POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

On-board cloud screening algorithms for satellite imaging

Supervisors

Prof. Enrico MAGLI

Prof. Diego VALSESIA

Candidate

Francesco SORRENTINO

October 2024

Abstract

Remote sensing data acquired through multi-spectral satellite sensors provide an opportunity to monitor and understand the Earth's physical, chemical, and biological systems. However, approximately 70% of the sky is often covered by clouds, which can lead to the processing and storage of images that are not useful for Earth observation tasks, such as monitoring vegetation, ice, or water bodies.

To better understand which images should be processed and how, cloud detection algorithms have been designed to identify cloudy pixels, generating cloud segmentation masks that can be used to compute cloud coverage. Accurate detection is essential to filter out highly cloud-covered images for reliable downstream processing. The variability in cloud characteristics makes this identification process complex, as clouds can vary in shape, density, and other spectral properties, typically resembling bright surfaces like snow, ice, deserts, or rocks in certain spectral bands.

Moreover, on-board cloud detection is a real-time application that requires algorithms to identify cloud-covered data, either to discard or to exploit cloudy pixels to efficiently compress data for downstream tasks. However, the primary challenges in on-board systems stem from the limited computational resources available on satellites, including power, memory, and processing capabilities. Balancing the trade-offs between accuracy, false positive rates, and efficiency is the main challenge in this context.

This thesis explores multiple approaches to cloud detection, including a threshold-based cloud detection algorithm, machine learning algorithms, and a deep learning model. The first approach is a physical cloud detection algorithm used in Sen2Cor, which involves a threshold-based test on image bands that, in this thesis, was optimized to minimize false positive rates (FPR). Support vector machines and random forests were also trained for cloud segmentation, despite being computationally heavy, to provide a better benchmark for comparison. Finally, a lightweight model for cloud segmentation is proposed, designed to be competitive with other deep learning models in terms of accuracy, while remaining efficient and comparable to threshold-based algorithms. The results obtained using these methods are presented and compared at the end of the thesis.

Acknowledgements

I would like to thank Prof. Enrico Magli and Prof. Diego Valsesia who gave me this opportunity, not having me as their student, and for the patience, support and feedback offered for the work performed in these months.

Secondly, i would like to thank my parents, my brother and all my family that supported me and cared of me throughout these years, allowing me to pursue my Master's Degree at Politecnico di Torino, far from my home, without making me feeling pressured by this adventure.

Last but not least, i would like to thank my second family, the *Kyssene* group, that started during my Bachelor's as a study group, which over time it strengthened, becoming like brothers, that made me grew up as a person and gave me the motivation to embark in this trip.

"A mio padre, a mio fratello, ma soprattutto a mia madre che per 22 anni mi ha supportato in ogni mia decisione, che mi ha cresciuto e voluto bene più di ogni altra persona che potrò mai incontrare nella mia vita e che da quattro anni ne custodisco il suo amore."

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Everyday large volumes of data are acquired through optical sensor from earth observation satellites, enabling researchers to grasp environmental changes on the planet, such as climate changes, natural disasters, land use, vegetation and water courses. However these images are often covered partially or totally by clouds: in fact clouds cover around 70% of the earth surfaces, leading to contaminated image and reducing the availability of clear sky data. Furthermore images are transmitted from the satellite to the ground station, this downstream operation can be useless if the image is totally covered by clouds.

Cloudy pixels can be compressed at an higher magnitude than clear sky pixel, optimizing the downstream of the images while maintaining the quality of the area we are interested in. Since multi-spectral images consists in multiple bands where each one is captured a different wavelengths, traditionally cloud detection algorithms (physical algorithms) leverage the spectral properties of clouds, utilizing predefined thresholds based on physical principles. While effective, physical methods can struggle in complex atmospheric conditions or over heterogeneous surfaces, such as snow or urban areas.

In the recent years, numerous works have been tackled cloud detection exploiting machine learning and deep learning techniques, training models on labeled satellite data, obtaining higher accuracy and generalization. The main problem with these algorithms is the high demand of resources, from huge amount of data for the training process, to specific hardware accelerator to run the models. Indeed physical algorithms are way faster then classic Semantic Segmentation models, even if the latter are more accurate, exploiting spacial features through convolutional neural networks.

The design of a new cloud detection algorithms depends on different variables, for example minimizing the false positive rate to avoid compressing useful pixels, minimizing latency to increase the throughput and the easiness of implementation.

Starting from and existing cloud detection algorithm implemented in Sen2Cor,

created by the European Space Agency, this thesis aim to the optimization and design of a new cloud detection model, evaluating three distinct approaches: physical algorithm, machine learning-based algorithms and a deep learning model. The final goal is to have a pool of optimized algorithm that can be compared and used on board of satellites.

The thesis is structured as follows:

- Chapter 1: Introduction
- Chapter 2: Remote sensing and Optimization
- Chapter 3: Machine Learning and Deep Learning Background
- Chapter 4: Methods
- Chapter 5: Experimental Results
- Chapter 6: Conclusion

Chapter 2

Remote Sensing and Optimization

2.1 Multi-spectral images

2.1.1 Image acquisition

In the field of Earth observation, satellite images are captured using sensors that can be either **active** or **passive**. Active sensors, like RADAR, work by sending waves toward the Earth's surface and then capturing what is reflected back. Passive sensors, on the other hand, consist of radiometers that passively capture radiation emitted by the sun that either bounces off the Earth or is emitted by the Earth itself. A radiometric sensor works by combining multiple filters that capture a single band image in grey-scale, sensitive to a specific wave-length of light.

2.1.2 Reflectance

The electromagnetic radiation captured by the sensors is expressed by **reflectance**: the capability of reflecting radiant energy.

$$R = \frac{\Phi_e^r}{\Phi_e^i}$$

The reflectance R can be expressed as the ratio of the radiant flux reflected by that surface over the radiant flux received by that surface. This measure is important because it is possible to identify different type of surfaces: every material has a different spectral response at different wave-length.

2.1.3 TOA reflectance vs BOA reflectance

The reflectance measured by a passive sensor is defined as Top Of Atmosphere (TOA) reflectance. Moreover this measure not only the reflected radiation from the Earth's surface, but also the one reflected by clouds and the contamination from neighbour pixels. TOA is stored as Digital Number (DN), namely an integer number for storing purpose and sometimes need a specific normalization to convert it into reflectance (floating number). Through post processing technique such as atmospheric correction, it is possible to filter out the reflectance contribution of clouds and atmosphere while preserving the surface radiation, obtaining Bottom Of Atmosphere (BOA) reflectance.

2.2 Sentinel-2 mission

Sentinel-2 is a Earth Observation mission launched by the European Space Agency (ESA) in the Copernicus program. The mission initially consists in two twin satellite with a sun-synchronous orbit phased 180 degrees, with the goal of capturing multi-spectral imagery of the Earth surface with a 5 days revisiting cycle.

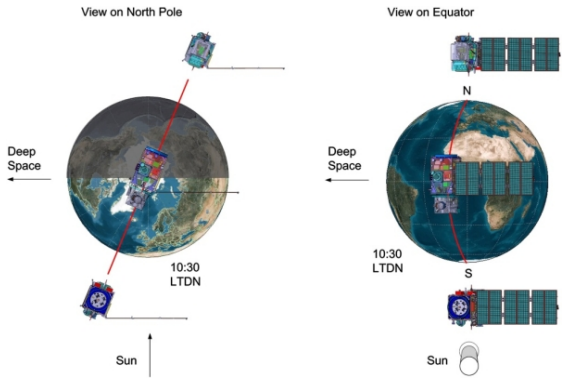


Figure 2.1: Sentinel-2 A/B satellites orbital configuration [1]

The multi-spectral images of sentinel are acquired by the MultiSpectral Instrument (MSI), that consists in multiple filters, mirror and detectors that capture the single band images.

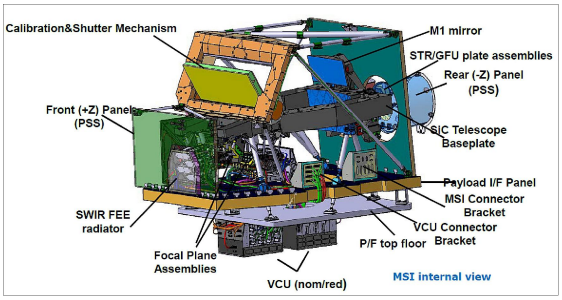


Figure 2.2: MSI instrument internal view (image credit: ESA)

Sentinel-2 MSI sensor delivers 13 spectral bands ranging from 10 to 60-meter pixel size in both near infrared and short wavelenght infrared. B4, B3, B2 are the three visible infrared bands (RGB), B9 and B10 are band used for water absorbtion, B11 and B12 are SWIR bands. The other bands are infrared with different wavelenghts used for different task.

Band	Resolution	Central Wavelength	Description
B1	60 m	443 nm	Ultra Blue (Coastal and Aerosol)
B2	10 m	490 nm	Blue
B3	10 m	560 nm	Green
B4	10 m	665 nm	Red
B5	20 m	705 nm	Visible and Near Infrared (VNIR)
B6	20 m	740 nm	Visible and Near Infrared (VNIR)
B7	20 m	783 nm	Visible and Near Infrared (VNIR)
B8	10 m	842 nm	Visible and Near Infrared (VNIR)
B8a	20 m	865 nm	Visible and Near Infrared (VNIR)
B9	60 m	940 nm	Short Wave Infrared (SWIR)
B10	60 m	1375 nm	Short Wave Infrared (SWIR)
B11	20 m	1610 nm	Short Wave Infrared (SWIR)
B12	20 m	2190 nm	Short Wave Infrared (SWIR)

Table 2.1: MSI spectral bands

2.3 Physical features

Fmask [2] [3] [4] and **Sen2Cor** [5] are two classical algorithm used for semantic segmentation of Sentinel-2 MSI images. These algorithms are often called "Physical": simple spatio-spectral physical features based on high or low reflectance values such as NDSI, NDVI, brightness and whiteness are used to differentiate different objects.

These feature can be obtained measuring the pixel reflectance in certain bands: for example knowing that green leaves absorb little energy, the TOA value in the visual NIR (band 8) will be higher for the vegetation, and so we can calibrate a threshold to discriminate clouds from vegetation.

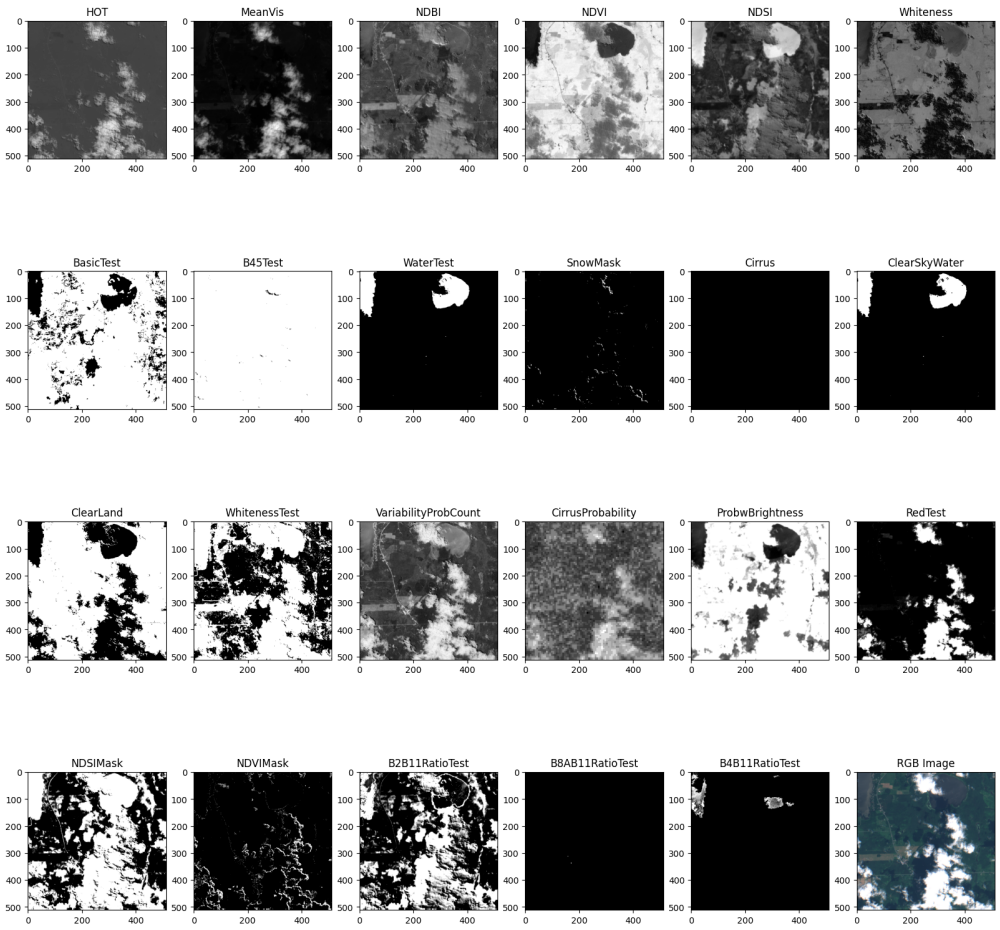


Figure 2.3: Overwview of some feature in greyscale extracted from one 512x512 Sentinel-2 IP at 10m resolution.

