most employed deep learning technique on state-of-the-art is precisely the FCNN, which were firstly developed to the Biomedical Science field, with the U-net architecture(RONNEBERGER; FISCHER; BROX, 2015). This neural network is called U-net because its architecture has the shape of the character "U"as can be seen in the Figure 4.3. An U-net can extract characteristics from an image and transform these characteristics in an one-dimension vector. Its architecture is built in two parts. The former is called encoder, and the latter decoder. The encoder step consists of applying convolutional operations followed by a maxpool downsampling to encode the input image into feature representations at multiple different levels. The decoder step consists of reconstructing the image through upsampling and concatenation at each layer level using the features learned in the encoder step. The enconder-decoder strategy is what makes U-nets to perform well in the segmentation task, extracting and mapping characteristics from an image.

As the tensor representing the subscene has several dimensions depending on the available memory and the number of bands of the remote sensor, in this case each tensor is a image with (256 pixels x 256 pixels x 4 bands), we have chosen a U-net implementation from GitHub with configurable input dimensions (ZAK, 2021), so that, the resulting workflow can also be configured.

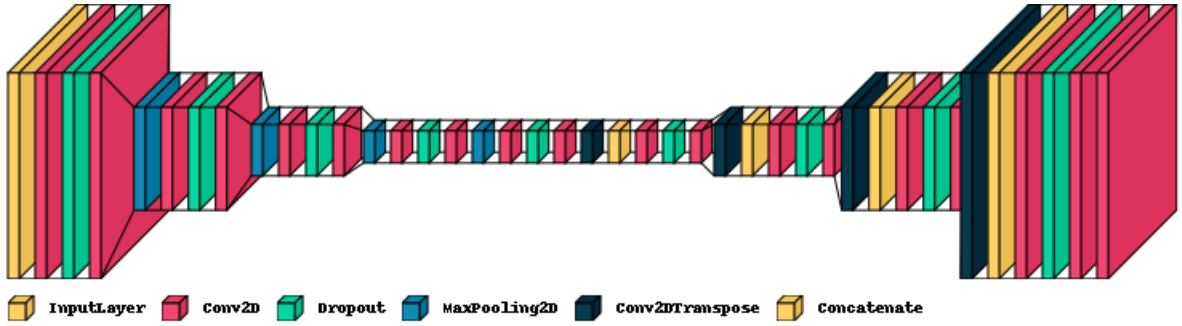

Figure 4.3 – U-net Architecture: The fist half of its architecture is the encoder, the second half is the decoder.

4.1.2.2 Metrics

After configure the U-net dimensions, we defined other hyperparameters for configuring the neural network and implemented some metrics to evaluate its performance. This section explains these details.

The Keras library contains many metrics available to interpret the confidence of a neural network output, except the F1-Score metric. This section will explain the most important ones, but the workflow allows the user to defined its own metric.

Starting by some well-known metric, the accuracy, with "TP"be true positives, "TN"be true negatives, "FP"be false positives and "FN"be false negatives, its equations are presented as follows:

$$Accuracy = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{4.1}$$

Accuracy is the proximity of a result to its actual reference value. Thus, the higher the level of accuracy, the closer to the reference or real value is the result found. The accuracy considers only the correctly predicted values without distinguishing whether false negatives or false positives are worse for the case study.

In addition of available metrics on Keras library, it was implemented one more metric that can be used in the project, the F1-Score. F1-Score is composed using other two well-known metrics: precision and recall.

The Precision can be written as follows:

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

When the result presents several false positives, or few true positives, this increases the denominator and makes the precision small. The Precision has a heavier weight to predicted false positives, this way, when this metric is used, the goal should be to improve the correct predictions by minimizing the false positive cases and increasing the cases of true positives.

The Recall can be wrote as follow:

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

The recall measures the model's ability to detect true positives. The higher the recall, the more positive samples are detected. The recall metric, unlike Precision, has a heavier weight to false negative predictions, that is, when this metric is used, the goal should be to improve the predictions by avoiding false negative cases.

Finally, the F1-Score consists of the harmonic mean of Precision and Recall, as in the following equation:

$$F_1 = 2 * \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.4}$$

That is, differently from accuracy, the F1-Score being a harmonic mean gives much more weight to low values. Thus, this metric is high just if both recall and precision are high. Therefore, it is a stable metric to compute the confidence level of a deep learning model.

Finally, it is expected that the user defines a loss function in order to finish the neural network model configuration. A loss function is used to optimize the parameter values in a neural network model. Loss functions map a set of parameter values for the network onto a scalar value that indicates how well those parameters accomplish the task that the network is intended to do. There are several loss functions already implemented in Keras library and all of them are allowed to be used as input at this step.

