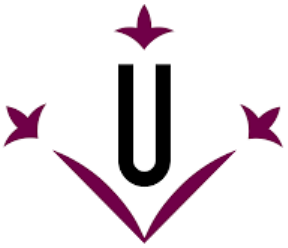


Universitat de Lleida

Generative deep learning based models for cloud removal satellite imagery

Made by
Oriol Alàs Cercós

Delivery
25th of May, 2022

Universitat de Lleida

Escola Politècnica Superior

Màster en Enginyeria Informàtica

Technological Business Management and Entrepreneurship

Professorate:

Josep Escribà Garriga

Contents

I Introduction **1**

1 Problem Analysis **4**

1.1 Sentinel2 4

1.2 Related work and state-of-the-art 7

1.2.1 Deep Learning & CNN 7

1.2.2 Generative Adversarial Networks 10

1.2.3 Other generative models 20

1.2.4 Transformers in computer vision 29

1.3 Decision on thesis alignment 36

1.4 Academic Datasets 36

II Development **39**

2 Exploratory Data Analysis **39**

2.1 Properties extraction and visualization of the first dataset 40

2.2 Data merge strategy 53

3 Model training **56**

3.1 Technologies 56

3.2 Losses and metrics 58

3.3 Models architecture 62

3.3.1 Convolutional Neural Networks 63

3.3.2 Autoencoders 65

3.3.3 Generative Adversarial Networks 67

3.3.4 Denoising Diffusion Models 68

4 Experiments and results **69**

4.1 Challenges with data normalization 69

4.2 Results 70

4.3 Experiments 76

5 Future directions **80**

III Conclusions **82**

Appendices **87**

A Exploratory Data Analysis **87**

List of Figures

1 Project timeline . . . 3
2 LeNet-5 architecture. . . . 8
3 Residual learning: a building block. . . . 8
4 Loss surfaces of ResNet-56 with/without skip connections. . . . 9
5 DSen2-CR model diagram. . . . 9
6 Examples of the influence of SAR data. Being (a, f) cloudy images, (b, g), SAR images, (c, h), the target cloudless images, (d, i), the synthesized images without SAR input data and (e, j) the predictions using SAR data as input. . . . 10
7 GAN model diagram. . . . 11
8 cGAN model diagram. . . . 12
9 Prediction results by McGAN with the synthesized cloud images . . . 13
10 Prediction results by McGAN with the real cloud images . . . 14
11 Comparison between the cloudy images, the outputs of McGAN and Pix2Pix-based models and the ground-truth image. . . . 15
12 Comparison between cloudy images, the outputs of McGAN versions and the ground-truth image. . . . 16
13 Failure cases in CloudGAN attempting to removal overly clouded images over-smoothening or completely failing. . . . 18
14 Failure cases in CloudGAN attempting to removal overly clouded images over-smoothening or completely failing. . . . 19
15 Generator $G_{\text{cloudy}\rightarrow\text{cloudfree}}$ of [1] . . . 19
16 Generator $G_{\text{cloudy}\rightarrow\text{cloudfree}}$ of [2] . . . 20
17 Generative Learning Trilemma . . . 21
18 Autoencoder architecture. . . . 22
19 MSE loss in autoencoders . . . 22
20 Autoencoder architecture. . . . 23
21 Comparison between non-probabilistic autoencoder and VAE. . . . 25
22 Forward diffusion process mapping . . . 26
23 Forward diffusion process . . . 26
24 Forward and diverse diffusion processes . . . 27

25 Transformer block architecture. **left block**, the encoder architecture, **right block**, the decoder architecture. 30
26 Multi-Head Attention block 31
27 Scaled Dot Product Attention Head 32
28 ViT architecture 34
29 L-TAE architecture 35
30 [3] model generator architecture 35
31 Training-validation-testing split sets in SEN12MS-CR. 40
32 Laplacian variance depending on the slope used in sigmoid scaling method. 42
33 Scaling methods 42
34 Sigmoid-scaled images with their RGB bands in different slopes. 43
35 Distribution of the dataset regarding the cloud cover percentage area 46
36 Sample of images labeled as cloudy but with low cloud cover percentage area. 47
37 Scatter plot of the laplacian variance in B2 and the saturation of the true color image. 48
38 Scatter plot of the temperature and the saturation of the true color image. 49
39 Scatter plot of the laplacian variance in B10 and the saturation of the true color image. 49
40 Bar plot of the mean value in each band of the laplacian variance. 50
41 Correlation of the region with clouds and the values of the band, by mean from each image. 51
42 Kernel density estimation of B10 in cloudy and cloudless the patches of a specific scene (130) 52
43 Cloudy images percentage in SEN12MS-CR-TS. 53
44 Patch exploration in all intervals. 53
45 Cloud percentage distribution in all intervals and its possible classification given the median. 54
46 Pairing algorithm. 54
47 Comparison of all the metrics ($w = 1$). 61
48 Carla model 63
49 Residual block from `Regina` 64
50 Regina architecture 65
51 Anna architecture 66

52 AnnaSkip architecture 67
53 Loss of the Carla best run. 71
54 Comparison of test PSNR with L1 (red) and MSE (orange) of AusiasConv, normalized data. 72
55 Stabilized training 74
56 Comparison of MSE between RegiGAN and Regina. 74
57 Sample of cloudy (left) and cloudless (right) images along with `Regina`'s synthetic. 77
58 Sample of cloudy (left) and cloudless (right) images along with `Ausias`'s synthetic. 78
59 Sample of cloudy (left) and cloudless (right) images along with `RegiGAN`'s synthetic. 79
60 12 images of the sample showing the diversity of cloudy images. 88
61 Unclassified images 89
62 Sample of images with high laplacian variance, blurriness. 90
63 Sample of images labeled as cloudy but with low cloud cover percentage area. 91
64 Sample of images labeled as cloudy but with low cloud cover percentage area. 92
65 Correlation of the image and the values of the band, by mean from each image. 93
66 Comparison between B1 and B10 regarding blurriness and contrast. 94
67 Kernel density estimation of B3 in cloudy and cloudless the patches of a specific scene (100) 95
68 Patch exploration in all intervals. 96
69 Cloud percentage and band mean value in all time instances (1). 97
70 Cloud percentage and band mean value in all time instances (2). 98
71 Cloud percentage and index mean value in all time instances (1). 99
72 Cloud percentage and index mean value in all time instances (2). 100

List of Tables

1 Sentinel 2 bands 5
2 PSNR and SSIM of the single-image and multi-temporal model versions . . 16
3 Complexity per layer and number of sequential operations for different type layers. n is the sequence length, d is the representation dimension and k is the kernel size of convolutions. 33
4 Academic datasets and its main features 38
5 Specifications of the machine 58
6 Comparison of all the metrics. 62
7 Comparison of metrics between `Carla` and original `Regina` (with data normalization) 71
8 Comparison of metrics between variants of `Regina` (with normalized data). 71
9 Comparison of metrics of the autoencoders (without normalizing data). . . 72
10 Comparison of loss functions in Ausias model (normalized data). 73
11 Hyper parameters of RegiGAN 73
12 Comparison of loss functions between RegiGAN and Regina. 74
13 Comparison of the scores of the models 75
14 Comparison of the scores of the models 76
15 Cost of the project 81
16 Bandwidths translation between Sentinel2 and the datasets [1] & [2]. 87

Acronyms

CARL Cloud-Adaptive Regularized Loss.

cGAN Conditional Generative Adversarial Network.

CNN Convolutional Neural Network.

DDM Denoising Diffusion Model.

ELBO Evidence Lower Bound.

GAN Generative Adversarial Network.

IR Infra Red.

L-TAE Lightweight Temporal Attention Encoder.

L1C Level-1C.

L2A Level-2A.

MAE Mean Absolute Error.

McGAN Multi-spectral Conditional Generative Adversarial Network.

MLP Multi-Layer Perceptron.

MSE Mean Squared Error.

NIR Near Infra Red.

NLP Natural Language Processing.

PSNR Peak signal-to-noise ratio.

RNN Recurrent Neural Network.

RS Remote Sensing.

SAM Spectral Angle Mapper.

SAR Synthetic Aperture Radar.

SSIM Structural Similarity Index Metric.

TAE Temporal Attention Encoder.

VAE Variational AutoEncoder.

ViT Vision Transformer.

Part I

Introduction

Remote Sensing (RS) imagery is critical to overcome challenges such climate change or natural resources management, including zone monitoring for reforestation, disaster mapping, evolution, and impact assessment, land surface change detection and coastal water quality monitoring. Nevertheless, on average 55% of the Earth's land surfaces is covered by clouds, being then a significant impediment to carry out a broad range of applications. Satellite imagery plagued by films of clouds that obstructs the scene implies a great loss of information or causing effects such as blurring, which mitigates the power of RS. Hence, RS applications would be greatly improved with techniques aimed at detecting and removing cloudy regions, and proviing in-painting capabilities for the underlying scene, thus allowing for better analytics and decission support.

On the other hand, my interest in deep learning and computer vision has truly flourished over the past two years. During my initial project in computer vision focused on satellite images, I suffered hard times because of cloud coverage. Regrettably, many ideas couldn't be realized without access to a dependable weekly record that would meet the end user's reliability requirements. This experience has piqued my curiosity for tackling new challenges and, most importantly, learning from them to pave the way for future endeavors.