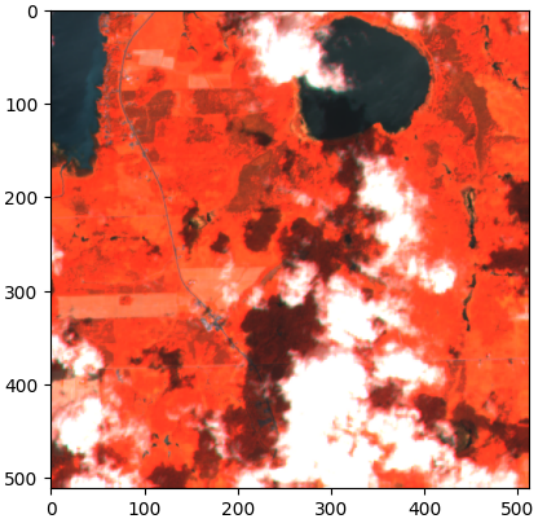


Figure 2.4: RGB composite of B8, B4, B3.

Fig. 2.4 is an RGB composite that is used to highlights vegetation: dense vegetation will appear as red pixels while yellowish-brown pixels would indicate a lower density or absence of vegetation. Building will appear as white. This is caused by the reflectance of chlorophyll on B8. The second RGB composite in Fig.

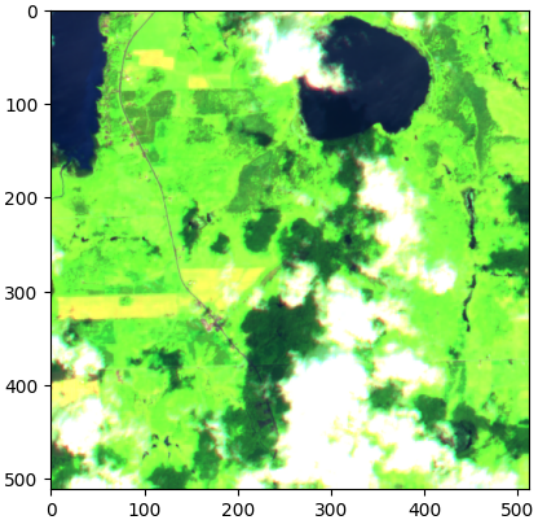


Figure 2.5: RGB composite of B11, B8, B2

2.5 is used also for vegetation but this time appears in green and it is useful to indicate healty crops.

2.3.1 NDSI

Normalised Difference Snow Index (NDSI) is the ratio of the visible infrared green band (B3) and one short wave infrared band (B11). Snow absorbs in the short-wave infrared light and reflects in the near infrared, but clouds reflects in both wavelenghts, making it a possible feature to differentiates between snow and clouds

$$NDSI = \frac{Band3 - Band11}{Band3 + Band11}$$

2.3.2 NDVI

Normalised Difference Vegetation Index (NDVI) is the ratio of the near infrared band (B8a) and the visible infrared red band (B4). Is an effective index to differentiates green vegetation from clouds, as green leaf high reflectance in NIR band and chlorophyll absorption in red wavelenghts.

$$NDVI = \frac{Band8A - Band4}{Band8a + Band4}$$

2.4 Fmask features

Fmask was designed for Landsat OLI/TIRS multi-spectral imagery. The TIRS (Thermal InfraRed Sensor) carries thermal infrared bands, but Sentinel-2 MSI (Multi Spectral Imager) sensor lacks. Zhu et al. [3] and Qui et al. [4] adapt the algorithm for Sentinel 2 imagery, making the thermal band optional. In this document, features or test that uses thermal band or auxiliary material (DEM, Snow metheorology map etc..) will be omitted to keep it simple.

NDBI Normalised Difference Built-up Index (NDBI) is the ratio of one NIR band and one SWIR band. Manufactured built-up areas such as urban zones are characterized by high reflectance in short wavelenghts and absorption in the near infrared, being an useful index to highlights non clouds features.

$$NDBI = \frac{Band11 - Band8}{Band8 + Band11}$$

Basic Test Fmask basic test try to highlights cloudy pixel using a threshold on B12, NDSI and NDVI: this test is an heritage to ACCA [6] algorithm used for Landsat 4.

$$BasicTest = Band12 > 0.03 \quad \text{and} \quad NDSI < 0.8 \quad \text{and} \quad NDVI < 0.8$$

MeanVis Mean visibility is the mean reflectance over the visible infrared bands 4,3,2 (RGB).

$$MeanVis = \frac{Band2 + Band3 + Band4}{3}$$

Whiteness Whiteness index was proposed by Gomez et al. [7] but was modified for Fmask because of the difference between the visible band of the OLI and the ENVISAT/MERIS sensors. The original index was derived by the overall brightness, Fmask uses the average value of the visible band instead.

$$Whiteness = \sum_{i=2}^{4} \left| \frac{Band_i - MeanVis}{MeanVis} \right|$$
$$WhitenessTest = Whiteness < 0.7$$

This test on the whiteness threshold is useful to exclude non white enough pixel and clear sky pixel with high variability in the visible band.

HOT test Haze Optimized Transformation (HOT) [8] it is used to separate haze and thin cloud from clear sky pixel. Because of high reflectance of objects in the visible bands, this test can highlights other features such as rocks or ice.

$$HOT\ Test = (Band2 - 0.5) * (Band4 - 0.08) > 0$$

B4/B5 Because rocks, sand and other objects reflect in short wavelenghts while clouds are more reflectant in the near infrared, it is useful to clean undesired objects from the HOT test. The name of this test comes from Landsat-8 OLI NIR and SWIR bands (band 4 and band 5), but for Sentinel-2 MSI are band 8 and band 11 (NIR and SWIR) [6]

$$B4/B5 = \frac{Band8}{Band11} > 0.75$$

Water test In the near infrared, water appears dark, but as we said before land features appears brighter. Knowing that NDVI value for land are usually above 0.1, so it is possible to differentiate water from land.

$$WaterTest = (NDVI < 0.01 \text{ and } Band8 < 0.11) \quad \text{or} \quad (NDVI < 0.1 \text{ and } Band8 < 0.05)$$

Cirrus test Cirrus are cloud forming wispy filamentous at high altitudes made of ice crystals. These clouds have and high reflectance in the near infrared, but can be confused with ice when covers high mountains (3000m+)

$$Cirrus\ test = Band10 > 0.01$$

PCP Potential cloud pixel (PCP) is a mask of pixels that passed the first four tests: Base, Whiteness, HOT and Nir/Swir. If the PCPs covers 99.9% of the image, then the algorithm will return the PCP mask as the cloud mask because there are not enough land pixel to continue the masking process.

$$PCP = BasicTest \quad \text{and} \quad WhitenessTest \quad \text{and} \quad HOT\ test \quad \text{and} \quad B4/B5$$
$$PCP = PCP \quad \text{or} \quad CirrusTest$$

Snow mask This is a simple test to identify possible snowy pixel, it takes NDSI, green and NIR bands to differentiate possible snowy pixels.

$$NDSI > 0.15 \quad \text{and} \quad Band8 > 0.11 \quad \text{and} \quad Band3 > 0.1$$

Brightness Normalized brightness probability of each pixel is computed as:

$$Brightness = \frac{\min(Band11, 0.11)}{0.11}$$

Water can have reflectance as high as 0.11 in short wavelentghs, so it is normalized over 0.11. Turbid water can have higher reflectance value than normal water caused by sediments. This feature is used in the algorithm to compute the water cloud probability.

Variability Land pixels are higher variability than other pixel: the distribution of the reflectance value between visible and nir/swir band is wide. The authors before calculating the variability, take in account modified NDSI, NDVI and NDBI, excluding from the calculation (setting to 0) saturated pixels in the visible bands to avoid wrong variability value for cloudy pixels.

$$Variability\ probability = 1 - \max(|modified\ NDVI|,\ |modified\ NDSI,\ |modified\ NDBI|,\ W$$

Saturated pixel can be retrived from the quality mask of L1C auxiliary data. If the quality mask is missing, classic indexes will be used, leading to miscalculation.

ClearSkyWater This feature highlights clear sky pixel that are water. Is it useful to identify clear land pixel for final classification.

$$ClearSkyWater = waterTest \quad \text{and} \quad (Band12 < 0.03)$$

This test was modified in Shi Qiu et al. 2018 [4].

$$ClearSkyWater = waterTest \quad \text{and} \quad \text{not } PCP$$

ClearSkyLand Clear sky land pixels can be identified taking in account pixel that are not possible clouds and that are not water. It is a simple yet effective feature for land mask.

$$ClearSkyLand = \text{not } waterTest \quad \text{and} \quad \text{not } PCP$$

Further processing After these feature are extracted, cloud probabilities for land (lCloudProb) and water (wCloudProb) will be computed, then the final mask will be computed as the union of cloudy pixel over water and cloudy pixel over land

$$\begin{aligned} cloudmaskwater &= PCP \quad \text{and} \quad WaterTest \quad \text{and} \quad wCloudProb > WaterThreshold \\ cloudmaskland &= PCP \quad \text{and} \quad notWater \quad \text{and} \quad lCloudProb > landThreshold \end{aligned}$$

waterThreshold and landThreshold are dynamic threshold that are defined as follow

$$\begin{aligned} waterThreshold &= 82.5\% \text{ of } clearskywater + CloudThreshold \\ landThreshold &= 82.5\% \text{ of } clearskyland + CloudThreshold \end{aligned}$$

CloudThreshold is set to 20% for Sentinel-2 imagery.

2.5 Sen2Cor features

Sen2Cor is the processor made by ESA to generate L2A user products. It includes a collection of algorithm for scene classification and atmospheric correction, implemented for Sentinel-2 MSI. The cloud detection algorithm consists in 7 step where for each one some feature are extracted. For each step, the probability is normalized between 0 and 1, where 0 is for no cloud and 1 is high confidence cloud.

Brightness The first step of the cloud detection algorithm is based on a brightness test on Band 4 (visible red)

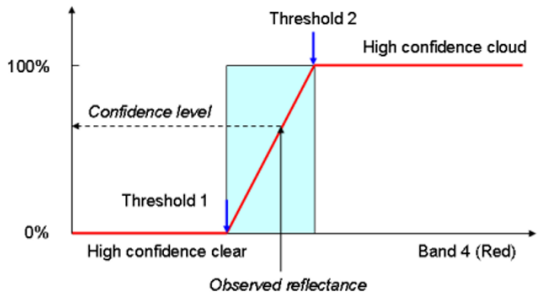


Figure 2.6: Brightness test on red band [9]

NDSI test NDSI is used to make a first differentiation between possible false positive cloud pixel with high reflectance similar to snow.

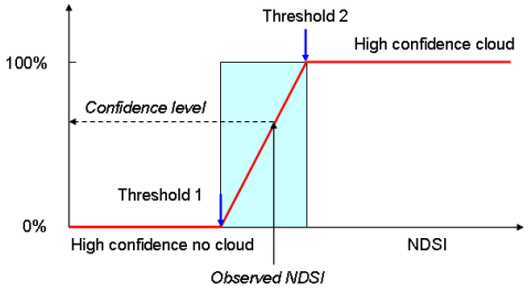


Figure 2.7: NDSI test [9]

Snow loop and snow features This step consists in a snow loop, it is needed to discern snow pixel from high brighness clouds. In the original algorithm the snow loop is used only when the area covered by the tile come from a region of the world where it can be snow or ice. For this task MODIS snow climatology database is used as auxiliary data. In the latest version a set of 12 "Snow Tri-MONTHLY climatology" images it is used. These auxiliary data are based on 52 snow conditions images from ESA CCI data package for each month of the year for the period of 2000-2010 with +/- one month margin.

NDVI test This test is used to differentiate vegetation from clouds as in Fmask.

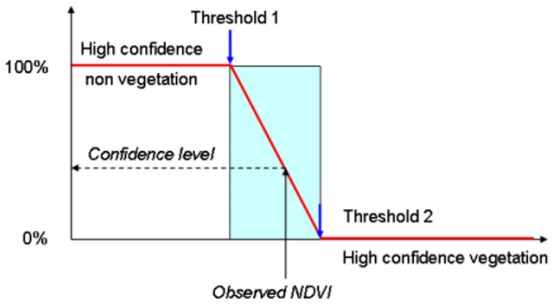


Figure 2.8: NDVI test [9]

To get the probabilities, this time the normalization is from 1 to 0, because NDVI reflectance is higher for vegetation.