4.1.3 Test Step

At the test step, all the subscenes on test folder will be loaded and used as input to the trained model. These images have never been seen by the model. It is important to avoid biased metrics and to see how the model handles new inputs. After the model consumes the input images and output their respective estimated cloud masks, the groundtruth masks will be compared to them. The metrics configured in the previous step are then computed to each pair estimated-groundtruth masks and are stored in a table. Finally, the general average of each metric is calculate from this table.

4.2 Production Model

The production model workflow component should have a pre-trained model. This component is responsible for cutting a new scene into small subscenes, using them as inputs to a trained model, and for reconstructing the output mask with the same size of the entire original scene.

4.2.1 Handling Data

The first step in handling the dataset is to build a directory hierarchy. The directory folders are divided into the following: production scene folder, to be filled with the original scenes; normalized scene folder, to be filled with the respective normalized scenes; cropped production folder, to be filled with subscenes from the normalized scenes; cropped production mask folder, to be filled with output masks estimated by the model for each subscene; segmented mask folder, to be filled with the entire reconstructed mask. If there several scenes in the input dataset, then the production model will create a entire directory hierarchy for each scene.

4.2.2 Segmentation Step

The next step after building the directory hierarchy is to normalize and to crop the scenes, a process that is identical to the step "Handling Dataset"in the Model Calibration component.

In the next step, the subscenes are inputted into the pre-trained neural network and the outputs are collected to reconstruct the entire segmented mask.

The process of reconstructing the segmented mask is called "mosaicing". A mosaic is a combination or merge of two or more images, the function uses the coordinates metadata present on a tif file to correctly locate each subscene in the geographical space. The process was implemented using the "WARP"function present on GDAL library.

5 Results

This chapter mentions the materials used to assess the workflow. Then, the parameters chosen for workflow calibration are shown and justified. Finally, the performance metrics resulting from the workflow execution are shown and discussed.

5.1 Materials

The workflow was entirely develop in the Python language and it was executed on a 2.60GHz Intel Core i7-9750H machine, with 8 gigabytes of RAM in the Windows 10 operating system. The main libraries used were: Keras, for model implementation; GDAL, to manipulate the georeferenced images; and NumPy, to handle tensors.

The workflow was developed to provide loosely coupled steps, allowing users to customize or replace each of them. In this way, users can customize the workflow to deal with datasets from different remote sensors. Users can also replace workflows steps by overwriting the functions that implement them.

In the case study conducted to evaluate the workflow, a dataset composed by images from CBERS-4 WFI remote sensor was used. These images have four spectral bands, Red, Blue, Green and near infra-red. The red, blue and green (RGB) bands, in cloud semantic segmenting task, are responsible for detecting albedo level, and infrared region is known to be absorbed by vegetation (ZHANG; XIAO, 2014).

Each scene presented in the training step, must have a groundtruth mask. The masks used in the case study were extracted by Cmask algorithm (FOGA et al., 2017).

5.2 Hyperparameters

The workflow must be able to run on a personal machine with little computational power, the "subscene dimension"hyperparameter is given fixed height and width values and the number of spectral bands so that the original scene is cut into smaller scenes that can be loaded into a small memory space. In the case study, each subscene was defined to have a size of 256x256x4 pixels, being 256 pixels in width and height and 4 layers for red, green, blue, and near infrared spectral bands. The generator mechanism implemented in this work allowed the pipeline to work using several images up to 2 gigabytes in size, even with a machine with only 8 gigabytes of RAM.

The next hyperparameters to be chosen are the loss function, the optimizer and the performance metrics. To choose the loss parameter the scope of problems must be analyzed.

Here, each existent pixel should be classified into two classes: "cloud"or "background."This is a classic case of binary classification, where one image can have its pixels belong or not to a class. To this scope of problem, the binary cross-entropy, which is based on Bernoulli's formula, is suitable (RUBY; YENDAPALLI, 2020).

$$loss = 1/n \cdot \sum_{y}^{i=1} yi \cdot log(\hat{yi}) + (1 - yi) \cdot log(1 - \hat{yi}) \tag{5.1}$$

The optimizer parameter is a responsible function for minimizing the loss (GÉRON, 2019). Adam optimizer is one of the most popular and famous gradient descent optimization algorithms. It is a method that computes adaptive learning rates for each parameter. It stores both the decaying average of the past gradients, similar to momentum and the decaying average of the past squared gradients, similar to RMS-Prop and Adadelta. Thus, it combines the advantages of both methods (KINGMA; BA, 2014).