Furthermore, deep learning has, in recent years, unveiled opportunities that were entirely unimaginable just a decade or two ago. Its immense potential, especially in generative models, holds the promise of substantial societal and environmental advancements. However, grasping the intricacies of these models and determining when to employ one over the other can prove rather challenging without a dedicated period of learning and experimentation.

In this ever evolving landscape, the intersection of RS and deep learning holds the promise of advancing our understanding of the world and our ability to tackle complex global issues. So why not embark on this journey?

Objectives

Once set the challenge and the motivation, the goals of this master thesis are:

- Search state-of-the-art approaches as well as frameworks that could bring new improvements and better results.
- Create or search for a multi-temporal and spatial imagery to design and evaluate a model that can remove the clouds given a satellite image.
- Design and implement an effective model and monitoring its training.
- Evaluate the model and provide a benchmark with other approaches.

Task planification

Creating a project plan and establishing timelines for each task are essential steps in effectively managing your deep learning project. To ensure efficient project management and maintain a consistent work pace, the project has been broken down into three main tasks, each further subdivided into smaller components. These tasks include:

- **Study of the problem and review of the State of the Art**. This task involves conducting a comprehensive analysis of the problem the project aims to solve. It not only includes a detailed examination of existing research and methodologies relevant to the problem but also comprehends the technical requirements to do the following tasks.

- **Data exploration**. Data exploration entails the thorough investigation of the dataset or datasets to be used in the project. This process includes data collection, cleaning, preprocessing, and augmentation as necessary. The goal is to understand the data's characteristics

- **Training and generation of sample results**. The training and generation of sample results constitute the practical implementation phase of the project. This step also involves hyperparameter tuning and iterative model improvements. Ultimately, the goal is to generate sample results that demonstrate the model's ability to remove clouds from satellite images, showcasing the effectiveness of the developed algorithms.

Figure 1: Project timeline

	Month 1		Month 2		Month 3		Month 4		Month 5	
1. SoA										
Problem Understanding										
Literature Review										
Deep Learning Review										
2. Data Exploration										
Data Collection										
Data Processing										
Data Analysis										
Data Merge										
3. Training										
Training development										
Model development										
Hyperparameter tuning										
Evaluation and sample results										

1 Problem Analysis

RS is a scientific method used to obtain information about objects or areas from a distance, typically from aircraft or satellites. This technique involves detecting and measuring the radiation that is emitted or reflected by these objects or areas. Remote sensing is widely used in various fields such as meteorology, agriculture, geology, and environmental science, among others.

Traditional ground-based data collection methods can be resource-intensive, requiring transportation, equipment, and personnel. In this way, Recurrent Neural Network (RNN) minimizes the need for these, conserving resources. Satellite-based observations reduce the need for on-ground surveys, which can involve vehicles and equipment that emit greenhouse gases. When cloud interference is removed, satellite imagery provides a more comprehensive view of large areas, ensuring that no critical changes or patterns are missed. Moreover, the timely access to clear images can be crucial for monitoring and responding to environmental changes or disasters.

Although there are several constellations and a high diversity of aerial and satellite images, the problem has been focused using Copernicus Sentinel2 constellation. The reason why is simple, Sentinel constellation imagery is freely available to the public, making it accessible for researchers, scientists, and organizations around the world. Definitely, open data policy encourages innovation and the development of a wide range of applications.

1.1 Sentinel2

Sentinel-2 is a valuable tool in remote sensing and Earth observation. Also, it is a valuable resource for a wide range of applications, thanks to its high-resolution, multi-spectral capabilities, global coverage, and open data policy. It plays a vital role in understanding and addressing environmental, agricultural, and land management challenges on a global scale.

Sentinel2 images are provided by two satellites, Sentinel 2A and Sentinel 2B, which orbit each other with a 180^{o} phase shift. Generally, the acquisition of the images needs around 10 days per satellite so that a new updated image of a specific area is available in

periods of time not exceeding five days. This makes Sentinel-2 data an excellent choice for studying environmental challenges. Sentinel-2 products are a compilation of elementary granules of fixed size, within a single orbit. A granule, also called tile, is a multi-spectral image with 13 bands in the visible, near-infrared, and short-wave infrared spectrum. These bands come in a different spatial resolution ranging from 10m to 60m, so the images can be classified as medium-high resolution. All the granules are 100x100km2 ortho-images in UTM/WGS84 projection. There are five types of Sentinel-2 data although only two are available for users: Level-1C (L1C) and Level-2A (L2A). The difference between them is that the latter provides background reflectance imagery of the atmosphere derived from associated L1C products. In 1, there are the bands with its most high resolution and ordered by its central wavelength, being B5-B12 Near Infra Red (NIR) or Infra Red (IR) bands.

Table 1: Sentinel 2 bands

Band	Name	Central wavelength (μm)	Bandwidth (nm)	Spatial resolution (m)
B1	Coastal aerosol	0.433	27	60
B2	Blue	0.490	98	10
B3	Green	0.560	45	10
B4	Red	0.665	38	10
B5	Vegetation Red Edge	0.705	19	20
B6	Vegetation Red Edge	0.740	18	20
B7	Vegetation Red Edge	0.783	28	20
B8	NIR	0.842	125	10
B8A	Vegetation Red Edge	0.865	33	20
B9	Water Vapour	0.945	26	60
B10	SWIR-Cirrus	1.375	75	60
B11	SWIR	1.610	143	20
B12	SWIR	2.190	242	20

The bands can give us a various range of aspects of the orthogonal view. Several spectral indices have been created over the years by performing operations between the bands, which can broaden the range of analysis and make them more accurate to finally better understand the features in the imagery. The most used bands to generate indices are B3, B4 and B8. The following list explains some of the most popular:

NDVI The Normalized Difference Vegetation Index is highly associated with the vegetation content. Higher values of NDVI correspond to areas that reflect more in the near-

infrared spectrum and to denser and healthier vegetation.

$$\text{NDVI} = \frac{\text{B8} - \text{B4}}{\text{B8} + \text{B4}}$$

GNDVI Green Normalized Difference Vegetation Index is modified version NDVI to be more sensitive to the variation of chlorophyll content in the crop.

$$\text{GNDVI} = \frac{\text{B8} - \text{B3}}{\text{B8} + \text{B3}}$$

NDMI Normalized Difference Moisture Index is used to determine vegetation water content.

$$\text{NDMI} = \frac{\text{B8} - \text{B11}}{\text{B8} + \text{B11}}$$

MSI Moisture Stress Index increases in leaf water content, so that makes it perfect for finding water stress in plants.

$$\text{MSI} = \frac{\text{B11}}{\text{B8}}$$

NBRI Normalized Burned Ratio Index detects burned areas and it is used to monitor the recovery of the ecosystem.

$$\text{NBRI} = \frac{\text{B8} - \text{B12}}{\text{B8} - \text{B12}}$$

BSI Bare Soil Index quantifies the soil mineral composition and the presence of vegetation.

$$\text{BSI} = \frac{(\text{B11} + \text{B4}) - (\text{B8} + \text{B2})}{(\text{B11} + \text{B4}) + (\text{B8} + \text{B2})}$$

NDWI Normalized Difference Water Index is used for the identification of water bodies although is sensitive to build-up land and result in over-estimated water bodies.

$$\text{NDWI} = \frac{\text{B3} - \text{B8}}{\text{B3} + \text{B8}}$$

NDSI Normalized Difference Snow Index identifies snow cover over land areas since in B11 snow absorbs most of the incident radiation while the clouds do not.

$$\text{NDSI} = \frac{\text{B3} - \text{B11}}{\text{B3} + \text{B11}}$$

1.2 Related work and state-of-the-art

1.2.1 Deep Learning & CNN

Deep learning have been a popular and efficient technique to solve challenges from satellite imagery. A neural network is a computational model inspired by the structure and functioning of the human brain. It's a type of machine learning algorithm that's used for a wide range of tasks, including pattern recognition, classification, regression, and more. Neural networks consist of interconnected nodes, or artificial neurons, organized into layers. Each neuron processes and transmits information, and the connections between neurons have associated weights that are learned from data. One of the simplest forms of a feedforward neural network is the Multi-Layer Perceptron (MLP), which is composed of multiple layers of interconnected neurons arranged in a sequence of at least three layers: an input layer, one or more hidden layers, and an output layer

Neural networks have found widespread applications in various fields, including computer vision, natural language processing, speech recognition, and reinforcement learning. Specifically, Convolutional Neural Network (CNN) have been the main architecture of neural networks to provide a solution from image-based problems. What differentiates CNN from the standard MLP is that they have hidden layers called convolutional layers, which are able to take patterns from their inputs using filters. A filter can be considered a relatively small matrix that slides from the input by operating with the weights of the matrix. This action is called *convolving* and, in its simplest case, the output value of a layer with (N, C_{in}, H, W) as the input and $(N, C_{out}, H_{out}, W_{out})$ as the output is described as:

$$out(N_i, C_{out_j}) = bias(C_{out_j}) + \sum_{k=0}^{C_{in}-1} weight(C_{out_j}, k) \star input(N_i, k)$$

where $\star$ is the 2D cross-correlation operator, N is the batch size, C denotes the number of channels and H and W are the height and width in pixels of the planes respectively. A graphic representation of a CNN architecture can be seen in 2

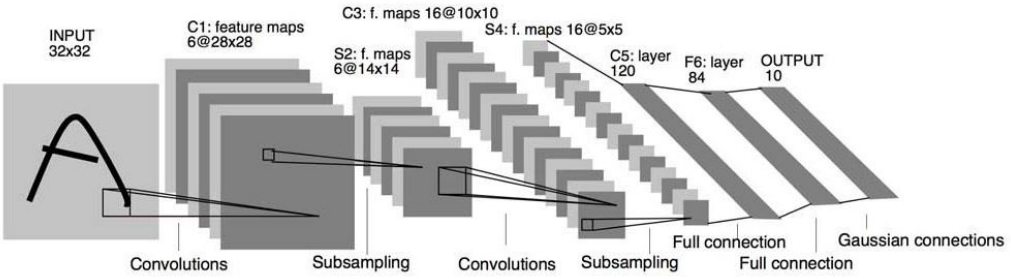


Figure 2: LeNet-5 architecture.