Non vegetation Blue/Infrared (B2/B11) ratio is used to detect non-vegetated pixel (like bare soils) if the reflectance ratio fall below a certain threshold, and if it exceed a certain threshold then it could be a water pixel. Additional test on infrared region (B12) and blue region (B2) are added to detect thin clouds over soil and water.

First only pixels that exceed a certain threshold on the blue band can be tested. This threshold is computed linearly between 0.15 and 0.32. Then a test on the ratio B2/B11 is done as said above.

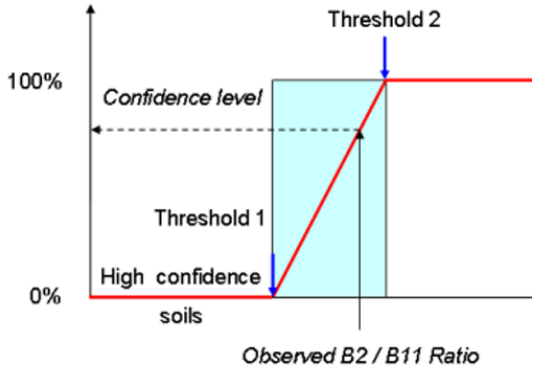


Figure 2.9: Non vegetation test [9]

Water body The test is conditioned by a threshold on B12 that varies linearly depending on B2/B11, as the same as in the previous test. Two additional condition are necessary for B2/B11 high reflectance value to be classified as water: their reflectance in blue band B2 shall be below 0.2 and NIR (B8A) less than red (B4).

The test for water bodies pass if the pixel is below a certain threshold, because the B2/B11 ratio is high for water bodies and not for cloud, in this case an inverse normalization is employed.

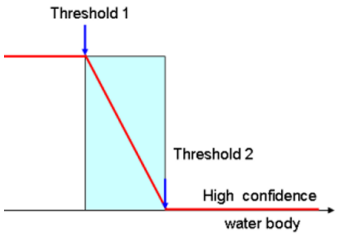


Figure 2.10: Water body test [9]

Additional conditions To ensure that no water pixel were missed, a check on the unclassified pixel is made: if the B2-B4 is higher than a certain threshold and B8A reflectance is less than B4 reflectance then if B2 is below a certain threshold these pixel can be classified as water. This step is not really useful if we want to classify only cloud, but can be useful to exclude pixel from future calculations.

High brightness objects Sand and rocks are object with high reflectance in band 11 while in band 8 cloud are more reflectant. Using the B8/B11 ratio, if a value exceed a certain threshold then is classified as cloudy pixel. To enter this test, B2 reflectance has to be below a certain threshold computed linearly in function of B8/B11 ratio. It helps to keep thin clouds detection over desert regions.

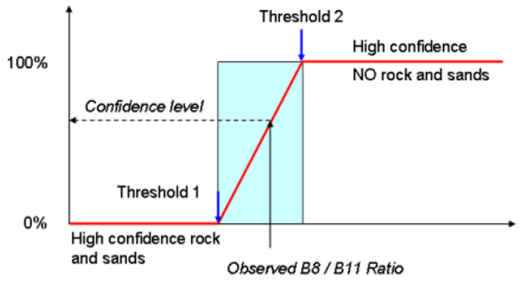


Figure 2.11: High brightness objects test [9]

Additional conditions If B2 is less than 80% of B11 reflectance and pass the B8/11 test (less than threshold 1), then it can be classified as non-veg. As stated above, this is only needed to speed up calculation.

False positive pixel test This step is used to eliminates false-positive cloudy pixel with high B4/B11 ratio. If B4/B11 exceed a certain threshold the pixels are set as non-cloud

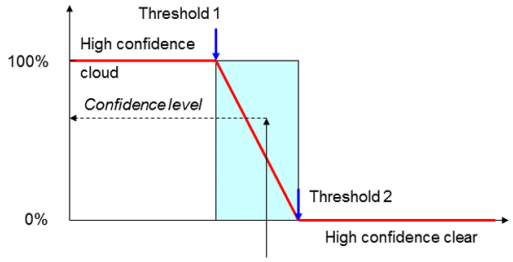


Figure 2.12: False positive pixel test [9]

Further processing Each of this feature is sequentially extracted and then applied on the cloud mask and classification mask: first the cloud mask is initialized as the brightness test and for each features the cloud mask's per pixel probability is multiplied with the feature's per pixel probability, setting at 0 high confidence features such as water or vegetation. In the last step of the cloud detection algorithm, cirrus cloud are identified with a threshold on B2 and on B10 (Cirrus band). Finally the cloudy pixel with probabilities higher than 20% will be labeled as clouds.

2.5.1 Post processing filters

Threshold tests, based on physical spectral features, can often misclassify small clump of pixel (for example, snowy mountain's peak or bright object) or marginal cloudy pixel of thick clumps. The noise created by bright pixels or other reflectance value that can be confused as clouds can be removed with **Median Filter** (**MF**) or **Binary Erosion** (**BE**). Firstly the MF uses a sliding window (for example a 5x5 square) over the image, where at every step it replaces the center with the median value of the image, smoothing the edges and removing noisy pixel. Binary Erosion is a morphology basic operator that taking a kernel K (for example 3x3), namely a structuring element (for example a disk or a square binary mask) as a binary mask, superimpose its center to each pixel of the image I: if K is completely contained by I the pixel is retained, else removed. Since the impact on pixel removal of BE is higher than the MF, often a **Binary Dilation Filter** (**BD**) is employed after the binary erosion. As for BE, binary dilation is a basic morphology operator that works the opposite as BE: given the kernel K and image I, it superimpose the pixel of I that has value equal to the filter (for a binary mask it is equal to one), turning every pixel superimposed on K equal to that value. Furthermore this filter is useful

to expand the margin of an object (possibly removed by BE) but could also expand clump of misclassified pixel (noise).

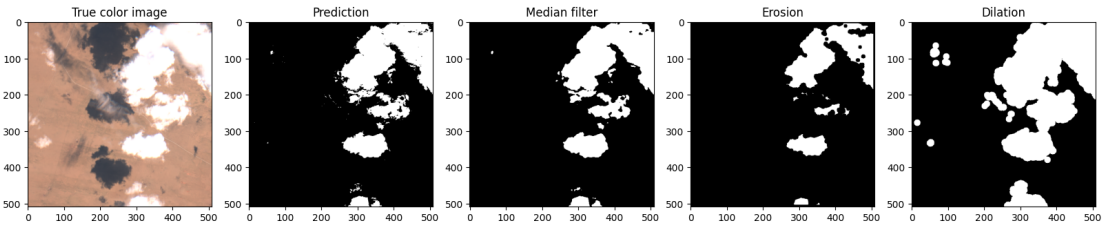


Figure 2.13: Post processing filters applied on a binary mask.

In Fig. 2.13 it is possible to see how each filter remove or dilate pixels, emphasizing how the binary erosion can delete important information.

2.6 Genetic Algorithm

Genetic Algorithms (**GA**) are a subfamily of Evolutionary Algorithms, based on *natural selection* to approximate solutions for a given problem. The concept behind lies into defining a population of individual that through a **fitness function** (e.g. score) can be selected as candidates to create an offspring for the next generation. Each individual is identified by a **genome**: a set of gene (e.g. variables) that serve as parameters for the fitness function. At each generation, two or more individual will be selected as parents to generate an offspring, using genetic operators.

- **Crossover**: the offspring genome will be composed by a portion of one parent concatenated to the one of the other.

- **Mutation**: by simply change one bit (or more) of the genome of one parents, a new offspring is created

At the end of each generation, the offspring will be added to the population and the fitness function of each individual will be evaluated, creating a new population where only the fittest will survive: this step is called Elitism. Iterating this process for multiple generation can be effective to find new solution to a problem.

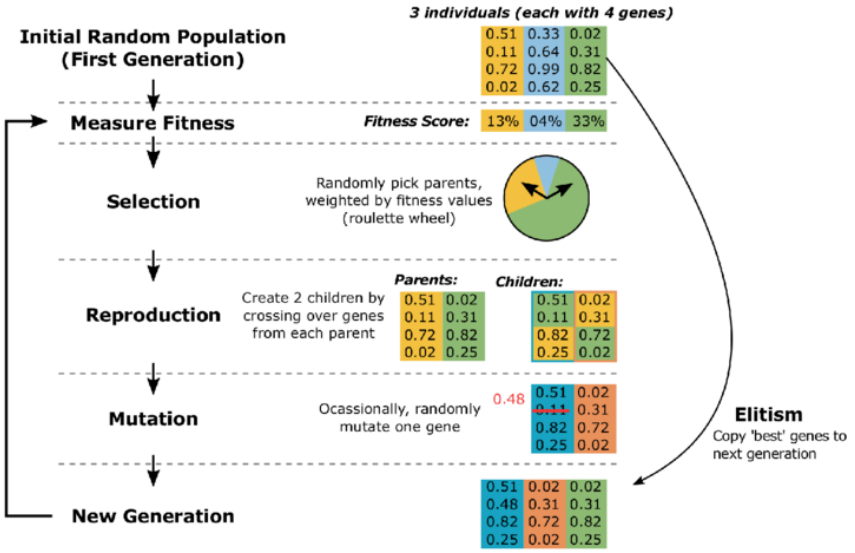


Figure 2.14: Example of a Genetic Algorithm [10].

Chapter 3

Machine Learning and Deep Learning Background

3.1 Machine Learning

Machine Learning (**ML**) is a field of study in Artificial Intelligence (AI), consisting in the development of algorithms that do not follow explicit instructions, but learn from data to recognise patterns and making statistical decision. These statistical models are powerful because can generalize over unseen data and perform multiple tasks like object detection, speech recognition and classification. The approaches to ML algorithms differs depending on the learning paradigm:

- **Supervised Learning**: data and label (desired output) are known, meaning that given a representation of the data through a feature vector, the model is capable of learning through an optimization function, comparing the output and label given with the data. Classification and regression are supervised tasks.

- **Unsupervised Learning**: in this case only the data are given, so instead of learning through feedback, in this case the mathematical model try to catch the similarities of the features, grouping and categorizing the data. Example of unsupervised tasks are clustering and dimensionality reduction.

- **Reinforcement Learning**: is used for task like autonomous driving and game agents that learn how to play a game. This technique is based on the concept of cumulative reward: taking actions interacting with the environment, the model will be rewarded if it takes right decision; otherwise, it will be penalized.

3.1.1 Support Vector Machines

Support Vector Machines (**SVMs**) [11] is a supervised learning algorithm which aims to separate the classes through a straight line (or an hyperplane, in more than 2 dimensions), by maximizing the margins between the classes.

Furthermore, SVMs provide a natural way to achieve non-linear separation without the need for an explicit expansion of the features (using **kernels**): this method enable to find a better hyperplane in a larger dimensional space. SVMs output cannot be directly interpreted as class posterior probabilities.

- **Hard SVM**: it is used when data is linearly separable, so it doesn't let the points lie within the boundaries (margin).
- **Soft SVM**: it soften the restriction of the Hard SVM, so it let the points be into the margins. This is the one that is mostly used in real life scenarios.