Finally, as the case study has no weight distinction for false positives and false negatives, we choose the F1-score metric to monitor the model learning.

As a next step, it is necessary to configure one more hyperparameter for the early stop function. A mechanism was built to interrupt the model training early. The function responsible for this mechanism works as follows:

First, a hyperparameter called "patience"is entered. This hyperparameter is used as a threshold for the maximum number of iterations the model can perform without improving the learning metric. If the number of interactions without improvements surpasses the patience value, the model will stop the training and their weights will be saved on the basis of the interaction that performed better. This mechanism helps to decrease the time spent with models that do not show improvement and it also helps save the model in its best iteration. To the study case the patience was defined as a threshold of five interactions.

Finishing the model configuration step, there is a last hyperparameter that is also responsible for being a threshold of interactions, but this threshold, called "epochs,"represents the maximum number of interactions. The training step will never exceed the number of epochs. For the case study, the epochs were defined as a value of twenty-five interactions. The complete setting data can be seen at the Table 5.1.

5.2.1 Performance metrics

The performance metrics presented in this section were extracted from the workflow execution using the materials and hyperparameters mentioned above.

The first information that will be generated is the history graph represented on the Figure 5.1. It shows the metrics in each interaction during the training step. In this way, it is possible to see the F1-score and loss in each epoch. It is possible to observe that, during the training,

Hyperparameter	KNN
Dimension	256
Bands	4
Loss Function	Binary cross-entrophy
Optmizer	Adam
Metric	F1-score
Patience	5
Epochs	25

Table 5.1 – List of hyperparameters

the model improves at each interaction (Training F1-score) and that it shows an asymptotic behavior after fifteen interactions. However, the model performance is not always confirmed by the validation metrics (Validation F1-Score).

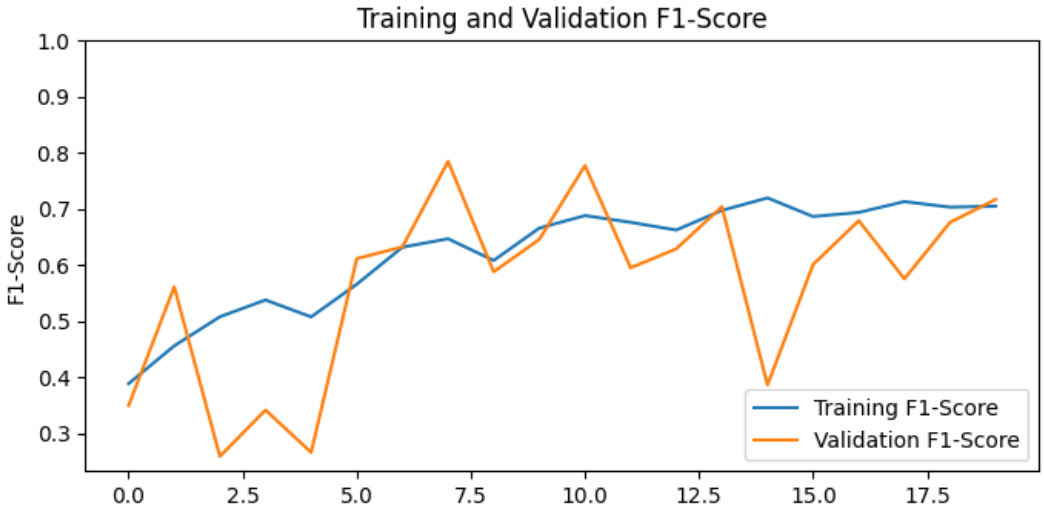

Figure 5.1 – Evolution of model along the training step

Bellow, Figure 5.2 shows some masks from the test subscenes. The input subscenes appear in the left column, the input masks in middle column, and the predicted mask in the right column. Each pixel of predicted masks has color that indicates the confidence level of the model in the prediction. In the input images, the density clouds range from black to white. Light grays register more presence of cloud. In the input mask, yellow means cloud and purple means no-cloud.