Being that told so, there are a lot of state-of-the-art using CNN. In [4], they create a deep learning approach to Sentinel-2 super-resolution. Their hypothesis was the existence of a complex mixture of correlations across many spectral bands over a large spatial context. Hence, the input of the model is a concatenation of the high-level resolution bands with the low-level resolution bandwidths upsampled to 10m by simple bi-linear interpolations. The model itself is a clear reference of residual networks [5].

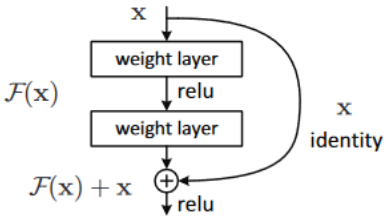


Figure 3: Residual learning: a building block.

Residual architectures use contiguous blocks of the convolution layers with the same properties. This layers are connected also with skip connections to reduce the average effective path length through the network, alleviate the vanishing gradient problem and greatly accelerates the learning during the training. As it can be seen in 4, networks without skip connections are harder to train as the loss surface is to hard to navigate. [6]

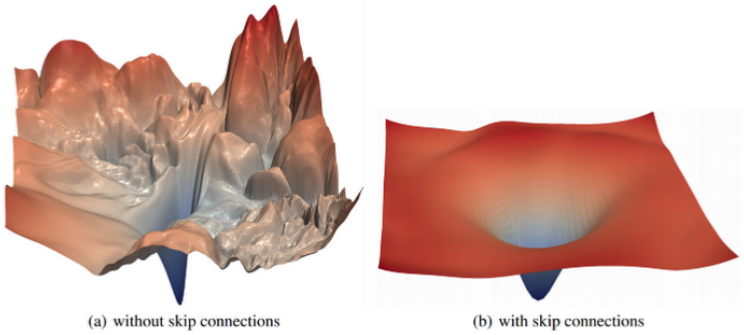


Figure 4: Loss surfaces of ResNet-56 with/without skip connections.

Similarly, in [7], a residual network is used with the same skip connections mechanism to bring a solution to cloud removal challenge. Regarding the design of the neural network, DSen2-CR is a fully convolutional network, so it can accept input images of any spatial dimensions (m), as it can be seen in 5. The output of DSen2-CR is a 13-channel layer, representing the thirteen bands from Sentinel2.

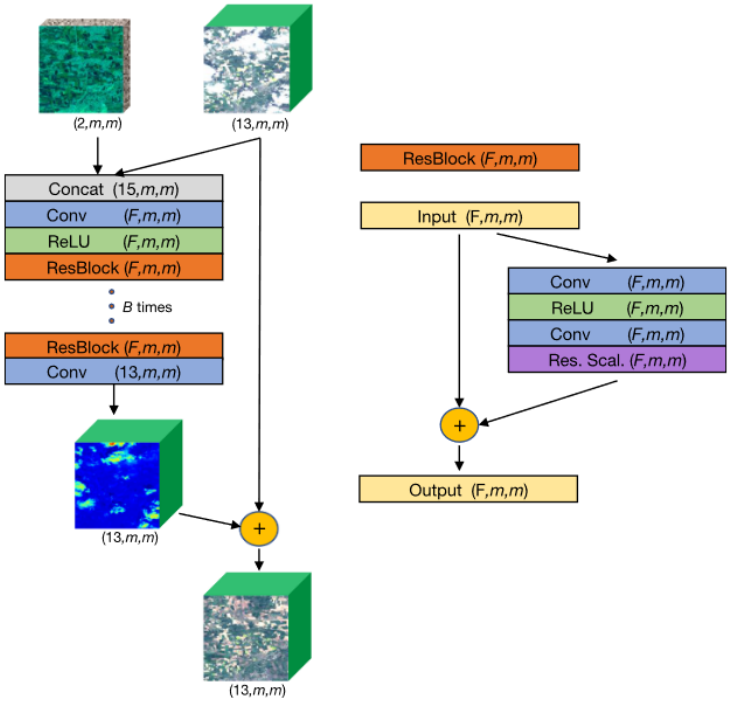


Figure 5: DSen2-CR model diagram.

It also uses Synthetic Aperture Radar (SAR) optical data, which represents an important complementary source to help the model to make greater results. However, SAR images are affected by a particular type of noise called *speckle*, which can difficult network's learning

and effectiveness. In addition to that, the model depends on one more source to remove the clouds, as SAR images cannot be downloaded in Sentinel-2. It can be seen in 6 that the lack of SAR data make the network less powerful.

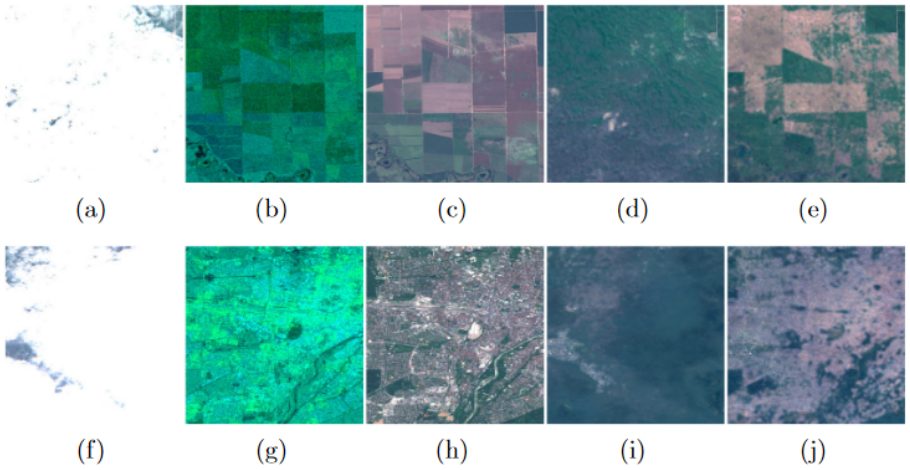


Figure 6: Examples of the influence of SAR data. Being (a, f) cloudy images, (b, g), SAR images, (c, h), the target cloudless images, (d, i), the synthesized images without SAR input data and (e, j) the predictions using SAR data as input.

1.2.2 Generative Adversarial Networks

Improvements have been found using Generative Adversarial Network (GAN) [8], which is a model architecture that belongs to the set of generative models. Generative models are a class of machine learning that learn a representation of the data trained on and they model the data itself. GAN is an unsupervised model made up of two neural networks: the generator and the discriminator. The idea is based on a game theoretic scenario in which the generator network must compete against an adversary. While the generator network produces samples, the aim of the discriminator is to distinguish between the real samples and the drawn by the generator. The discriminator is a binary classifier trying not to be fooled.

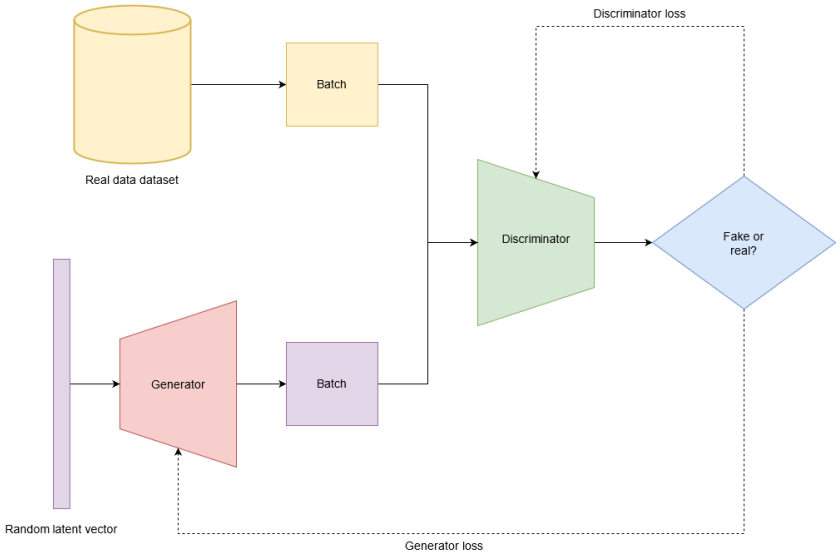


Figure 7: GAN model diagram.