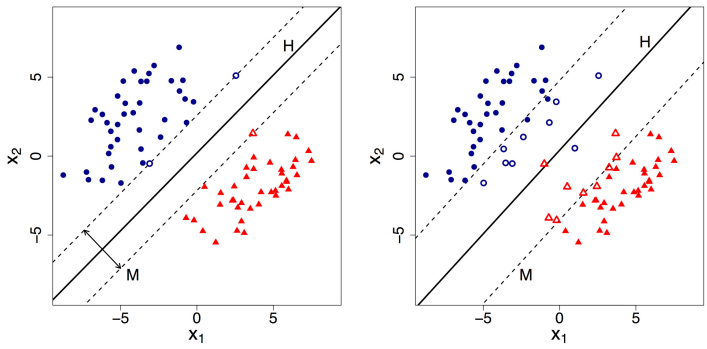


Figure 3.1: On the left Hard SVM, on the right Soft SVM [12]

Kernel trick

Finding linear separable data it is hard to encounter, even with softer restrictions it could be difficult, so projecting data to a higher dimensional space could be the solution.

$$\phi : X \mapsto \mathbb{R}^n$$

Kernel is a method that enable to solve non-linear problem through implicit mapping of the data to an higher dimensional space; It is defined by means of pairwise similarities.

$$k(x, z) = \langle \phi(x), \phi(z) \rangle$$

Reducing the computations as the inner product between the feature mappings, without the need to apply the transformations explicitly. The most common known kernels are:

- Polynomial: $K(x_i, x_j) = (x_i \cdot x_j + 1)^p$
- Gaussian: $K(x_i, x_j) = \exp\left\{\frac{-1}{2\sigma^2(x_i - x_j)^2}\right\}$
- Radial Basis Function: $K(x_i, x_j) = \exp\{-\gamma(x_i - x_j)^2\}$

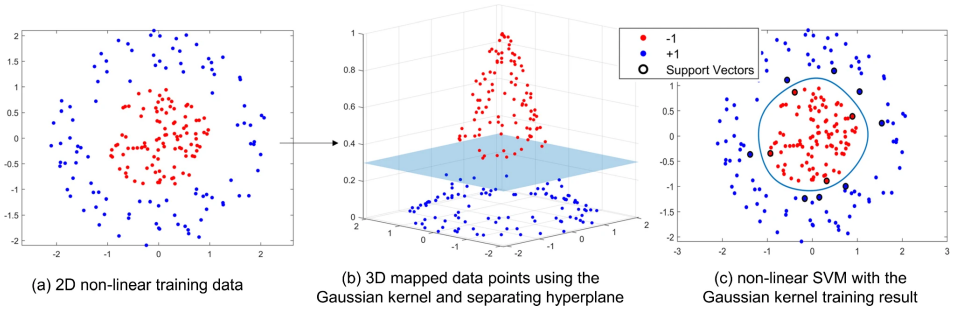


Figure 3.2: An example of training the non-linear SVM with the Gaussian kernel (The circled data points are the support vectors) [13]

3.1.2 Decision Trees

Decision tree [14] is a supervised learning algorithm used for classification and regression, that consists in a hierarchy of simple decision rules taking the shape of a tree. The tree is constructed through a greedy top down approach, known as recursive binary splitting, that split the tree in non-overlapping regions representing the labels. The node splitting depends on the "purity" metric: It refers to how

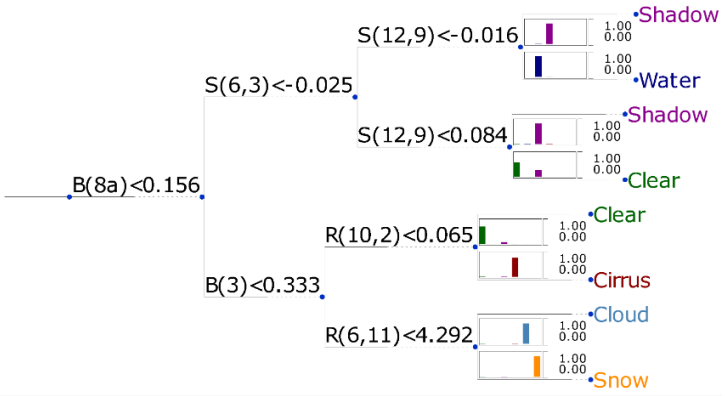


Figure 3.3: Example of a Decision Tree [15]

homogeneous the data is within a group; higher the purity, higher number of data in that group belongs to the same class. Two main metrics are used for classification:

- Gini impurity: $1 - \sum_i p_i^2$
- Information gain: $1 - \sum_i p_i \log_2 p_i$

3.1.3 Ensemble method

Ensemble learning consists in exploit multiple models, called weak learners, to combine their decision and overcome their limitations. Each weak learner is trained on the same data (or distribution of data).

Bagging

Bagging Ensemble works by training independent base models and taking as output the majority vote on a class (classification) or the average of the outputs (regression). Before feeding the data to the weak models, there is a bootstrap phase where from the original dataset a new bootstrapped set of data is created for each model; these set are composed by repeating or selecting only a subset of samples or features.

Boosting

Instead of training independent models, Boosting consists in training one model after another, propagating the misclassification by weighting the data for the next model. This technique is useful to reduce bias.

Stacking

A Stacking is a group of output generated by training individual weak estimators and is used to train a final estimator.

3.1.4 Random Forests

Random Forests [16] (RF), also known as Random decision forests, is a bagging technique that uses multiple decision trees as weak learners. Each tree is trained on random independent sets generated from the training set and the output is decided by a majority vote of the models.

3.2 Neural Networks and Deep Learning

Neural Networks (**NN**) are a type of algorithm that are designed to work similarly to the human brain. Likewise a real brain, the main elements of a neural network are the neurons (main computational unit) and synapses (connection between neurons). The concept of neuron can be defined as a threshold function

(called activation function) applied to the linear combination of the inputs and the weights of the synapses.

$$z = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b = \sum_{i=1}^{n} w_ix_i + b$$

$$a = \sigma(z)$$

A network with a single neuron is called **Perceptron**, while combining multiple perceptron in parallel form a Layer, named **Multi Perceptron Layer** (**MLP**). A neural network is composed by three main parts: the input layer, the hidden layers and the output layer; While the input layer have a number of neurons defined by the number of inputs, the output layer neurons depends on two thing: the task and what it is needed, for example in a classification task the neurons will be equal to the number of labels. **Deep Learning** (DL) concerns neural networks with multiple hidden layers also known as **Deep Neural Networks (DNN)**.

3.2.1 Convolutional Neural Networks

Convolutional Neural Network (CNN) is among others, one of the most used type of neural networks. Through mathematical operation called **convolution**, the model has the ability to reduce the feature space while learning high level features, enabling its application on complex task that uses grid-like data such as images in computer vision. While in the initial part of the network it can learn low level features, in the deeper part it can recognize features like single object in an image. Main elements of a CNN are:

Convolutional Layer

Convolutional layers learn filters that capture spatial hierarchies in data. The convolution operation is made by a small matrix named kernel (where K is the number of kernels), performing an element-wise multiplication and sliding by a factor called stride S (number of pixel). It is important to note that depending on which output dimension is desired, it could be needed to add a padding P on the image to ensure that the filter fits the entire input (especially near the edges). Given an image of height H and width W, the output size can be computed as:

$$\text{Output Volume} = \left(\left\lfloor \frac{(W - K + 2P)}{S} \right\rfloor + 1\right) \times \left(\left\lfloor \frac{(H - K + 2P)}{S} \right\rfloor + 1\right)$$

Since the convolution is a linear operation, as it was stated before to add non-linearity an activation function in employed on the output.

Separable Depthwise Convolution

Separable Depthwise Convolution (SDC) is a form of factorized convolutional layer split in a separate depthwise convolution and a separate pointwise convolution. While a classical convolution filters and and combines the inputs, the depthwise convolution first apply a signgle filter to each input and then the pointwise combine the filtered inputs. This operation reduce up to 9 time the number of weights of the layer. [17]

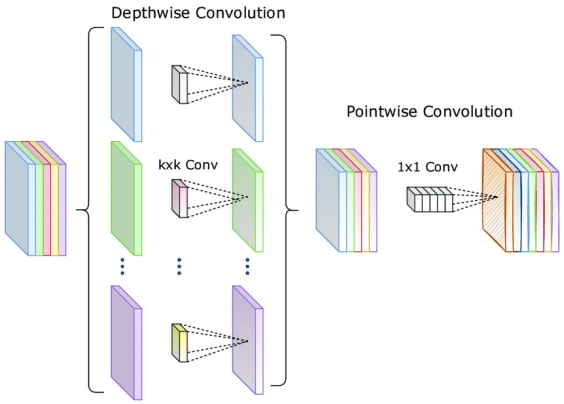


Figure 3.4: Depthwise Separable Convolution [18]

Pooling Layer

Pooling is a **downsampling** operation used to reduce the feature spatial dimension, resulting in a reduction of computation and risk of overfitting. There are two type of pooling:

- **Average Pooling**: return the average value of the kernel window.
- **Max Pooling**: return the max value of the kernel window.

Fully Connected Layer

The Fully Connected Layer (FCL) is the final part of a CNN, consists in a MPL that takes in input the high level features extracted through the consecutive convolutional layers, namely feature extractor.

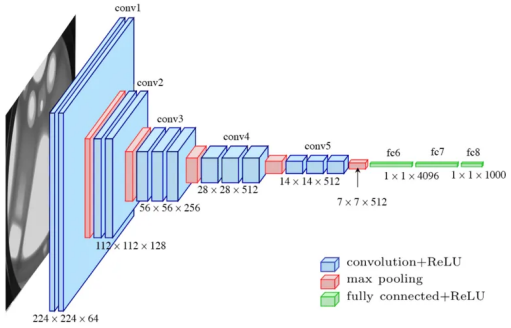


Figure 3.5: Example of Convolutional Neural Network architecture [19].

3.2.2 U-Net

U-Net [20] is a fully convolutional neural network used for segmentation tasks which consists of a contracting path (**Encoder**) followed by an expansive path (**Decoder**). It is convenient to use it because, through upsampling operation such as inverse convolution, allow to produce output images with the same size as the inputs.

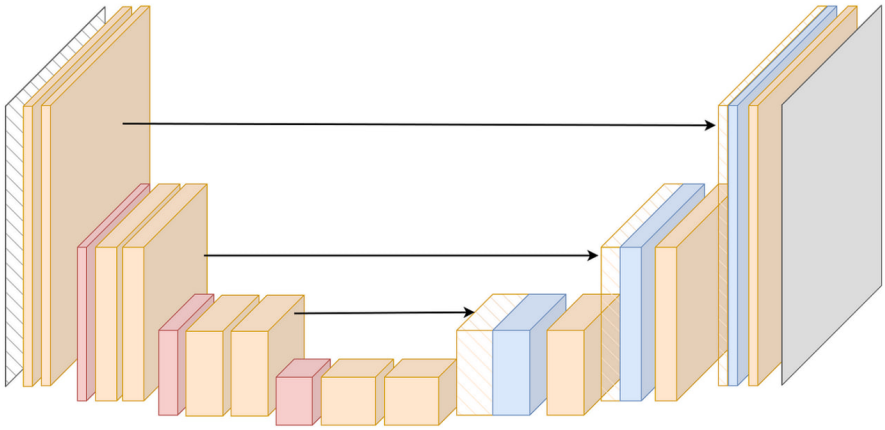


Figure 3.6: U-Net architecture [21]

The difference between this architecture and other antoencoder is the skipping connection between the encoder and the decoder; the main idea behind this is to concatenate, after the upsampling operation, the feature extracted at each step of the encoder with the upscaled feature from the decoder to keep the information of the original image, because after the bottleneck (the last part of the encoder) the general information (low level features) would be lost and reconstructing the image could be harder.

Encoder

The Encoder of a U-Net consists in a CNN with multiple blocks composed by: convolutional layer, activation function and a downsampling operation (strided convolution or pooling operation). It aims at extracting higher level feature at each block, that will be passed with skipping connectiong to the decoder.

Decoder

This part of the network aims at reconstructing the image, usually a segmentation mask, starting from the high level feature of the bottleneck. It has a symmetrical structure to the Encoder, but the downsamplig operation at each block is replaced with an upsampling operation to expand the feature maps. After each upsampling operation, the new upscaled feature map is concatenated with the feature map at the same level of the encoder. Some of the most used upsampling operations are:

- Interpolation: it uses a deterministic mathematical formula to increase the resolution of a feature patch.
- Pixel shuffle: it shuffles the feature channels, reducing the number of channel by the squared upsampling factor. Before this operation a convolutional layer is needed to increase the number of filters [22]
- Transposed Convolution: opposed to the classic convolution, this operation expands the dimension of the features, multiplying each element of the feature patch with the kernel. Differently from the other two methods, this operation has learnable parameters.

3.3 Related Works

3.4 Machine Learning based

In classical machine learning the Cloud and cloud shadow detection (CCSD) task can be seen as a supervised classification problem described as the mapping of input data to a fixed and finite set of labels (e.g. cloud, cloud shadow, snow). Usually the feature space of the input data is not used as it is, but feature engineering is needed for complex data, to select, creating, transforming and extracting features from the existing raw data. Band channels and simple band functions can be used, like band-differences, band-ratios or indices.

Hollsten et al. [15] in their work focus the attention on two ready-to-use model: Decision trees and classical (i.e. not naive) Bayesian classifier, showing how the optimization process in the feature engineering part it is crucial to get better results.

The Decision Tree algorithm can be seen as an automated method of rule based algorithm: it is a hierarchy of threshold on single features, with decision paths as branch.

Moreover the depth in this algorithm plays an important role: increasing the depth increase the complexity of the algorithm, adding new branches and using more bands or mathematical function. This can lead to an increase in the ratio of correctly classified spectra or in the worst case to overfitting. The Bayesian classifier instead of making decision trees, is based on the joint probability of each instance with a feature space F. The joint probability it is estimated for each class C and feature space F of the instance: the class with the highest probability will be the label. Moreover a confidence measure is used to post process the classification mask, taking in consideration only the clear sky pixel for which the classifier is very certain.