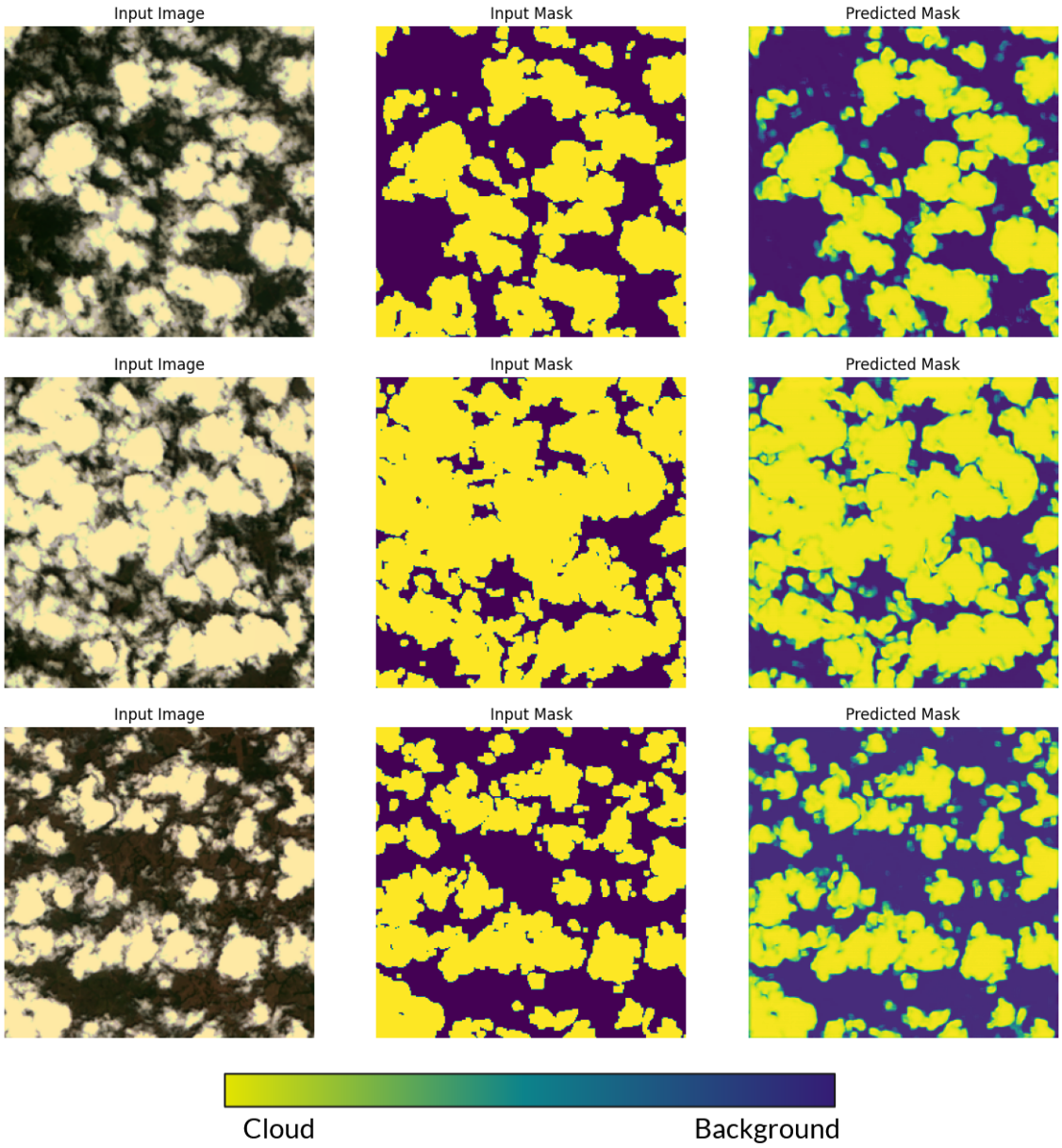

Figure 5.2 – Some input data with the respective predicted mask for the U-net model

As can be seen, the resulting masks present a color gradient between the background color (purple) and the cloud color (yellow), according to confidence level of the model in the classification. To make a binary mask, another hyperparameter must be defined, a confidence-level threshold. If the threshold is set to, for example, 0.97, then only pixels for which the model is 97% confident that they are cloud will be colored in yellow. Everything else will be colored in purple. Therefore, it is important to find a threshold value that best avoids wrong coloring. We avoid to arbitrarily choose a threshold value in our case study, preserving the level of confidence analysis.