The generator loss is calculated using the discriminator as a reference of how much far is from real images while the discriminator loss is calculated by how much accurate is discerning between the synthesized data and the real one. The standard function can be known as the min-max loss:

$$Loss_D(D) = E_x[log(D(x)] \tag{1}$$

$$Loss_G(G) = E_y[log(1 - D(G(y)))] \tag{2}$$

$$Loss_{GAN}(G, D) = Loss_D(D) + Loss_G(G) \tag{3}$$

During training, both networks constantly try to outsmart each other, in a zero-sum game. At some point of the training, the game may end up in a state that game theorists call a Nash equilibrium, when no player would be better off changing their own strategy, assuming the other players do not change theirs. GANs can only reach one possible Nash equilibrium: when the generator produces so realistic images that the discriminator is forced to guess by 50 % probability that the image is real or not. Nevertheless, the training process not always can converge to that equilibrium. There are several factors that make the training hard to reach the desired state. For instance, there is a possibility that the discriminator always outsmarts the generator so that it can clearly distinguish between fake and real images. As it never fails, the generator is stuck trying to produce better images as it cannot learn from the errors of the discriminator. Possible solutions can be carried out such as making the discriminator less powerful, decreasing the learning rate or adding noise to the

discriminator target. Another big obstacle is when the generator becomes less diverse, and it learns only to perfectly generate realistic images of a single class, so it forgets about the others. This is called *mode collapse*. At some point, the discriminator can learn how to beat the generator, but then, the latter is forced to do the same but in another class, cycling between classes never becoming good at any of them. A popular technique to avoid is *experience replay*, which consists in storing synthetic images at each iteration in a replay buffer. There is a lot of literature of obstacles and solutions to improve GAN training and it is still very active, as it is in its applications too. The tuning of hyper-parameters and the design of the model will be a key to pursue the Nash equilibrium. For instance, there is a variant called Conditional Generative Adversarial Network (cGAN). Traditionally, the generative network only produces the image from a random vector as an input, which is also called *latent vector* since it cannot be manipulated or with prior convictions of how will be. Unfortunately, this only allows to generate a random image from the domain of the latent space, which is hard to map to the generated images. However, cGAN can be trained so that both generator and discriminator models can be conditioned to some class labels or multi-dimensional vectors and produce synthetic images from a specific domain. In 8, it can be seen a cGAN diagram where the generator is conditioned by some inputs as well as the real data have the condition vector in each sample.

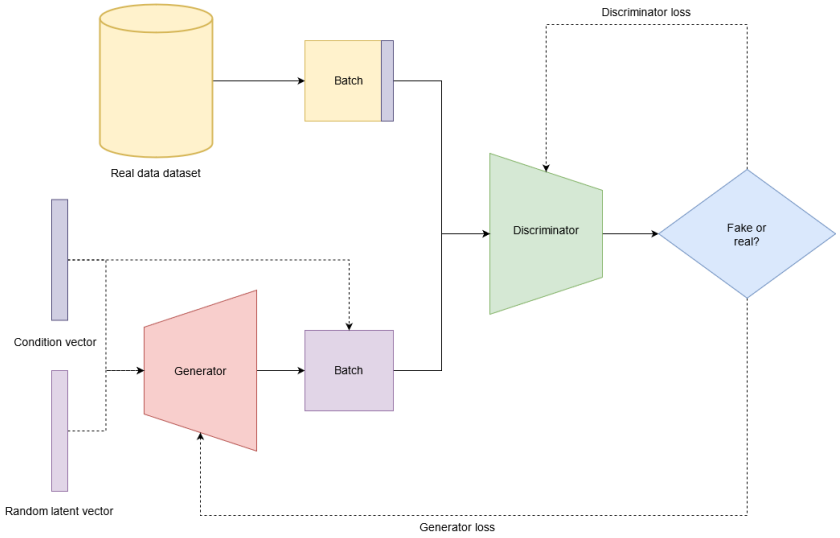


Figure 8: cGAN model diagram.

Actually, it is hard to find another GAN that can fit better in cloud removal problem without being cGAN, since the model needs to be conditioned to the input image. In [9], they proposed a McGAN trained using RGB and NIR cloud-free bands as input and

synthetic RBG cloudy images as target. It is true that short wave bands are unaffected by cloud cover and using NIR images to guide to uncover the clouds of satellite imagery is great since NIR bands posses have higher penetration through fog than visible light bands. Nevertheless, synthesizing the target might not be realistic enough to feasibly deploy the model in real-state. Regarding the experimental results, they have only showed qualitative metrics by comparing the ground-truth with cloud-free images, ones synthesized and ones real, accompanied both by the cloud obscure image and the cloud mask. As the NIR channel is not altered by the perlin noise, there is a huge difference about the effectiveness comparing the synthesized images with the real ones, as it can be seen in 9 and 10

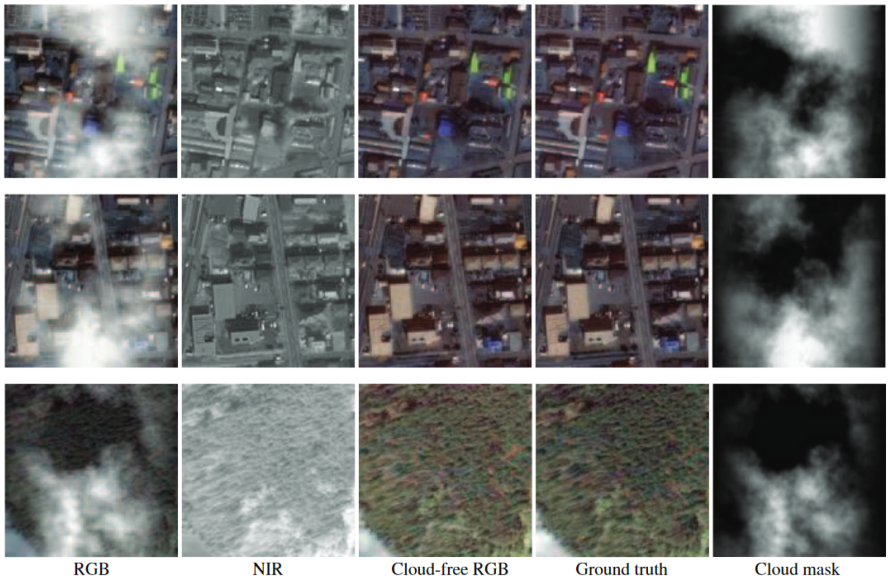


Figure 9: Prediction results by McGAN with the synthesized cloud images

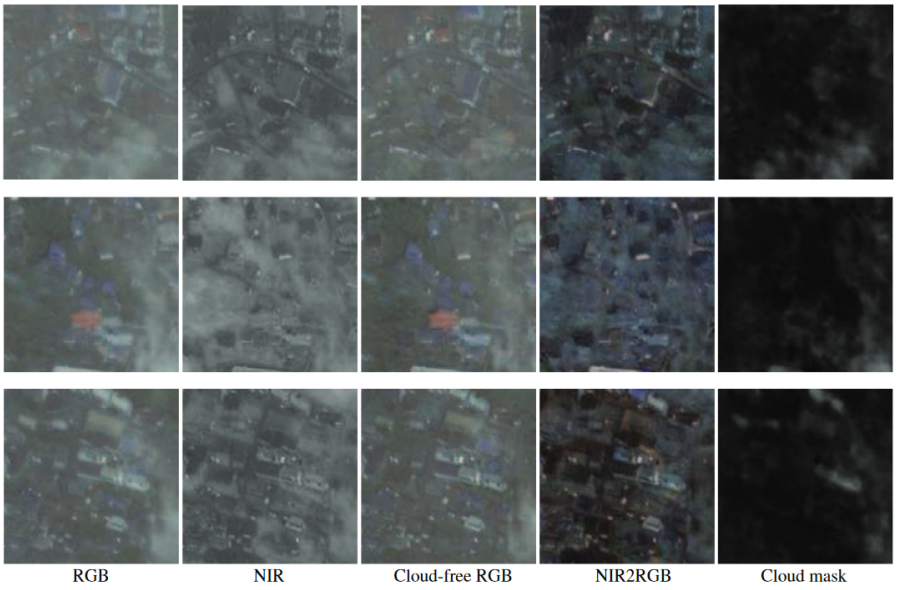


Figure 10: Prediction results by McGAN with the real cloud images

To overcome the need of synthesized data, [10] proposed two datasets and two models respective to each of the datasets. Both neural networks have versions depending on including or excluding the NIR channel as an input. The first is based on Pix2Pix [11] model, a cGAN which performs very well in image-to-image translation tasks. The generator network of Pix2Pix uses a encoder-decoder architecture but with skip-connections to avoid loss of information of the bottleneck, similar to U-Net architecture [12]. The model is trained by a paired dataset of single-images. That means the dataset have a cloudy image as input and a real cloud-free image of the same zone. The generator tries to produce the image as real as the target ones, so it is compulsory to have both images. The second model proposed is a spatio-temporal cGAN trained by pairing clear images with three cloudy corresponding from diverse points in time. Instead of only trying the U-Net architecture as in the first model, they have proposed two designs for the generator:

- A branched ResNet generator. It has three encoder-decoder residual networks to learn feature maps from each image, then concatenating their output to learn from these features again and finally, concatenating the result through a final encoder-decoder to generate the new image.