Zupanc [23] developed the Sentinel Hub Cloud Detector, a LightGBM (Grandient Boosting Machine) machine learning model used in the "s2cloudless" framework. The algorithm is a more optimized and faster version of classical decision-tree function, that build histogram decision tree in a leaf-wise fashion, while the classical approach is row-wise. The model was trained on MAJA [24][25] labels and validated on S2-Hollstein [15] data. While it was trained on MAJA's data, s2cloudless outperform MAJA in term of cloud and cirrus classification, but not for land and snow misclassifcation ratio.

3.4.1 U-net

Multiple research in CCSD that exploit neural networks use this architecture like in KappaMask [26], UNetMobV2 [27] and DL-L8S2-UV [28]. López-Puigdollers et al. [28] propose a modified U-Net architecture with only two downsampling steps and using depthwise separable convolutions have achieved significant reduction of parameters: in fact the architecture has only around 1% of the original U-Net parameters and can compute cloud masks of 256 x 256 image patch.

Moreover in the training process of this model, the authors exploit transfer learning, training the model only on Landsat-8 data and using sentinel-2 as benchmark in the validation phase. Domnich et al. [26] is heavier compared to DL-L8S2-UV, but while the latter only uses 3 output classes, the former has 6 classification classes: higher accuracy at the price of higher processing time. However neural network can be speed up by using hardware accelerator. Cesar Aybar et al. [27] uses a MobileNetV2 as backbone for the encoder, resulting in a lightweight U-Net.

Chapter 4

Methods

4.1 Problem statement

This work addresses the problem of cloud detection on-board satellites, that is a semantic segmentation task over multi-spectral imagery to detect clouds, by optimizing and comparing different algorithms. The final goal is to find an optimized family of algorithm that can be efficient and precise. Moreover the two major challenge to overcome are:

- **Latency**: Processing images in real-time

- **False Positive Rate (FPR)**: since cloudy pixel will be exploited for image compression, it is necessary to minimize the false cloudy pixel to avoid compressing useful information about the observed environment

4.2 Dataset

Cloudsen12 [27] is the dataset used in this work, consisting of 49,400 image patches (IP) evenly spread through the globe. Each IP covers 5090 x 5090 meters and contains different type of data such as Sentinel-2 Level-1C, Level-2A, Sentinel-1 SAR, Digital Elevetion Model (DEM) and other ancillary data. The dataset is designed to support the training of deep learning models (self/semi/supervised), in fact it includes three type of hand-crafted annotation: high-quality, scribble and no-annotation. For this work, only high quality annotations (around 10000 IPs) at 10 meter resolution (509x509 images) were used to train and test the models.

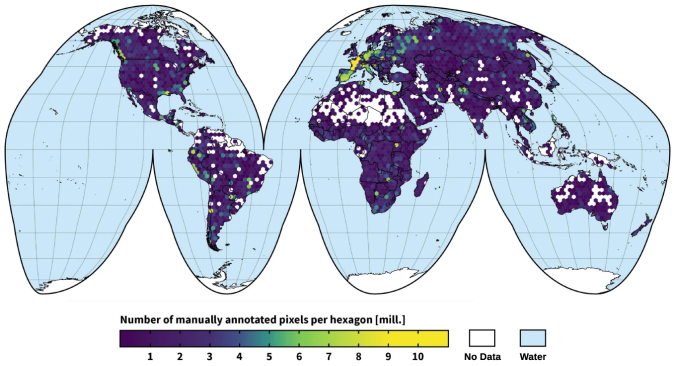


Figure 4.1: CloudSEN12 spatial coverage, purple-to-yellow color gradient represents the amount of manually annotated pixels per hexagon. The annotated pixels were collocated in an equal-area hexagonal discrete grid with a facet size of 140 km. [27]

The dataset annotations are divided into 4 classes: clear (clear-sky), thick cloud, thin cloud and cloud shadow; these classes were grouped into two superclasses as shown in Tab. 4.1.

Code	Class	Superclass
0	Clear	non-cloud
1	Thick Cloud	cloud
2	Thin Cloud	cloud
3	Cloud Shadow	non-cloud

Table 4.1: Mapping of the dataset annotations into the relative superclasses

4.3 Physical algorithm

Physical algorithm are often really fast, because are based on simple threshold operation, but the main problem is the false positive rate; cloud probabilities are computed on spectral features that are often confusing when similar spectral response are occurred in different object: for example snow, turbid waters or bright objects like rocks can have similar reflectance values. Moreover to solve this problems, auxiliary data such as elevation (cirrus) and climatology maps (snow and cirrus) are used to discern bright surface from clouds. The problem comes since in orbit it is not possible to have updated auxiliary data overtime, so the two physical algorithm used for this study have been re-implemented without using ancillary data, namely:

- **Sen2Cor cloud detection algorithm (sen2cor)**

- **Function of mask (Fmask)**

To select the best algorithm, analysis on computation complexity and postprocessing were employed. For the latter, since physical algorithm has no spatial awareness on the image but operates only on single pixel features, different combination of the following three filters were used:

- **Median Filter (MF)**
- **Binary Dilation (BD)**
- **Binary Erosion (BE)**

Finally sen2cor was optimized using integer operation instead of floating point operation to study the trade-off between latency and classification.

4.3.1 Algorithm complexity evaluation

To better analyse the difference in term of computational capability of the two above mentioned algorithm, test regarding classification metrics and number of basic operation per pixel were made.

Pixel wise operations

Physical algorithm for cloud segmentation are highly affected by the number of input pixels and the number of operation like multiplication and division. Below are reported the number of worst-case single pixel wise operations and the specification for each algorithm. Other operations (single value multiplication or filters) are omitted.

- **Fmask**: 13d + 19m + 41a
- **Sen2Cor**: 13d + 15m + 24a

Where "d" stands for division, "m" for multiplication and "a" for addiction/subtraction. It is worth to note that Fmask has 2 sum() and one cumsum() operation that depends on the total number of image. While on fmask most of the features are boolean map, Sen2Cor works mostly on probabilities (floating points), but some features limits the multiplication of the probability map only on a subset of pixel. Moreover this implementation of Fmask uses 7 bands while sen2cor uses 8 bands in total. Since Sen2Cor

4.3.2 Integer optimization of Sen2Cor

Microprocessor and processor often works better (faster) with **integer operation**. Since the original code of sen2cor works on float numbers, two integer version of Sen2Cor were implemented in C: the first one, when it is needed to pad a number (to avoid float or number between 0 and 1), power of 10 were used, while in the second version, instead of multiplying, binary left shift were used on number. For example if it is needed to multiply by 100, instead a left shift of 7 ($2^7 = 128$) is employed. Instead of using TOA, Digital Number (original pixel value) of each pixel was used, dividing it by 10 to slightly reduce de size of the variable.

The main challenges of this optimization were:

- **Integer size**: TOA values are between 0 and 1, that in DN it translates in between 0 and 10000 (or greater in case of saturated pixels). Dividing by 10 allow to use Int32 (size of 32bit) variable without risking significant overflows.

- **Cloud probabilities**: to keep the probabilities between 0 and 100, as opposed to the float version, more division and products by 100 (or left shift by 7) were needed in order to normalize it.

- **Normalization and bands ratios**: Normalization and ratios often return float values between 0 and 1; to keep these values as integer, inverse formula and approximation were needed.

For the last challenge, here there are the two major example of code adaptation:

Algorithm 1 Min-max normalization

Ensure: x is a pixel value
  if $x < min$ **then**
    return min
  else if $x > max$ **then**
    return max
  end if
  $num \leftarrow x - min$
  $den \leftarrow max - min$
  if $den = 0$ **then**
    return $int32_max$ ▷ Return the max value that can be contained into an int, division by 0
  else return $(num * 100)/den$
  end if

In Alg.1 it can be seen how a simple normalization (2 sum and one division) needs more operations. Computing a normalized difference index (eg: NDVI) it is similar but, for example, computing a dynamic threshold can add more operation than needed with floating point operation as shown in Alg.2.

Algorithm 2 Dynamic threshold normalization

Ensure: x is a pixel from band i
Ensure: y is a pixel from band k ▷ min and max are bounds for normalization
▷ thr are the thresholds for pixel band test

```
if x = 0 then
    return (1000 * max1)/max2
else if y = 0 then
    return (1000 * min1)/min2
end if
if thr12 = thr22 then
    num ← y * thr11 + x * thr21
    den ← x * th12
else
    num ← y * thr11 * thr22 + x * thr21 * thr12
    den ← x * thr12 * thr22
end if
if x < (y * min1)/min2 then
    return (10000 * min1)/min2
else if (x > (y * max1)/max2) then
    return (10000 * max[0])/max[1]
else
    return (x * 10000)/y
end if
```

4.3.3 Sen2Cor threshold optimization

As it was stated before, sen2cor is a physic algorithm that works with threshold based test on the image bands. The thresholds of the algorithm can be tuned in order to make the algorithm less prone to false positive. The techniques used for this optimization were:

- **Shifting thresholds**: since every test has a couple (min and max) of thresholds, a right shift of a percentage of these was employed. This will raise the min threshold and increase the level of uncertainty for higher probabilities pixel.

- **Manual calibration**: the cirrus test, since is the last step and is not involved in the computation of the cloud pixel probabilities, was calibrated manually.
- **Genetic algorithm**: tests with dynamic threshold are difficult to be calibrated, so in order to optimize it a genetic algorithm was used.

Shifting thresholds

Each couple of threshold (for the same test) were shifted in the same direction: right for cloud test, left for non cloud test (eg: Vegetation); only the non dynamic thresholds, from the first step to the sixth bis, were tuned, shifting from -90% of the threshold value to 90%. The direction of shifting threshold is bound to the meaning of the test: test for cloudy pixel has a lower bound in which lower values are labeled as "not cloud", while values greater than the upper bound are clouds. Furthermore a linear probability distribution is computed for the values between the two threshold; shifting the thresholds adds a layer of uncertainty for lower cloud probabilities.

Cirrus manual calibration

Cirrus are difficult to classify because these can be confused with snow at high altitude. With the 10th band (short wave-length infrared) it is possible to calibrate the prediction. During the study it was observed that minimal changes on the threshold parameters would bring considerable decrease in false positive rate. A manual right shift on the threshold was employed.

Genetic Algorithm

Shifting threshold technique and manual calibration were applied on static threshold, but the tests on water, land and bright object (Step 5, 6 and 6bis) uses dynamic threshold for additional constraints on specific wavelength. Since these tests can be tricky to calibrate, a simple **genetic algorithm** was designed in order to find a possible solution space evolving the thresholds:

- **Genome**: composed by the thresholds and bounds used to compute the dynamic thresholds

For the **genetic operator** two types of crossover and mutation were used:

- **crossover**: classic single cut crossover, splits the genes at a random point, taking one part from each parent.
- **crossover2**: it's a classic crossover but perform a slightly mutation on the genes, increasing or decreasing each gene.

- **mutation**: it increase or decrease by 0.1 one gene.

The population is created by taking the default thresholds of sen2cor and making random mutation on each gene. At the end of the population generation, the default individual is added as the best individual of the first generation (fittest). Lastly to choose the parents, a tournament of 2 individual (based on best fitness) is used.

4.4 Machine Learning

Machine Learning algorithm has been proven effective for cloud detection, such as decision trees [15] and support vector machines [23]. To make a better benchmark comparison with physical algorithm, two ML models were trained and compared: a **SVM** model and a **Random Forest**.

4.4.1 Data sampling and preprocessing

Typically machine learning algorithms fits on all the data points all together, without feeding batches of data unlike other optimization function (Stochastic Gradient Descend, Neural Networks); multi-spectral images can be memory heavy, since normal RGB composite has 3 channels, a multispectral image from Sentinel-2 weights at least 4 times a classic images. Furthermore the training set has around 2 billion of pixels, that translates into 2 billion sample of 13 features.

To solve this problem, a **random sampling (noise)** was applied to the images of a percentage of the pixel as can be seen in Fig.4.2.

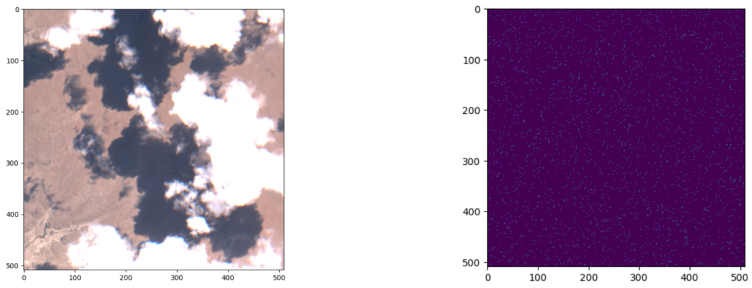


Figure 4.2: RGB image on the left, noise random sampling mask on the right (1% pixels)

The idea behind this method is to capture the general features of the image: this is possible, except for artificial objects like buildings, because environmental features like clouds or green areas comes often as big cluster of pixel. Furthermore

it is possible to reduce the dataset from billions to thousands of sample, without reduce the feature space. Finally a Z-Score normalization is applied on each pixel, turning TOA into values between -1 and 1.

$$z = \frac{x - \mu}{\sigma}$$

Where:

- μ is the mean value of the whole resampled dataset,
- σ is the standard-deviation value of the whole resampled dataset.