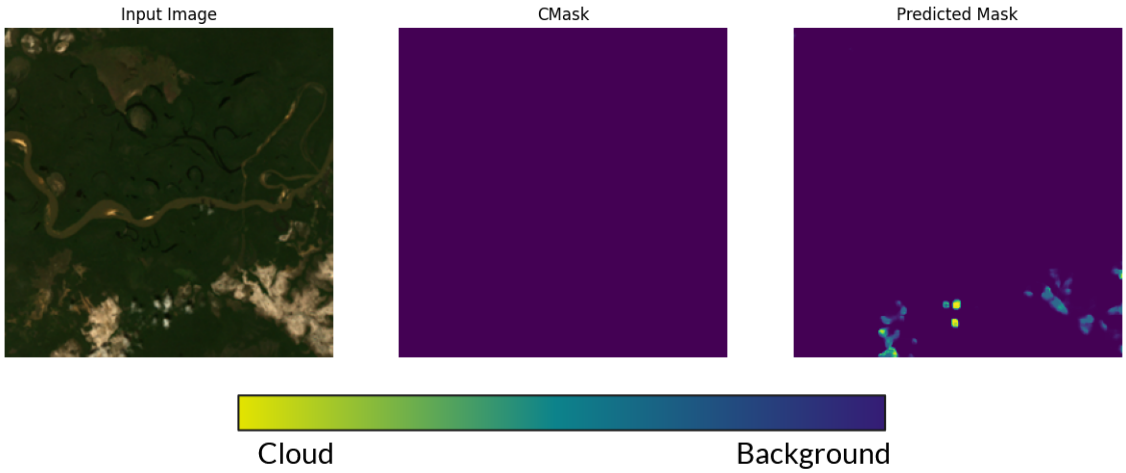

Figure 5.3 – Some input data with the respective predicted mask that present a case of false positive cloudy pixels.

As can be seen in the Figure 5.3, exposed soil has high reflectance level (albedo) in comparison with vegetation, then the algorithm can get confused whether this object is a cloud or not.

The next step is to assess how the workflow works in a production scenario. After entering the scene in the workflow, the entire scene will be segmented and its output mask will be reconstructed and saved in the "Merged Files"directory.

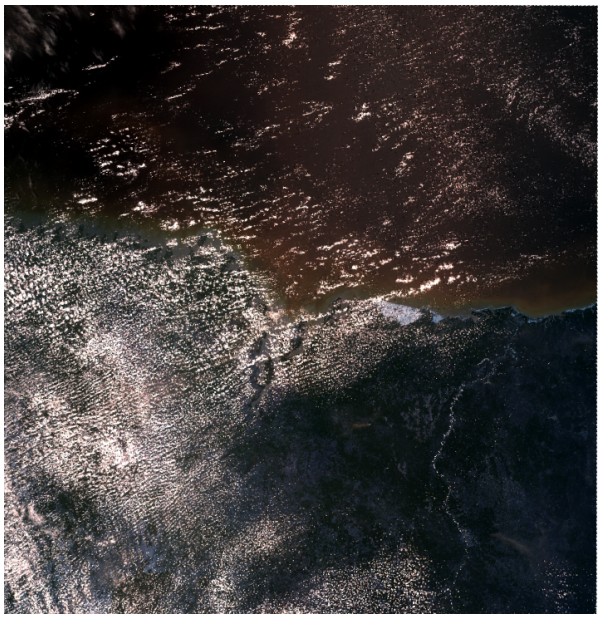

Figure 5.4 – Raw Scene.

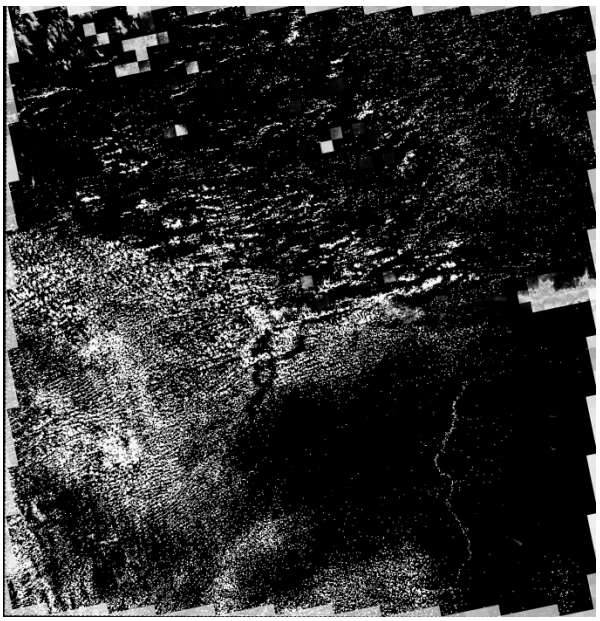

Figure 5.5 – Mosaic mask.