- A branched U-Net generator. A slightly modification of the U-Net to allow multiple input images to encode them separately but decoding together and skip connections.

Moreover, they have also created two paired datasets to train the single-image models and

the multi-temporal models respectively to overcome the need of synthesizing the images to make them cloud obscured.

Regarding experimental results, they have compared the single-image model with McGAN. On the other hand, they have compared the multi-temporal model with their different versions since it was the first of its kind. The results can be found in 11 and 12. All models from the same comparison were trained using the same datasets created by them. In both cases they have used Peak signal-to-noise ratio (PSNR) and Structural Similarity Index Metric (SSIM) as quantitative metrics and random figure comparison as qualitative metrics. While PSNR is a pixel-based metric capturing the difference between the corresponding pixels from two images, SSIM tracks similarity between visible structures and large-scale features in the images.

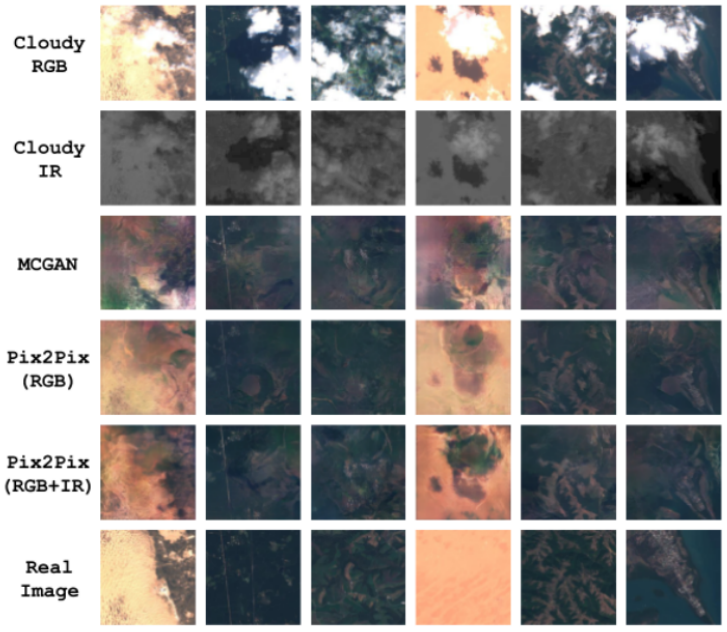


Figure 11: Comparison between the cloudy images, the outputs of McGAN and Pix2Pix-based models and the ground-truth image.

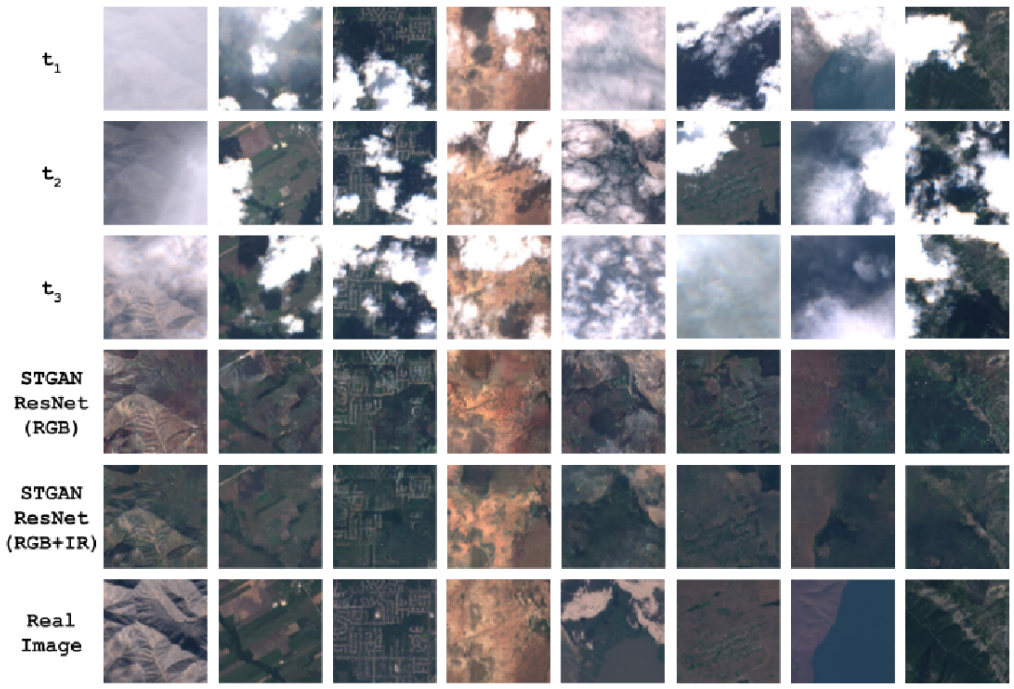


Figure 12: Comparison between cloudy images, the outputs of McGAN versions and the ground-truth image.

Table 2: PSNR and SSIM of the single-image and multi-temporal model versions .

	Validation Set		Test Set	
Model	PSNR	SSIM	PSNR	SSIM
Pix2Pix (RGB)	23.130	0.442	22.894	0.437
Pix2Pix (RGB + IR)	21.352	0.485	21.146	0.481
MCGAN (RGB + IR)	20.871	0.424	21.013	0.381
STGAN U-Net (RGB)	25.484	0.534	25.822	0.564
STGAN ResNet (RGB)	25.519	0.550	26.000	0.573
STGAN U-Net (RGB + IR)	25.142	0.651	25.388	0.661
STGAN ResNet (RGB + IR)	25.628	0.724	26.186	0.734
Raw Cloudy Images	7.926	0.389	8.289	0.422

To overcome the paired-data, Cloud-GAN [13] is a CycleGAN [14] which uses two generators (G_A and G_B) and two discriminators (D_A and D_B). CycleGANs can create new samples of output data, but also transforming the desired data to samples of input data. In essence, they learn to transform data from the two sources by the two generators

respectively being also these sources the target. In other words, CycleGAN is a model that learns two data transformation functions between two domains. In our case, the generator G_A is generating cloud-free images from cloudy images while the generator G_B is turning the cloud-free images into cloudy. Hence, there is no need to train the model by paired cloudy-cloud-free imagery. The datasets for G_A and G_b are respectively:

$$X_A : \{x \mid x \text{ is a real cloud obscured image}\}$$
$$X_B : \{x \mid x \text{ is a real cloud-free image}\}$$
$$X = X_A \cup X_B$$

The synthesized data Y can be split into:

$$Y_A : \{y \mid y \text{ is a synthesized cloud obscured image}\}$$
$$Y_B : \{y \mid y \text{ is a synthesized cloud-free image}\}$$
$$Y = Y_A \cup Y_B$$
$$X_A \cap Y_A = \emptyset, \qquad X_B \cap Y_B = \emptyset$$

The transformation functions from each generator can be expressed as:

$$G_A : \text{Generates } y \in Y_B \text{ from } z \in Z_A, \qquad Z_A = X_A \cup Y_A$$
$$G_A : Z_A \longrightarrow Y_B$$
$$G_B : \text{Generates } y \in Y_A \text{ from } z \in Z_B, \qquad Z_B = X_B \cup Y_B$$
$$G_B : Z_B \longrightarrow Y_A$$

It is well-known that each generator will learn its corresponding transformation function by minimizing the loss, which is calculated by how much the discriminator can discern between their produced data and the real data. On the other hand, the discriminator loss is how good it is to distinguish between the real and the synthesized data.

However, training a cyclic GAN using only the two network losses does not guarantee that cycle consistency is held. Cycle consistency means that given an input, it is desired the back-and-forth transformation $G_B(G_A(z_A)) = z'_A$ to output the the original input z_A. In other words, we want the composition of the transformation functions to be the identity

function.

$$G_A \circ G_B = Id \iff G_B \circ G_A = Id$$

Thus, an additional cycle consistency loss is used to enforce this property. This loss is defined as the uniform norm between an input value z_A from the dataset Z_A and it's forward-cycle prediction $G_B(G_A(z_A))$ and the same for the data $z_B \in z_B$, $G_A(G_B(z_B))$. The higher the difference, the more distant the predictions are from the original inputs. Ideally, our network would also minimize this loss by a weighting factor λ which is used to control the relevancy of this loss compared to the others.

$$Loss_{Cycle} = E[G_B(G_A(z_A)) - z_A] + E[G_A(G_B(z_B)) - z_B]$$

$$Loss_{CycleGAN} = Loss_{GAN}(G_A, D_A) + Loss_{GAN}(G_B, D_B) + \lambda \cdot Loss_{Cycle}$$

As mentioned before, in [13] they did novel deep learning model for cloud removal, since CloudGAN did not need paired-data and they had promising results removing thin clouds, small cloud patches and synthetic clouds. They have great PSNR in 13 although they could only do it for synthetic clouds as the data is not paired, so that there is no quantitative results for real data, neither a comparison with other models proposed. In addition to that, the images to train and test seem to be not diverse enough since they only download satellite images over Paris region.