4.4.2 Model selection

The best model for each algorithm is selected through a K-Fold cross validation, using different techniques:

- Randomized Search
- Grid Search

The first one was used for Support Vector Machines, because of the higher training time it was impossible to test all the parameters, while for random forests a grid search was used. Moreover to prove how ML algorithms can generalize with fewer samples than other type of models, two different training support (number of pixel) were employed.

4.5 Lightweight Deep Learning model for cloud detection

The implementation of this neural network aims to demonstrate that a deep learning model is comparable with a physical algorithm in the task of cloud detection on board of satellites. The presented model is based on the U-Net architecture, consisting of an encoder, a decoder, and skip connections that link each downsampling module of the encoder to its corresponding one in the decoder . The main challenge is to reduce the number of parameters from millions of classical lightweight CNNs (EfficientNet, MobileNet [29] [17]) to thousands, trying to match the performance of classical algorithms.

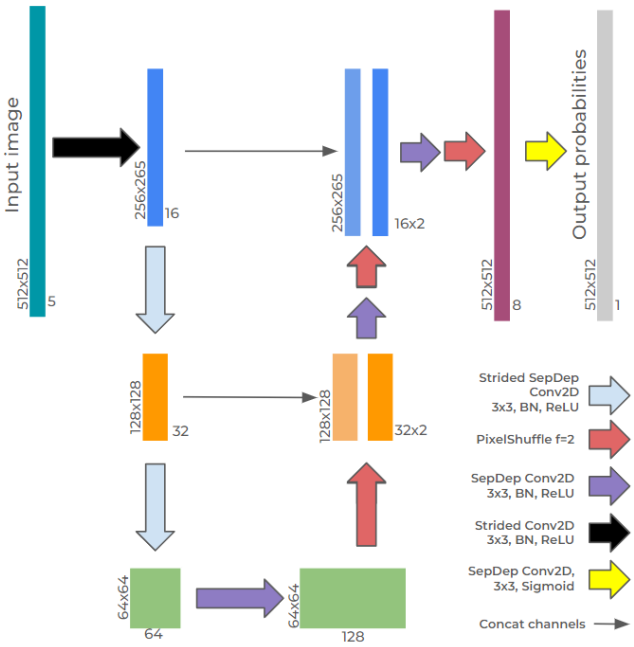


Figure 4.3: Light U-Net architecture

4.5.1 Encoder

The encoder consist in three blocks: one **stem layer** and two **downsampling blocks**. It differs from classical U-Net architectures because of the number of blocks and the number of convolutions in each blocks. The number of blocks was reduced from five to only three and for each block only one convolutional layer was employed. In Fig. 4.3 a representation of the model architecture can be seen.

Stem layer

The first layer (stem) serves to reduce the spacial dimension of the images and capture general features. Consists in a classical convolutional layer.

Downsampling block

Each downsampling block uses a **Separable Depthwise Convolution (SDC)** instead of classical Conv2D, to reduce the number of weights while decreasing the inference time. These SDC uses strided equal to 2 in order to halve the spatial dimension without using Pooling layers that would increase the computational time.

4.5.2 Decoder

The decoder consists of three upsampling step: the first block is preceded by a feature expansion step, while the last is followed by a final convolution and a **Sigmoid** activation to predict the per pixel label.

Feature depth expansion

Between the decoder and the encode a feature depth expansion block, composed by a SDC, is used as a bottleneck to increase the feature channels, considering the **PixelShuffle** upsampling block that will reduce the channel by the squared factor (from 128 to 32).

Upsampling block

Like classical U-Net, the upsampling block consists in concatenating channel-wise the feature extracted from the corresponding downsampling block, followed by a SDC to learn the features. To upsample each feature block, PixelShuffle is used instead of classical ConvTrasposed: because of the nature of strided convolution, deconvolution suffers from checkerbox artifacts [30].

4.5.3 Training

The model was trained using only the bands 2,3,4,8 and 10: three visibile infrared bands (4,3,2, RGB), one near infrared (8, I) and lastly one short wave-length band (10, C), namely cirrus band. Using only RGBIC instead of all 13 bands is something already seen in literature [28], resulting effective for multi-spectral images; Other than RGB bands, the infrared and cirrus bands were chosen because clouds have particular spectral response respectively for thick and thin clouds. Furthermore the loss used for the training phase is a **Binary Crossentropy**.

$$\text{Binary Crossentropy} = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log(p_i) + (1-y_i)\log(1-p_i)\right)$$

Throughout the neural network, Batch Normalization layers were used at every block before the activation function in an attempt to reduce the internal covariance-shift, while smoothing the gradient [31]. In fact the shallow nature of the designed architecture makes the model prone to underfitting and vanishing gradient.

Chapter 5

Experimental Results

In this chapter, the experiments conducted on the various algorithms will be illustrated, comparing the performance in terms of:

- True Positive Rate (TPR)
- False Positive Rate (FPR)
- Precision
- Balanced Overall Accuracy (BOA)
- Latency

The dataset used for all the experiments is cloudsen12, for more information see Sec. 4.2

5.1 Physical algorithms

The experiments on the physical algorithms were tested on the test set.

	TPR	**FPR**	**Precision**	**BOA**	**Latency**
Fmask	0.85	0.24	0.70	0.81	0.045
Sen2Cor	0.68	0.12	0.78	0.78	0.018
Sen2Cor C (int32)	0.67	0.12	0.78	0.78	0.030
Sen2Cor C v2 (int32)	0.69	0.12	0.78	0.78	0.030

Table 5.1

From Tab.5.1 and Tab.5.2 it seems that the median filter has little to no effect on the classification, but it can be seen that for all Sen2Cor versions the precision is

higher. Moreover for the C version algorithm, the number of sample with precision lower than 0.5 were around 130, but after applying the median filter only 98 masks had low precision: it is important to say that most of the images with "low precision" were also images with no positive labels, so clear sky images with snow, desert, soil or building, all objects with high reflectance in specific wavelength that can be confused as clouds (Fig. 5.1, Fig. 5.2). Lastly the reported latency for sen2cor and Fmask is measured using multi threading, exploiting the python numpy library, while for the C version is single thread.

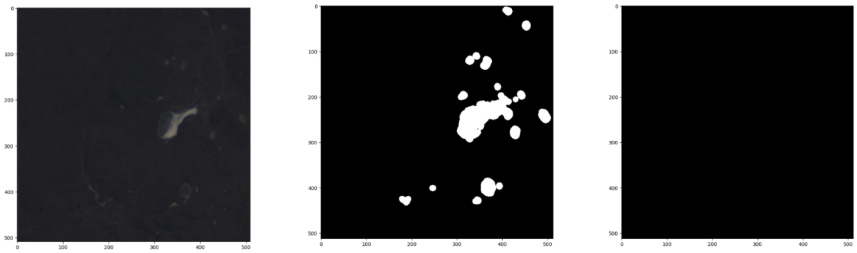


Figure 5.1: Misclassification with snow pixels. From left to right: RGB composite, model prediction, ground truth

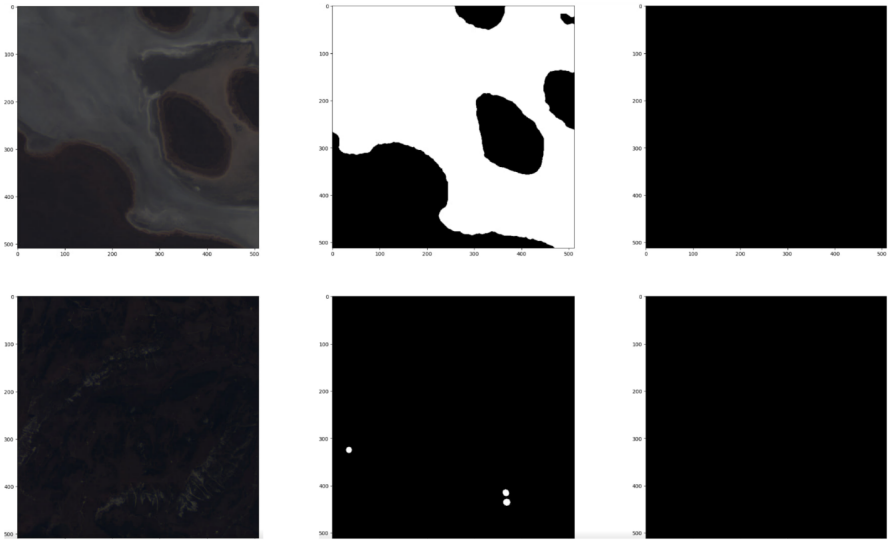


Figure 5.2: Misclassification with turbid water and snow pixel. From left to right: RGB composite, model prediction, ground truth

	BE (r)	MF	BD (r)	TPR	FPR	Precision	BOA	Latency
Fmask	-	3x3	-	0.85	0.24	0.70	0.81	0.054
Fmask	-	5x5	-	0.85	0.24	0.70	0.81	0.065
Sen2Cor	-	3x3	-	0.68	0.12	0.78	0.78	0.028
Sen2Cor	-	5x5	-	0.68	0.11	0.79	0.78	0.039
Sen2Cor C (int32)	-	3x3	-	0.67	0.11	0.79	0.78	N/A
Sen2Cor C (int32)	-	5x5	-	0.67	0.11	0.79	0.78	N/A
Sen2Cor C v2 (int32)	-	3x3	-	0.68	0.12	0.78	0.78	N/A
Sen2Cor C v2 (int32)	-	5x5	-	0.68	0.12	0.79	0.78	N/A

Table 5.2

Binary dilation filter dilates positive pixels, so it can be useful to dilate the cloud mask to compensate the effect of the median filter. However it can dilate also clump of false positive clouds. In Tab.5.3 it is possible to see how using a dilation filter after the median can cause the false positive rate to increase greatly and how it worsen all the other metrics. Overall it is not even worth since the latency raises.

	BE (r)	MF	BD (r)	TPR	FPR	Precision	BOA	Latency
Fmask		5x5	6	0.92	0.30	0.65	0.81	0.082
Sen2Cor		5x5	6	0.78	0.17	0.74	0.80	0.060
Sen2Cor C (int32)		5x5	6	0.78	0.17	0.73	0.80	N/A
Sen2Cor C v2 (int32)		5x5	6	0.79	0.18	0.73	0.80	N/A

Table 5.3

In Tab.5.4 instead of using the median filter, a binary erosion filter is used before dilation. It can be seen how, after the dilation, the precision is nearly preserved. The problem with binary erosion comes with the size of cloud: it is possible that small clumps of cloud could be completely deleted, so the accuracy could drop in some cases.

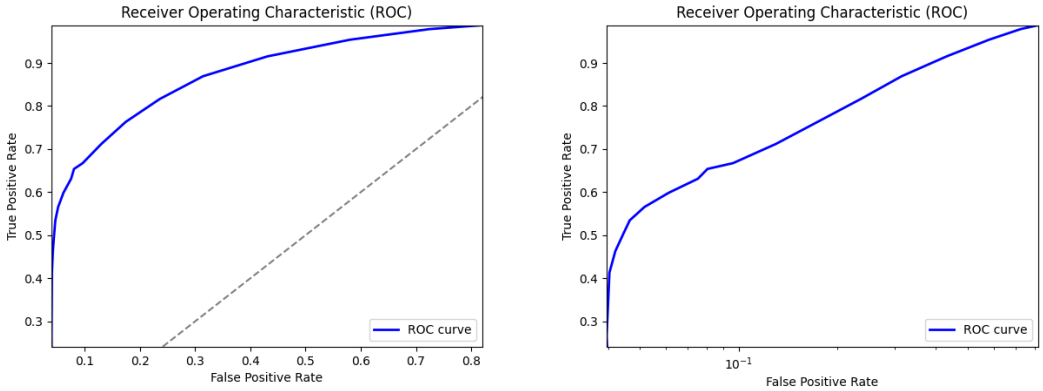
	BE (r)	MF	BD (r)	TPR	FPR	Precision	BOA	Latency
Fmask	3		6	0.88	0.25	0.69	0.81	0.066
Fmask	5		6	0.83	0.22	0.70	0.81	0.072
Sen2Cor	3		6	0.70	0.12	0.79	0.80	0.044
Sen2Cor	5		6	0.65	0.10	0.80	0.78	0.047

Table 5.4

Overall the algorithms benefits from the filters, but working at really high resolution (10m) it could be computational heavy if we want to use bigger filters (e.g. 9 radius dilation is a 18x18 matrix). Moreover in some cases (e.g. erosion) small clumps of cloud could be accidentally deleted, or small negative clumps (e.g.

median) could be dilated, increasing false positive pixels. Finally the algorithms could benefits from some false positive pixel retrieval for snow or other things like water: both algorithms already implements some of these techniques to an extent, but often it use auxiliary data.