Figure 5.5 shows a raw image from CBERS-4A dataset on the left, which the deep learning model has never seen. On the right side, it shows the mosaic of the output-segmented masks. It can be seen that the mosaic image does not have a defined threshold for pixel confidence levels. In this way, the pixel values are a gradient of values between 0 and 1. By applying a threshold, it

is possible to select only pixels where the model has been segmented with high confidence. In the Figure 5.6 no threshold was set, and the Figure 5.7 a 97% confidence limit was used.

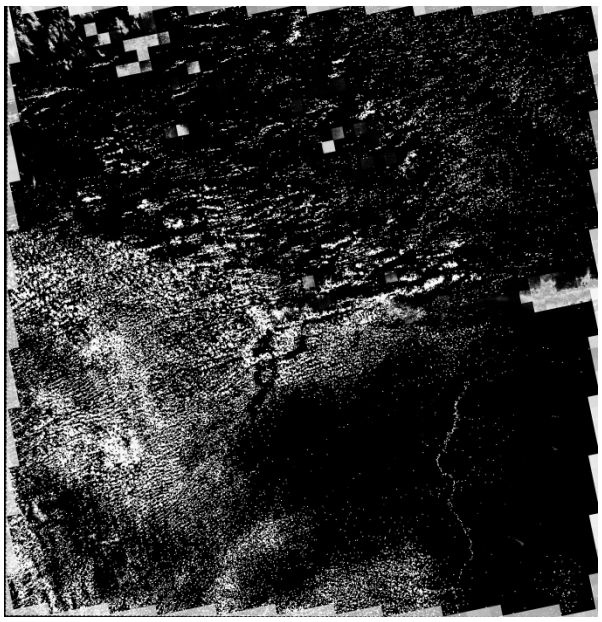

Figure 5.6 – Mosaic mask.

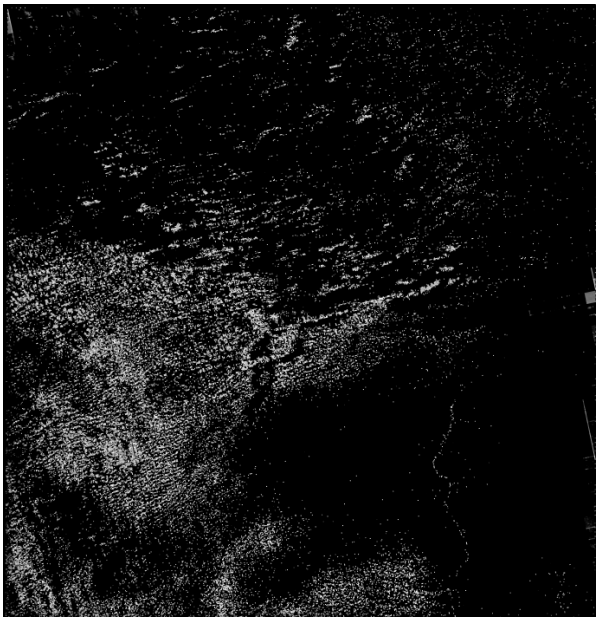

Figure 5.7 – Mosaic mask with threshold.

These two last Figures show the importance of defining a threshold for classifying pixels. This process is called threshold tuning and it requires to balance the recall-precision trade-off (BUCKLAND; GEY, 1994). There are strategies to define a threshold in segmentation masks, but they are not addressed in this study.

6 Final Considerations

This work developed and evaluated a workflow for semantic segmentation of remote sensor images. The proposed workflow handles the dataset automatically, requiring only hyperparameter inputs. The workflow is distributed as a free and open source software, under the LGPL GNU license. Its source code can be found in the repository: https://gitlab.com/ufopterralab/projeto-segmentacao-semanticas-cbers-4a-wfi.

The workflow can be configured for a more powerful machine (increasing sub-scene size and batch size) or running on a personal computer, with less computing power. Even with only 8 gigabytes of RAM the workflow was able of training a deep learning model with several images of more than 2 gigabytes each one. The choice of a threshold was not included in the scope of this project, but was left open for future work.

The initial idea of building this workflow was to improve semantic segmentation for the CBERS-4A remote sensor, but in the process of building the dataset and workflow, it became clear that the workflow itself could be a project that would help scientists working with semantic segmentation of remote sensor images, requiring only the hyperparameterization of the model to obtain an initial result.