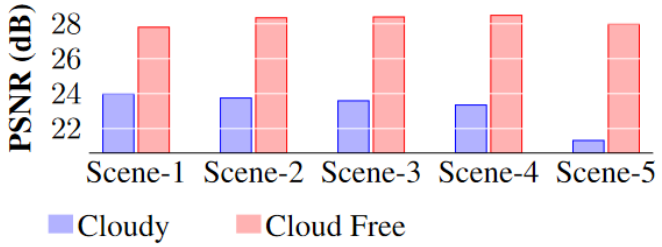


Figure 13: Failure cases in CloudGAN attempting to removal overly clouded images over-smoothening or completely failing.

Even though cyclic GANs work excellently to transforming images to different styles, they seemed to be not enough to create new shapes or providing geometric changes as well as they are not general enough providing unexpected results when the input data fed is quite different from the trained. This issue can be also found in Cloud-GAN as its removal cannot be performed well in overly clouded images 14.

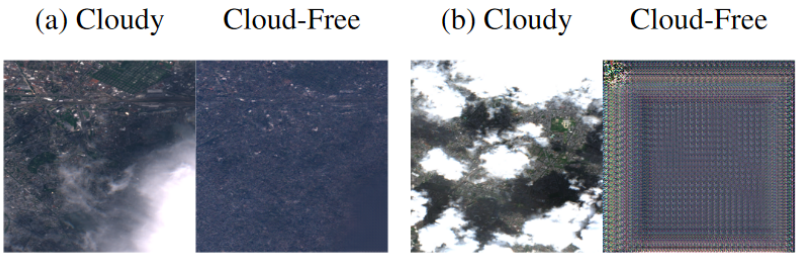


Figure 14: Failure cases in CloudGAN attempting to removal overly clouded images over-smoothening or completely failing.

This might also be because of the lack of more data in the training phase or, at least, enough variability of the data. In that case, [1] provides large data set for training new cloud removal approaches capturing a diverse range of observations from all continents and meteorological season. Also, they designed a cycle-GAN with two generators similar as [], but with changes of the generator of cloud-free images so that the generator receives the optical data, the three visible bands and the cloud map and outputs the cloud-free image and the cloud map prediction 15.

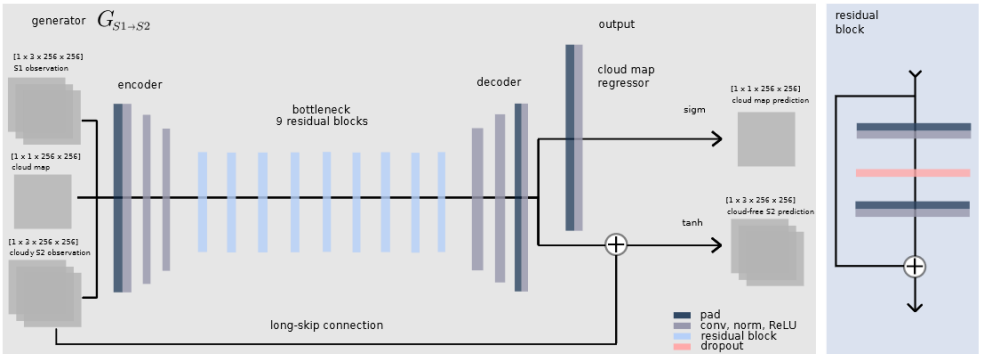


Figure 15: Generator $G_{\text{cloudy}\to\text{cloudfree}}$ of [1]

Regarding training, they used paired non-cloudy and cloudy optical observations, adding supervised losses to improve its performance. Secondly, they compared the performance of the model against a set of baseline models. This is important to determine whether your proposed model is significantly better than existing approaches and can provide new insights into the problem of recovering cloud-occluded information. Thirdly, they retrained the models on synthetic data of generated cloudy observations and evaluate them on real data to assess their generalization performance. This is important because models that perform well on synthetic data may not always generalize well to real-world scenarios.

Regarding track of training, they used precision, recall and f1 score to evaluate discriminators while Mean Absolute Error (MAE), MSE, PSNR, SSIM and Spectral Angle Mapper (SAM) were used to evaluate the generators.

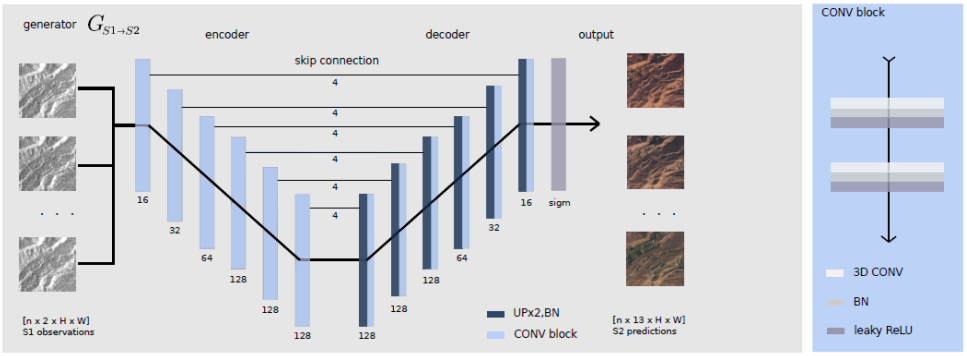


Figure 16: Generator $G_{\text{cloudy}\to\text{cloudfree}}$ of [2]

From a different baseline, in [2], they propose two models: one that predicts a cloud-free image given a sequence of cloudy optical and SAR images and a sequence-to-sequence translation model that predicts a cloud-free timeseries from cloud-covered time series.

1.2.3 Other generative models

Ideally, generative models should satisfy the following key requirements in a real environment:

High quality samples refers to those samples that captures the underlying patterns and structures present in the data making them indistinguishable from human observers.

Fast Sampling is about the efficiency of image generation and the computational overhead that can cause generative models.

Mode Coverage/Diversity points out how the model is able to generate a full range of mods and diverse patterns present in the training data.

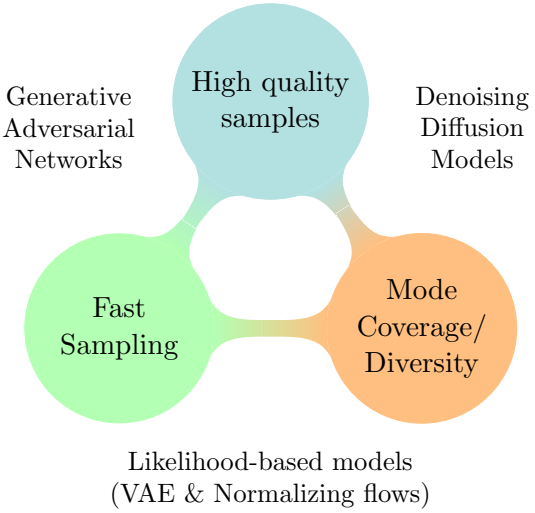


Figure 17: Generative Learning Trilemma

Since now, we have talked a lot about GANs, but there are other architectures of generative models. We have seen that GANs have serious problems with *mode collapse*, which affects the diversity of synthetic data. While likelihood-based models and Denoising Diffusion Model (DDM) can cover a wide spectrum of possibilities, they suffer from low sample quality and thousands of network evaluations respectively. Most of the deep generative learning focus on high-quality definition, although all the requirements are highly important and key factors in a real environment. This is called the *generative learning trilemma* 17.

Likelihood-based models are alternatives to GANs that can cover a wide range of possibilities since they focus on estimating the likelihood or probability distribution of the data while training. There are several types of likelihood-based models although we will focus on VAE [15]. The architecture of autoencoders are quite simple, they consist of an encoder, a smaller feature vector also called *latent vector* z and a decoder , as it can be seen in *Figure 18*. Therefore, the main goal of the encoder is to comprise the input vector x while the decoder attempts to perform the conversion from lower to higher dimensional data, being the output vector $\bar{x}$. Hence, the best purpose of autoencoders is dimensionality reduction given its architecture. Then, the main purpose is to find the best set of encoder/decoder that keeps the maximum information with the less reconstruction error while decoding. In fact, one of the most popular usages of autoencoders in computer vision is the image reconstruction due to their architecture.