Threshold optimization



(a) ROC curve **(b)** Simlog ROC curve

Figure 5.3: ROC cruve of shifting threshold. Each point is sampled for each percentage point

In Tab.5.5 it is possible to see how in some cases shifting threshold can compensate the erosion and dilation filters: for example shifting threshold by 20% can lead to similar results in using erosion and dilation (slowing the processing time) on the mask. Overall the last two rows are the best results, in terms of TPR an FPR, even if the last one seems to be similar to the 4th row the precision value is higher (88% vs 86%)

	Threshold Shift	Cirrus opt	BOA (%)	TPR (%)	FPR (%)	Precision (%)	Post processing filters
Sen2Cor	N	Y	0.79	0.65	0.08	0.84	N
Sen2Cor	Y (0.20)	Y	0.78	0.63	0.07	0.85	N
Sen2Cor	Y (0.30)	Y	0.78	0.61	0.06	0.87	N
Sen2Cor	N	Y	0.78	0.63	0.06	0.86	e(5) d(6)
Sen2Cor	Y(0.20)	Y	0.79	0.65	0.06	0.86	e(5) d(9)
Sen2Cor	Y(0.30)	Y	0.79	0.63	0.06	0.88	e(5) d(9)

Table 5.5: Results for optimized thresholds (shifting and cirrus)

Using the Genetic Algorithm, an optimization on dynamic threshold was done. The GA was configured with an initial population of 21 individual (default plus

20 generated individual), 5 offspring per generation, for 100 generation. Every generation an elitism of 20 individual was employed, keeping only the best 20 genomes. To compute the fitness function the following formula was used:

$$TPR - 4 * FPR$$

Where the metrics were evaluated on the test set, using sen2cor algorithm changing the thresholds with the genome. This fitness function is used to preserve the True Positive Rate while weighting the False Positive Rate in order to reduce it.

	Dynamic Thr	**BOA (%)**	**TPR (%)**	**FPR (%)**	**Precision (%)**
Sen2Cor (Cirrus Opt)	N	0.79	0.65	0.08	0.84
Sen2Cor (TS 0.20 + CO)	N	0.78	0.63	0.07	0.85
Sen2Cor (TS 0.20 + CO)	Y	0.78	0.62	0.06	0.87
Sen2Cor (TS 0.30+ CO)	N	0.78	0.61	0.06	0.87
Sen2Cor (TS 0.30 + CO)	Y	0.77	0.60	0.05	0.88

Table 5.6: Metric changes adding dynamic threshold optimization.

In Tab. 5.6 is possible to see how changing the thresholds with the best genome can slightly reduce FPR without having impactful TPR loss. Surely the result using a 20% threshold shifting (TF 0.20) and cirrus optimization (CO) could be interesting, but increasing the shifting percentage can worsen the result.

5.2 Machine learning algorithms

Classical algorithms

As for the previous experiment, cloudsen12 was used to train and test the models. Since the training set has around 2 billion of pixels, a random sampling was applied to the images of a percentage of the pixel (depending on the algorithm, default 1% of pixel for each image. Fig.4.2). Taking the resampled dataset, the pixels were shuffled and 10% of the total were used for training and another 10% of pixels were used for validation, but in some cases the training support was further reduced because of slow training time and to test the performance on limited data. For the feature space, all 13 bands were used, having for each sample 13 feature, representing the different spectral response at different wave length. Finally each pixel value was normalized with standardization (Z-score).

	Training support (pixels)	BOA (%)	TPR (%)	FPR (%)	Latency (s)
RF-1 (100 estimator)	∼850000	0.87	0.81	0.05	0.5
RF-2 (100 estimator)	∼85000	0.86	0.81	0.08	0.36
SVM-RBF-1	∼1000	0.81	0.76	0.14	8
SVM-RBF-2	∼85000	0.83	0.69	0.03	384
s2cloudless	∼14000000	0.85	0.78	0.08	11

Table 5.7: Machine learning algorithms results

In Tab.5.7 can be seen how even if SVM is trained with a few pixel, it can be really accurate and surpass most of the physic based algorithm; but the problem comes when we have to compare the latency. The best random forest model is better than Sen2Cor in terms of FPR, but is 25 times slower.

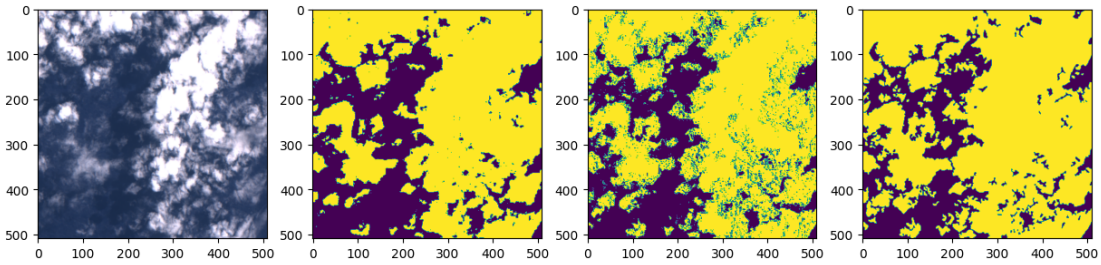
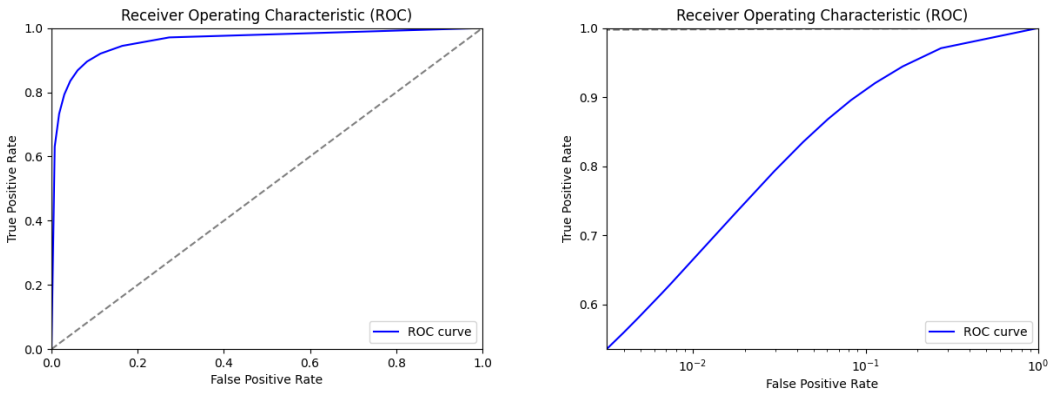


Figure 5.4: From left to right: RGB composite, SVM-RBF-2, RF-1, ground truth

Overall even if the sampling strategy seems effective to train classic ML algorithm, the processing time is considerably high.

Light Weight Model

To train the model, the images pixel were normalized as TOA (Top-of-Atmosphere), the images were padded from 509x509 to 512x512. The training was made on a GPU Quadro P6000, with a batch size of 16 for 50 epochs using Adam optimizer with an initial learning rate of 0.001, decreased to 0.0001 after 30 epochs, on around 9000 images of the training set. Likewise for the others algorithms and model, the test was made on the test set, comprehensive of 975 images.



(a) ROC curve **(b)** Simlog ROC curve

Figure 5.5: ROC curve of the lightweight model, changing CPT from 0 to 1

During the test the Cloud Probability Threshold (CPT), applied to the sigmod, was tweaked from 0 to 1, in Fig.5.5 it is possible to see how the TPR and FPR changes, and how the model is robust to TPR loss, even with high thresholds (0.80) the net drops from 87% to 73% TPR while from 6% FPR to only 2%.

	Model parameters	CPT (%)	BOA (%)	TPR (%)	FPR (%)	Precision (%)	Latency (ms)
Lightweight model	19769	0.50	0.90	0.87	0.06	0.90	20
Lightweight model	19769	0.60	0.90	0.84	0.04	0.92	20
Lightweight model	19769	0.70	0.88	0.79	0.03	0.94	20
Lightweight model	19769	0.80	0.86	0.73	0.02	0.96	20
Lightweight model	19769	0.90	0.81	0.63	0.01	0.98	20
Lightweight model	19769	0.92	0.79	0.58	0.00	0.99	20

Table 5.8: Results for different Cloud Probability Threshold (CPT)

In Tab.5.8 is it possible to see how the model, tuning the threshold, can be precise, keeping a good accuracy and a low FPR. Furthermore, the model runs on CPU at 20-23 ms,

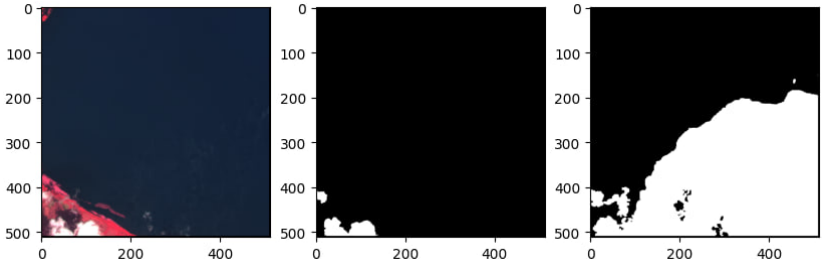


Figure 5.6: From left to right: RGB composite, model prediction, ground truth

5.3 Comparison

In Tab. 5.9 it is possible to see how the lightweight model outperform sen2cor in all the metrics, with only being a couple milliseconds slower than sen2cor with only threshold optimization while being 2 times faster when postprocessing is used. Lastly in Fig. 5.6 it is shown how the model find difficult to classify thinner clouds.

	Model parameters	CPT (%)	BOA (%)	TPR (%)	FPR (%)	Precision (%)	Latency (ms)
sen2cor (THR 0.30 + CO)	N/A	0.20	0.78	0.61	0.06	0.87	18
sen2cor (THR 0.30 CO + e(5)+d(9)	N/A	0.20	0.79	0.63	0.06	0.88	50
Lightweight model	19769	0.80	0.86	0.73	0.02	0.96	20
Lightweight model	19769	0.90	0.81	0.63	0.01	0.98	20

Table 5.9: Comparison of the best performing algorithms

In Fig. 5.7 it can be seen the difference in how the algorithms perform changing the thresholds. It is worth to note that no optimizations were made on the model other than using SeparableDepthWise convolutions. Moreover it is possible to further optimize the model exploiting weight pruning and quantization. For a last note, the model run at 3 ms on average when running on a GPU Quadro P6000 (5x faster than sen2cor).

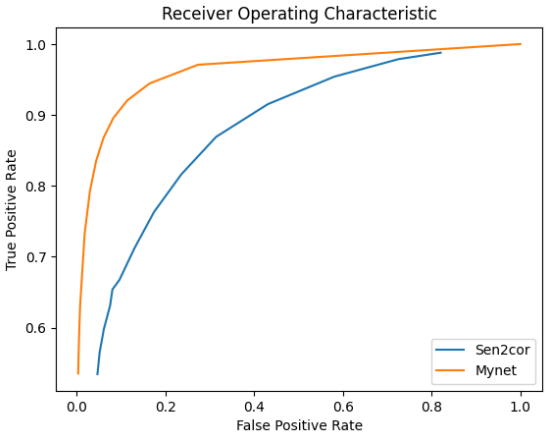


Figure 5.7: ROC of Sen2Cor shifting threshold with cirrus opt vs ROC of Lightweight model

5.3.1 Thick vs Thin clouds

Since thin clouds are harder to be detected due to their spectral feature similarities with other objects, the two best algorithm were evaluated on the single labels (thin and thick clouds). To obtain the prediction for thick and thin clouds, taking the true pixel from the thin clouds label and the thick cloud label, it was employed

a subtraction of the true positive think cloud from the prediction mask and vice versa for the thin mask, as can be seen in Fig. 5.8.