In the process of building the case study dataset, was identified a scientific and technological gap in cloud removal algorithms for the CBERS-4A WFI remote sensor. Today the INPE works with the Cmask algorithm and it does not perform well for CBERS-4A data. Additionally, a groundtruth dataset is missing for this remote sensor, limiting the attempts to implement and evaluate other machine learning models to segment could in CBERS-4A WFI imagery. This way, there are possible continuations of this work, as they are listed below:

- Develop a CBERS-4A WFI dataset with groundtruth masks.
- Develop a CBERS-4A WFI dataset with one more class, e. g., including "cloud shadows".
- Develop a function to select a confidence threshold to apply to the mosaic output mask.
- Develop a mechanism to avoid the loss of pixels in the step of cropping the sub scenes.
- Develop a mechanism to ignore pixels that have "no-data"values.

References

ALYAFEAI, Z.; GHOUTI, L. A fully-automated deep learning pipeline for cervical cancer classification. *Expert Systems with Applications*, Elsevier, v. 141, p. 112951, 2020.

BUCKLAND, M.; GEY, F. The relationship between recall and precision. *Journal of the American society for information science*, Wiley Online Library, v. 45, n. 1, p. 12–19, 1994.

CHAUDRON, M.; LARSSON, S.; CRNKOVIC, I. Component-based development process and component lifecycle. *Journal of Computing and Information Technology*, Fakultet elektrotehnike i računarstva Sveučilišta u Zagrebu, v. 13, n. 4, p. 321–327, 2005.

CHEN, H.; CHEN, A.; XU, L.; XIE, H.; QIAO, H.; LIN, Q.; CAI, K. A deep learning cnn architecture applied in smart near-infrared analysis of water pollution for agricultural irrigation resources. *Agricultural Water Management*, Elsevier, v. 240, p. 106303, 2020.

DOWLING, D. R.; RADKE, L. F. A summary of the physical properties of cirrus clouds. *Journal of Applied Meteorology and Climatology*, v. 29, n. 9, p. 970–978, 1990.

EPIPHANIO, J. C. N. Cbers-3/4: características e potencialidades. In: *Proceedings of the Brazilian Remote Sensing Symposium, Curitiba, Brazil*. [S.l.: s.n.], 2011. v. 30, p. 90099016.

FOGA, S.; SCARAMUZZA, P. L.; GUO, S.; ZHU, Z.; JR, R. D. D.; BECKMANN, T.; SCHMIDT, G. L.; DWYER, J. L.; HUGHES, M. J.; LAUE, B. Cloud detection algorithm comparison and validation for operational landsat data products. *Remote sensing of environment*, Elsevier, v. 194, p. 379–390, 2017.

FRANCIS, A.; SIDIROPOULOS, P.; MULLER, J.-P. Cloudfcn: Accurate and robust cloud detection for satellite imagery with deep learning. *Remote Sensing*, Multidisciplinary Digital Publishing Institute, v. 11, n. 19, p. 2312, 2019.

GÉRON, A. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. [S.l.]: O'Reilly Media, 2019.

INPE. *Satélite sino-brasileiro CBERS-4 completa seis anos em órbita*. 2020. Disponível em: <http://www.inpe.br/noticias/noticia.php?Cod_Noticia=5624>.

JEPPESEN, J. H.; JACOBSEN, R. H.; INCEOGLU, F.; TOFTEGAARD, T. S. A cloud detection algorithm for satellite imagery based on deep learning. *Remote sensing of environment*, Elsevier, v. 229, p. 247–259, 2019.

KETKAR, N. Introduction to keras. In: *Deep learning with Python*. [S.l.]: Springer, 2017. p. 97–111.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

LI, Z.; SHEN, H.; CHENG, Q.; LIU, Y.; YOU, S.; HE, Z. Deep learning based cloud detection for medium and high resolution remote sensing images of different sensors. *ISPRS Journal of Photogrammetry and Remote Sensing*, Elsevier, v. 150, p. 197–212, 2019.

LONG, J.; SHELHAMER, E.; DARRELL, T. Fully convolutional networks for semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2015. p. 3431–3440.

LOUIS, J.; DEBAECKER, V.; PFLUG, B.; MAIN-KNORN, M.; BIENIARZ, J.; MUELLER-WILM, U.; CADAU, E.; GASCON, F. Sentinel-2 sen2cor: L2a processor for users. In: SPACEBOOKS ONLINE. *Proceedings Living Planet Symposium 2016*. [S.l.], 2016. p. 1–8.

MARKHAM, B.; STOREY, J.; MORFITT, R. *Landsat-8 sensor characterization and calibration*. [S.l.]: Multidisciplinary Digital Publishing Institute, 2015.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943.