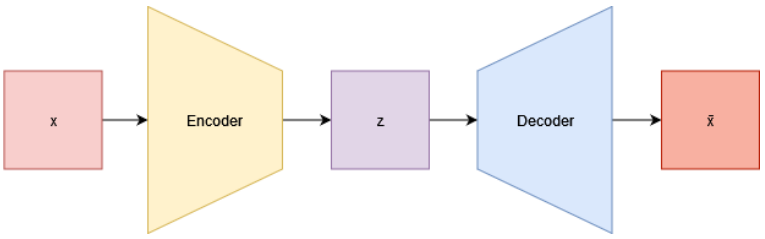


Figure 18: Autoencoder architecture.

$$\begin{aligned}\mathcal{L}_{\text{MSE}} &= \frac{1}{N}\sum_{i=1}^{N}\|x_i - \bar{x}_i\|^2 \\ &= \frac{1}{N}\sum_{i=1}^{N}\|x_i - \text{decoder}(\text{encoder}(x_i))\|^2\end{aligned}$$

Figure 19: MSE loss in autoencoders

Let's assume an autoencoder that both encoder and decoder have only one layer without non-linearity. In that sense, we can see clearly a link with PCA since we are looking for the best linear subspace to project data on with as few information loss as possible. However, deep neural networks comes with also non-linear layers[1]. The more complex of autoencoders architecture, the more they can proceed to a high dimensionality reduction while keeping the loss low. In an ideal case, if the encoder and the decoder have enough degrees of freedom and infinite power, the latent vector could be reduced to 1. Nevertheless, the dimensionality reduction comes with a price. First of all, we will lack of interpretable structures in the latent space. Secondly, the major part of the data structure information will not be in a reduced representation but in arbitrary one without any context of the patterns that the autoencoder could infer. Therefore, it is important to control and adjust the depth and the latent space dimensions depending on the final purpose.

At this point, we have learned the surface from autoencoders, but how do they fit in image generation? Once the autoencoder has been trained, we have both encoder and decoder with the weights to reconstruct the data. At first, we might think that changing the latent vector we could take a point randomly from the space and decode it to get a new content. Although that could work, the regularity of the latent space for autoencoders is hard since it depends on several factors such as the data distribution, the latent space

[1]Non-linear layers allow to learn complex relationships. Some of them are used as activation functions.

dimension and the architecture of the encoder. To sum up, there will be several biases that we are not able to control.

In order to make the latent vector regular and continuous, VAEs try to solve this problem by mapping the inputs to a normal probability distribution, so they introduce explicit regularization during the training process. Hence, the latent vector will be sampled from that distribution, being the decoder more robust at decoding latent vectors as a result. A VAE is an architecture composed of both encoder and decoder too. However, instead of encoding a latent vector, they encode it as a distribution over the latent space. The following enumeration details the process:

1. The input is encoded as a distribution over the latent space.
2. A point is generated given the distribution encoded.
3. The sampled point is decoded so the reconstruction and regularization error can be computed.

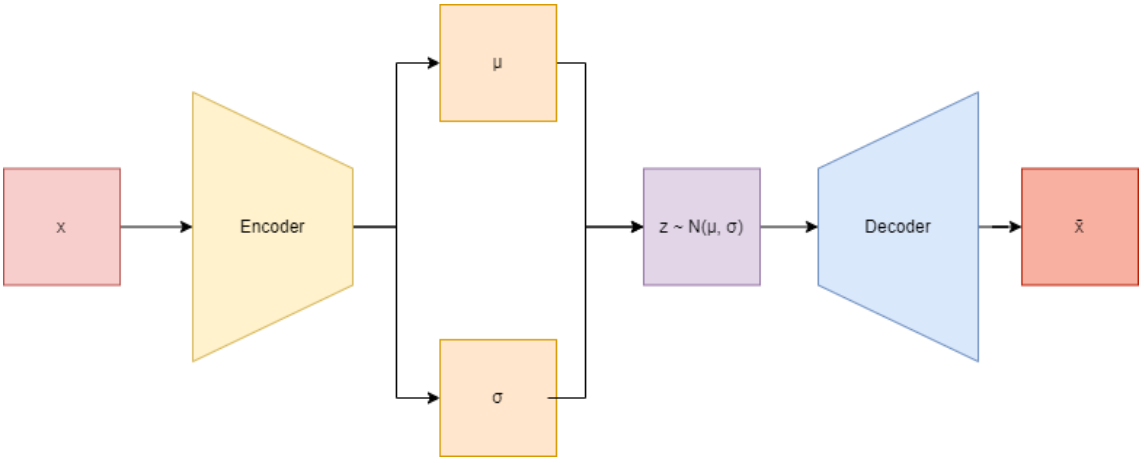


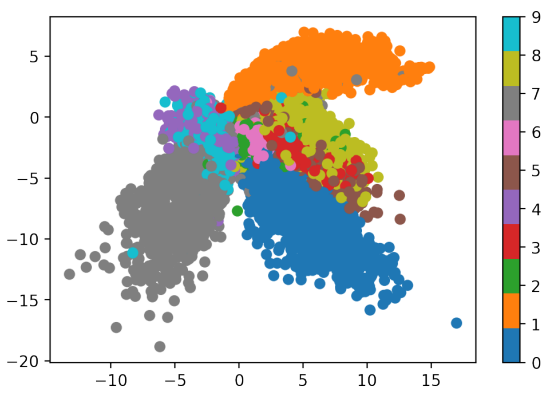
Figure 20: Autoencoder architecture.

The encoded distributions are chosen to be normal so that the encoder can be trained to return the mean and covariance matrix. This makes a way of both local and global regularization of the latent space respectively. Hence,the loss will not be only about the reconstruction of the data in the last layer, called *reconstruction term*, but also how the latent space is organized by making the latent space close to a standard normal distribution, called the *regularization term*. The latter will be calculated by how distant is from the gaussian distribution. Kullback-Leibler (KL) Divergence, is a measure of how

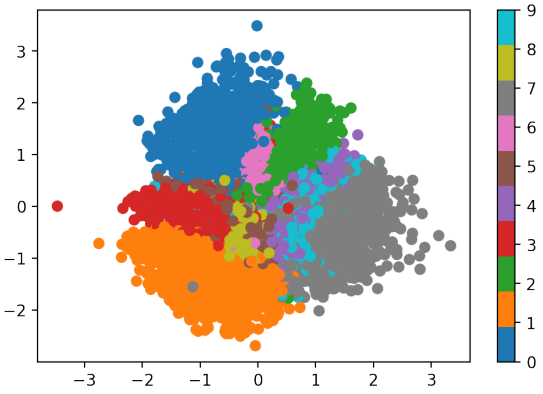
one probability distribution differs from a second, reference probability distribution. It's commonly used in various fields, including information theory, statistics, and machine learning. Using the KL divergence, which is expressed in terms of the means and the covariance matrices of the two distributions, the variety of outputs are better represented in the latent space in VAEs than traditional autoencoders:

$$\begin{aligned} \mathcal{L}_{\text{VAE}} &= \text{Reconstruction term} \quad + \quad \text{Regularization term} \\ &= \left(\frac{1}{N}\sum_{i=1}^{N}\|x_i - \bar{x}_i\|^2\right) \quad + \quad \sum_{j=1}^{J}\left(\mu_j^2 + \sigma_j^2 - \log(\sigma_j) - 1\right) \end{aligned} \tag{4}$$

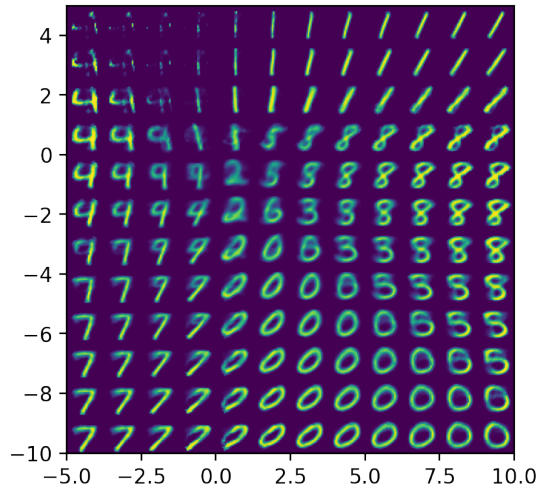
The loss in VAE is also called the variational lower bound or Evidence Lower Bound (ELBO) as a way to estimate the likelihood of the observed data given the model's parameters and latent variables. To demonstrate the differences we can compare an autoencoder and a VAE trained by MNIST data, although it is a really balanced dataset. We can see in that the range of values in the latent vector of the latter is much smaller and centralized, representing better any class, whereas autoencoder has the images more sparsed in the latent domain.



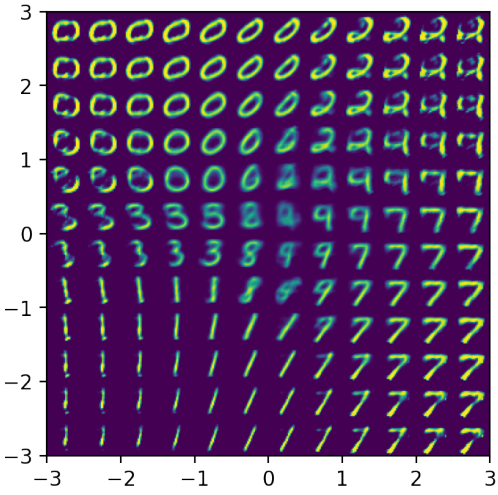
(a) Latent vector scatter plot of autoencoder



(b) Latent vector scatter plot of VAE



(c) Reconstructed digits from autoencoder given latent domain



(d) Reconstructed digits from VAE given latent domain

Figure 21: Comparison between non-probabilistic autoencoder and VAE.

Moreover, one particularity that makes VAE architectures good in business and industry related problems or even in processing images[2] is that they are really fast when sampling due to its bottleneck and the dimensional reduction feature. Nevertheless, the reduction of the bottleneck also makes that the quality of the samples produced by VAEs are lower than other generative models. In particular, when they are compared to DDMs.