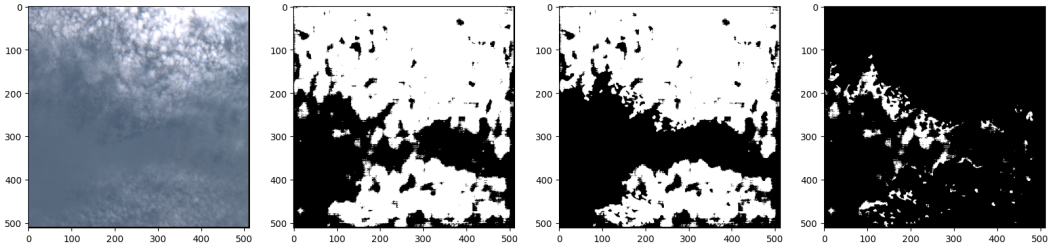


Figure 5.8: From left to right: RGB composite, segmentation mask, thick cloud prediction and finally thin cloud prediction

	CPT (%)	**BOA (%)**	**TPR (%)**	**FPR (%)**	**Precision (%)**
Sen2Cor (Cirrus Opt)	0.20	0.81	0.69	0.07	0.80
Sen2Cor (THR 0.30 + CO)	0.20	0.80	0.65	0.05	0.84
Sen2Cor (THR 0.30 + CO + e(5) d(9))	0.20	0.81	0.67	0.05	0.85
Lightweight model	0.50	0.94	0.93	0.05	0.88
Lightweight model	0.80	0.91	0.84	0.02	0.96
Lightweight model	0.90	0.87	0.76	0.01	0.98

Table 5.10: Thick clouds statistics

In Tab. 5.10 it can be seen how the Lightweight model outperforms the physical algorithm in all the metrics, without even scaling the CPT. However the optimization used for sen2cor makes it robust to cirrus detection; in Tab. 5.11 the deep learning model suffer from considerable TPR loss, probably due to the shallow architecture that make it hard to learn cirrus features. Moreover, as in Fig. 5.6 it is possible that cirrus labeling in this dataset can increase the difficulty in learning spacial features for thin clouds.

	CPT (%)	**BOA (%)**	**TPR (%)**	**FPR (%)**	**Precision (%)**
Sen2Cor (Cirrus Opt)	0.20	0.73	0.52	0.05	0.49
Sen2Cor (THR 0.30 + CO)	0.20	0.72	0.48	0.04	0.55
Sen2Cor (THR 0.30 + CO + e(5) d(9))	0.20	0.73	0.49	0.04	0.57
Lightweight model	0.50	0.81	0.65	0.04	0.61
Lightweight model	0.80	0.69	0.39	0.01	0.76
Lightweight model	0.90	0.61	0.23	0.01	0.81

Table 5.11: Thin clouds statistics

Chapter 6

Conclusion

In summary, the goal of this work was to understand how existing algorithms and architecture can be optimized and used for cloud screening, reducing misclassification on false positive pixel while keeping good precision and low latency. Starting from the results on physical algorithm optimization, these allowed us to understand how can be complex to reduce FPR only by changing the threshold, while in the complete Sen2Cor preprocessor there are other post processing steps that uses ancillary data. Moreover ML algorithms, even if more accurate than classical algorithms, have been proved to be expensive in terms of performance. Despite the promising performance of the deep learning model, the results could be improved: as a matter of fact the research community focus is shifting more and more towards the development of robust deep learning models and implementation on Field Programmable Gate Array (FPGA) and other hardware accelerator. Furthermore a physical algorithm is easier to implement and need less expensive hardware to run, making it more attractive to engineers, but neural networks are closing the gap: explainability and optimization are becoming the standard in deep learning model design. Currently, this work shows how a neural network can be competitive, both in terms of performance and latency, with a classical algorithm, especially how it is possible to minimize (up to 1%) the FPR without sacrificing segmentation performance. Surely, building on this work, one could consider quantizing the weights to 8-bit integers to reduce latency down to four times, also the use of hardware accelerators could further speed up the model; finally being an ultra-light model (around 20k weights) it could be easier to implement redundancy to withstand hardware faults.

Bibliography

[1] URL: https://sentiwiki.copernicus.eu/web/s2-mission (cit. on p. 4).

[2] Curtis E. Woodcock Zhe Zhu. «Object-based cloud and cloud shadow detection in Landsat imagery». In: *Remote Sensing of Environment* (2012) (cit. on p. 5).

[3] Curtis E. Woodcock Zhe Zhu Shixiong Wang. «Improvement and expansion of the Fmask algorithm: cloud, cloud shadow, and snow detection for Landsats 4–7, 8, and Sentinel 2 images». In: *Remote Sensing of Environment* (2015) (cit. on pp. 5, 8).

[4] Shi Qiu, Zhe Zhu, and Binbin He. «Fmask 4.0: Improved cloud and cloud shadow detection in Landsats 4–8 and Sentinel-2 imagery». In: *Remote Sensing of Environment* 231 (2019), p. 111205. ISSN: 0034-4257. DOI: https://doi.org/10.1016/j.rse.2019.05.024. URL: https://www.sciencedirect.com/science/article/pii/S0034425719302172 (cit. on pp. 5, 8, 11).

[5] Magdalena Main-Knorn, Bringfried Pflug, Jerome Louis, Vincent Debaecker, Uwe Müller-Wilm, and Ferran Gascon. «Sen2Cor for Sentinel-2». In: Oct. 2017, p. 3. DOI: 10.1117/12.2278218 (cit. on p. 5).

[6] Prasad S. Thenkabail and Zhuoting Wu. «An Automated Cropland Classification Algorithm (ACCA) for Tajikistan by Combining Landsat, MODIS, and Secondary Data». In: *Remote Sensing* 4.10 (2012), pp. 2890–2918. ISSN: 2072-4292. DOI: 10.3390/rs4102890. URL: https://www.mdpi.com/2072-4292/4/10/2890 (cit. on pp. 8, 9).

[7] Luis Gómez-Chova, Gustavo Camps-Valls, Javier Calpe, Luis Guanter, and Jose Moreno. «Cloud-screening algorithm for ENVISAT/MERIS multispectral images». In: *Geoscience and Remote Sensing, IEEE Transactions on* 45 (Jan. 2008), pp. 4105–4118. DOI: 10.1109/TGRS.2007.905312 (cit. on p. 9).

[8] Y Zhang, B Guindon, and J Cihlar. «An image transform to characterize and compensate for spatial variations in thin cloud contamination of Landsat images». In: *Remote Sensing of Environment* 82.2 (2002), pp. 173–187. ISSN: 0034-4257. DOI: https://doi.org/10.1016/S0034-4257(02)00034-2. URL:

https://www.sciencedirect.com/science/article/pii/S0034425702000342 (cit. on p. 9).

[9] *Sentinel 2 MPC Team, Level-2A Algorithm Theoretical Basis Document*. Nov. 2021. URL: https://sentiwiki.copernicus.eu/web/document-library#DocumentLibrary-AlgorithmTechnicalBaselineDocumentationLibrary-S2-ATBD (cit. on pp. 12–15).

[10] Subhadip Mitra. *Evolutionary Bytes - Harnessing Genetic Algorithms for Smarter Data Platforms (Part 1/2)*. URL: https://subhadipmitra.com/blog/2023/genetic-algorithm-inspired-data-platforms-part-1/ (cit. on p. 17).

[11] Corinna Cortes and Vladimir Vapnik. «Support-vector networks». In: *Machine learning* 20.3 (1995), pp. 273–297 (cit. on p. 19).

[12] Antje Kirchner and Curtis S. Signorino. «Using Support Vector Machines for Survey Research». In: *RTI International. P.O. Box 12194, Research Triangle Park, NC 27709-2194. Tel: 919-541-6000; e-mail: publications@rit.org; Web site: http://www.rti.org* (). URL: https://doi.org/10.29115/SP-2018-0001 (cit. on p. 19).

[13] Yanran Wang, Jonghyuk Baek, Yichun Tang, Jing Du, Mike Hillman, and Jiun-Shyan Chen. «Support vector machine guided reproducing kernel particle method for image-based modeling of microstructures». In: (Apr. 2024). URL: https://doi.org/10.1007/s00466-023-02394-9 (cit. on p. 20).

[14] L. Breiman, Jerome H. Friedman, Richard A. Olshen, and C. J. Stone. «Classification and Regression Trees». In: *Biometrics* 40 (1984), p. 874 (cit. on p. 20).

[15] André Hollstein, Karl Segl, Luis Guanter, Maximilian Brell, and Marta Enesco. «Ready-to-Use Methods for the Detection of Clouds, Cirrus, Snow, Shadow, Water and Clear Sky Pixels in Sentinel-2 MSI Images». In: *Remote Sensing* 8.8 (2016). ISSN: 2072-4292. DOI: 10.3390/rs8080666. URL: https://www.mdpi.com/2072-4292/8/8/666 (cit. on pp. 20, 25, 26, 33).

[16] Leo Breiman. «Random Forests». In: *Mach. Learn.* 45.1 (Oct. 2001), pp. 5–32. ISSN: 0885-6125. DOI: 10.1023/A:1010933404324. URL: https://doi.org/10.1023/A:1010933404324 (cit. on p. 21).

[17] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. arXiv: 1704.04861 [cs.CV]. URL: https://arxiv.org/abs/1704.04861 (cit. on pp. 23, 34).

[18] Furkat Sultonov, Jun-Hyun Park, Sangseok Yun, Dong-Woo Lim, and Jae-Mo Kang. «Mixer U-Net: An Improved Automatic Road Extraction from UAV Imagery». In: *Applied Sciences* 12 (Feb. 2022), p. 1953. DOI: 10.3390/app12041953 (cit. on p. 23).

[19] Siddhesh Bangar. *VGG-Net Architecture Explained*. June 2022. URL: https://medium.com/@siddheshb008/vgg-net-architecture-explained-71179310050f (cit. on p. 24).

[20] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV] (cit. on p. 24).

[21] Conor O'Sullivan. *U-Net Explained: Understanding its Image Segmentation Architecture*. Mar. 2023. URL: https://towardsdatascience.com/u-net-explained-understanding-its-image-segmentation-architecture-56e4842e313a (cit. on p. 24).

[22] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. *Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network*. 2016. arXiv: 1609.05158 [cs.CV]. URL: https://arxiv.org/abs/1609.05158 (cit. on p. 25).

[23] Sergii Skakun et al. «Cloud Mask Intercomparison eXercise (CMIX): An evaluation of cloud masking algorithms for Landsat 8 and Sentinel-2». In: *Remote Sensing of Environment* 274 (2022), p. 112990. ISSN: 0034-4257. DOI: https://doi.org/10.1016/j.rse.2022.112990. URL: https://www.sciencedirect.com/science/article/pii/S0034425722001043 (cit. on pp. 26, 33).

[24] O. Hagolle, G. Dedieu, B. Mougenot, V. Debaecker, B. Duchemin, and A. Meygret. «Correction of aerosol effects on multi-temporal images acquired with constant viewing angles: Application to Formosat-2 images». In: *Remote Sensing of Environment* 112.4 (2008). Remote Sensing Data Assimilation Special Issue, pp. 1689–1701. ISSN: 0034-4257. DOI: https://doi.org/10.1016/j.rse.2007.08.016. URL: https://www.sciencedirect.com/science/article/pii/S0034425707004130 (cit. on p. 26).

[25] O. Hagolle, M. Huc, D. Villa Pascual, and G. Dedieu. «A multi-temporal method for cloud detection, applied to FORMOSAT-2, VENμS, LANDSAT and SENTINEL-2 images». In: *Remote Sensing of Environment* 114.8 (2010), pp. 1747–1755. ISSN: 0034-4257. DOI: https://doi.org/10.1016/j.rse.2010.03.002. URL: https://www.sciencedirect.com/science/article/pii/S0034425710000908 (cit. on p. 26).

[26] Marharyta Domnich et al. «KappaMask: AI-Based Cloudmask Processor for Sentinel-2». In: *Remote Sensing* 13.20 (2021). ISSN: 2072-4292. DOI: 10.3390/rs13204100. URL: https://www.mdpi.com/2072-4292/13/20/4100 (cit. on p. 26).

[27] Cesar Aybar et al. «Cloudsen12, a global dataset for semantic understanding of Cloud and cloud shadow in sentinel-2». In: *Scientific Data* 9.1 (Dec. 2022). DOI: 10.1038/s41597-022-01878-2 (cit. on pp. 26–28).

[28] Luis Gómez-Chova Dan López-Puigdollers Gonzalo Mateo-García. «Benchmarking Deep Learning Models for Cloud Detection in Landsat-8 and Sentinel-2 Images». In: *Remote Sensing* (2021) (cit. on pp. 26, 36).

[29] Mingxing Tan and Quoc V. Le. *EfficientNetV2: Smaller Models and Faster Training*. 2021. arXiv: 2104.00298 [cs.CV]. URL: https://arxiv.org/abs/2104.00298 (cit. on p. 34).

[30] Augustus Odena, Vincent Dumoulin, and Chris Olah. «Deconvolution and Checkerboard Artifacts». In: *Distill* (2016). DOI: 10.23915/distill.00003. URL: http://distill.pub/2016/deconv-checkerboard (cit. on p. 36).

[31] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. *How Does Batch Normalization Help Optimization?* 2019. arXiv: 1805.11604 [stat.ML]. URL: https://arxiv.org/abs/1805.11604 (cit. on p. 36).