MORAN, M. S.; HYMER, D. C.; QI, J.; KERR, Y. Comparison of ers-2 sar and landsat tm imagery for monitoring agricultural crop and soil conditions. *Remote Sensing of Environment*, Elsevier, v. 79, n. 2-3, p. 243–252, 2002.

OLIPHANT, T. E. *A guide to NumPy*. [S.l.]: Trelgol Publishing USA, 2006. v. 1.

POUYANFAR, S.; SADIQ, S.; YAN, Y.; TIAN, H.; TAO, Y.; REYES, M. P.; SHYU, M.-L.; CHEN, S.-C.; IYENGAR, S. S. A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, ACM New York, NY, USA, v. 51, n. 5, p. 1–36, 2018.

QIU, S.; ZHU, Z.; WOODCOCK, C. E. Cirrus clouds that adversely affect landsat 8 images: what are they and how to detect them? *Remote Sensing of Environment*, Elsevier, v. 246, p. 111884, 2020.

RONNEBERGER, O.; FISCHER, P.; BROX, T. U-net: Convolutional networks for biomedical image segmentation. In: SPRINGER. *International Conference on Medical image computing and computer-assisted intervention*. [S.l.], 2015. p. 234–241.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, American Psychological Association, v. 65, n. 6, p. 386, 1958.

ROSSOW, W. B.; SCHIFFER, R. A. Advances in understanding clouds from isccp. *Bulletin of the American Meteorological Society*, American Meteorological Society, v. 80, n. 11, p. 2261–2288, 1999.

RUBY, U.; YENDAPALLI, V. Binary cross entropy with deep learning technique for image classification. *International Journal of Advanced Trends in Computer Science and Engineering*, v. 9, n. 10, 2020.

SARANYA, C.; MANIKANDAN, G. A study on normalization techniques for privacy preserving data mining. *International Journal of Engineering and Technology (IJET)*, Citeseer, v. 5, n. 3, p. 2701–2704, 2013.

SOUZA, A.; MONTEIRO, A. M. V.; RENNÓ, C. D.; ALMEIDA, C. A.; VALERIANO, D. d. M.; MORELLI, F.; VINHAS, L.; MAURANO, L. E. P.; ADAMI, M.; ESCADA, M. I. S. et al. Metodologia utilizada nos projetos prodes e deter. *INPE: São José dos Campos, Brazil*, 2019.

VANROSSUM, G. Python reference manual. *Department of Computer Science [CS]*, CWI, n. R 9525, 1995.

WARMERDAM, F. The geospatial data abstraction library. In: *Open source approaches in spatial data handling*. [S.l.]: Springer, 2008. p. 87–104.

XU, A.; LIU, Z.; GUO, Y.; SINHA, V.; AKKIRAJU, R. A new chatbot for customer service on social media. In: *Proceedings of the 2017 CHI conference on human factors in computing systems*. [S.l.: s.n.], 2017. p. 3506–3510.

YUAN, X.; SHI, J.; GU, L. A review of deep learning methods for semantic segmentation of remote sensing imagery. *Expert Systems with Applications*, Elsevier, v. 169, p. 114417, 2021.

ZAK, K. *keras-unet*. [S.l.]: GitHub, 2021. <https://github.com/karolzak/keras-unet>.

ZHANG, Q.; XIAO, C. Cloud detection of rgb color aerial photographs by progressive refinement scheme. *IEEE Transactions on Geoscience and Remote Sensing*, IEEE, v. 52, n. 11, p. 7264–7275, 2014.

ZHANG, Q.; YUAN, Q.; LI, J.; LI, Z.; SHEN, H.; ZHANG, L. Thick cloud and cloud shadow removal in multitemporal imagery using progressively spatio-temporal patch group deep learning. *ISPRS Journal of Photogrammetry and Remote Sensing*, Elsevier, v. 162, p. 148–160, 2020.

ZHANG, Y.; ROSSOW, W. B.; LACIS, A. A.; OINAS, V.; MISHCHENKO, M. I. Calculation of radiative fluxes from the surface to top of atmosphere based on isccp and other global data sets: Refinements of the radiative transfer model and the input data. *Journal of Geophysical Research: Atmospheres*, Wiley Online Library, v. 109, n. D19, 2004.

ZHENG, J.; LIU, X.-Y.; WANG, X. Single image cloud removal using u-net and generative adversarial networks. *IEEE Transactions on Geoscience and Remote Sensing*, IEEE, 2020.

ZHU, Z.; WOODCOCK, C. E. Object-based cloud and cloud shadow detection in landsat imagery. *Remote sensing of environment*, Elsevier, v. 118, p. 83–94, 2012.