DDM [16] are a type of generative model that operates through a different framework than VAEs, although their basis is also probabilistic. While VAEs focus on encoding data into a latent space and then decoding it to generate reconstructions, DDMs revolves around modeling the diffusion process of data. This process involves the gradual transition from a

[2]One problem that VAEs solve really well is in image reconstruction since it gets the main and reduced idea of an image.

noisy version of the data to the actual clean data. The process of training a DDM involves optimizing the parameters of the diffusion process so that it can effectively recreate the observed data distribution. But, what is this process about? We can assume that all data comes from a distribution. Basically, any dataset is sampled from the real distribution. The goal of generative models is to learn that distribution so they could sample from it and get another data point that looks like is from the dataset trained. The way that DDM learns that distribution is by trying to convert well-known and simple base distribution (like gaussian) to the target data iteratively, with small steps, using a Markov chain, treating the output of the markov chain as the model's approximation for the learned distribution. Hence, the diffusion process can be split into two parts: forward and reverse diffusion processes.

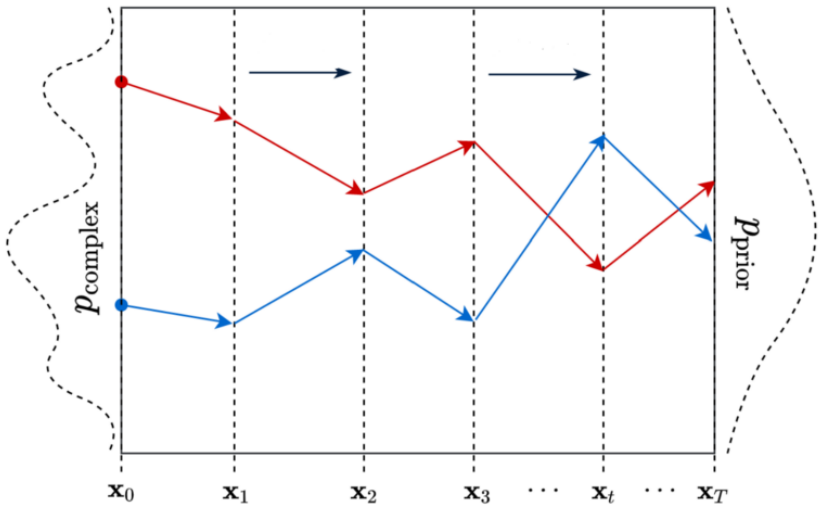


Figure 22: Forward diffusion process mapping

In the forward process $q(x_t|x_{t-1})$, the model slowly and iteratively add noise to the images so that they move out from their existing subspace.

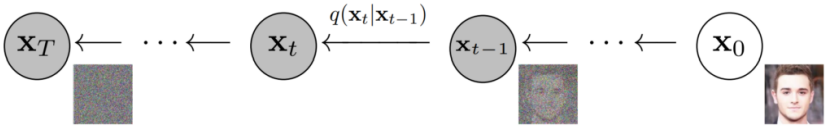


Figure 23: Forward diffusion process

Since it is proved that doing it infinitely the image would eventually end up into a point from a normal distribution $q(x_T|x_0) \approx \mathcal{N}(0, 1)$, the final corrupted image would loss all information from original sample. So, what it is aimed is to convert the unknown and

complex distribution of the dataset into one that it is easier to sample a point from and understand. The forward process takes the form of a markov chain, where the distribution at a particular time step only depends on the sample of the immediately previous step. Therefore, it is easy to write out the distribution of corrupted samples conditioned on the initial data point x_0 as the product of successive single step conditionals:

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$

In the case of continuous data, each transition is parameterized as a diagonal gaussian probability distribution function and that is the reason why approximating to T would end up to a gaussian distribution centered at 0, since the parameter β will always approach to 1 and, then, $\sqrt{1-\beta_T}$ theoretically will approach to 0:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

$$\beta_1 < \beta_2 < ... < \beta_T \quad ; \quad \beta_t \in (0,1)$$

Although taking small steps has a cost, learning to undo the steps of the forward process would be less difficult. Adding little noise at each step, there would be less ambiguity about determining the probability density function of the last step in the reverse process.

The goal of the reverse process is to undo the forward and to learn the denoising process in order to generate new data from random noise. Unfortunately, infinite paths can be taken starting from the corrupted image but only few of them will turn the noisy image into a data from the desired subspace. Hence, DDM takes small iterative steps during the forward diffusion processes and take those steps which the probability distribution function satisfies that the corrupted images differs slightly at each step.

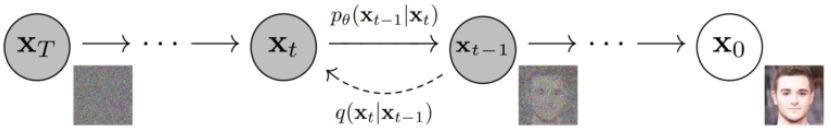


Figure 24: Forward and diverse diffusion processes

Like the forward process, the reverse process is set up as a markov chain, being the pure noise distribution $p(x_T) = \mathcal{N}(x_T; 0, 1)$:

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$

[17] shows that theoratically the true reverse process will have the same functional form as the forward process. Therefore, the each learned reverse step will be also a diagonal gaussian distribution:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t; t), \sigma_\theta(x_t, t))$$

The forward objective is to push a sample off the data manifold turning into noise and the reverse process is trained to produce a trajectory back to the data manifold, resulting in a reasonable sample. The objective that we attempt to optimize is not about optimizing the maximum likelihood objective to turn a point to x_0, because $p_\theta(x_0) = \int p_\theta(x_{0:T}) dx_{1:T})$ includes all the possible trajectories, all the ways a noisy point could have arrived at x_0. If we compare VAE with DDM, the encoding part would be the forward process while the decoder would be the reverse model. Hence, the latent variables would be $\{x_i | i \in \{1, 2, ..., T\}\}$ while the observed variable, x_0. Unlike a VAE, the encoder part is typically fixed but the reversed process is the one being focused while training, so a single network needs to be trained. When we have a model with observations and latent variable, we can use the ELBO to maximize the expected density assigned to the data while the KL divergence encourages the approximate posterior $q(z|x)$ to be similar to the prior on the latent variable $p_\theta(z)$.

$$\begin{aligned}
log\, p_\theta(x) \geq & \quad \mathbb{E}_{q(x_{1:T}|x_0)}[log\, p_\theta(x_0|x_{1:T})] - D_{KL}(q(x_{1:T}|x_0)||p_\theta(x_{1:T})) \\
\geq & \quad \mathbb{E}_q[log\, p_\theta(x_0|x_{1:T})] - \mathbb{E}_q[log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{1:T})}] \\
\geq & \quad \mathbb{E}_q[log\, p_\theta(x_0|x_{1:T})] - \mathbb{E}_q[log \frac{p_\theta(x_{1:T})}{q(x_{1:T}|x_0)}] \\
\geq & \quad \mathbb{E}_q[log \frac{p_\theta(x_{0:T})}{q(x_{1:T|x_0})}] \\
\geq & \quad \mathbb{E}_q[log\, p_\theta(x_T) + \sum_{t \geq 1} log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}]
\end{aligned}$$

Therefore, any step forward and back would include a loss of the divergence between both distributions. Nevertheless, although at training time any term of this objective can be obtained without having to simulate an entire chain, different trajectories may visit different samples at time $t-1$ on the way to hitting x_t, the setup can have high variance. limiting training efficiency. To help with this, the objective must be arranged as follows.

$$\mathbb{E}_q[-D_{KL}(q(x_T|x_0) \mid\mid p(x_T)) - \sum_{t>1} D_{KL}(q(x_{t-1}|x_t, x_0) \mid\mid p_\theta(x_{t-1}|x_t)) + log\, p_\theta(x_0|x_1)] \quad (5)$$

The first part it compares the noise distribution[3] $p(x_T)$ with the forward process $q(x_T|x_0)$, which both are fixed. The second component is a sum of divergences each between a reverse step and a forward process posterior conditioned on x_0. When the x_0 is treated as known like it is during training, all q terms are actually gaussians. Then, the divergences is comparing two gaussians and helps reducing variance during the training process.

In order to add conditional inputs in the model is to feed the conditioning variable y as an additional input during training $p_{theta}(x_0|y)$. Moreover, adding a separate trained classifier [18] can help guiding the diffusion process in the direction of the gradient of the target label probability with respect of the current noise image.

Comparing DDMs and VAEs, DDMs can capture complex dependencies and distributions in the data due to their inherent modeling of the diffusion process. Estimating and analyzing small step sizes is more tractable than describing a single non-normalizable step from random noise to the learned distributions, which is what VAE and GAN do. However, they can be computationally more intensive to train due to the intricacies involved in estimating the diffusion process parameters accurately.

1.2.4 Transformers in computer vision

CNN have worked very well for cloud removal, but latest and disruptive state-of-the-art deep learning attention-based architectures [19] uncover new paths to achieve remarkable improvements and results. It has been demonstrated that transformers can excellently overcome challenges such Natural Language Processing (NLP) [20], Text-To-Image Generation [21] or Image Completion [22] with large datasets, great model size and enough compute.

[3]The start of the reverse process

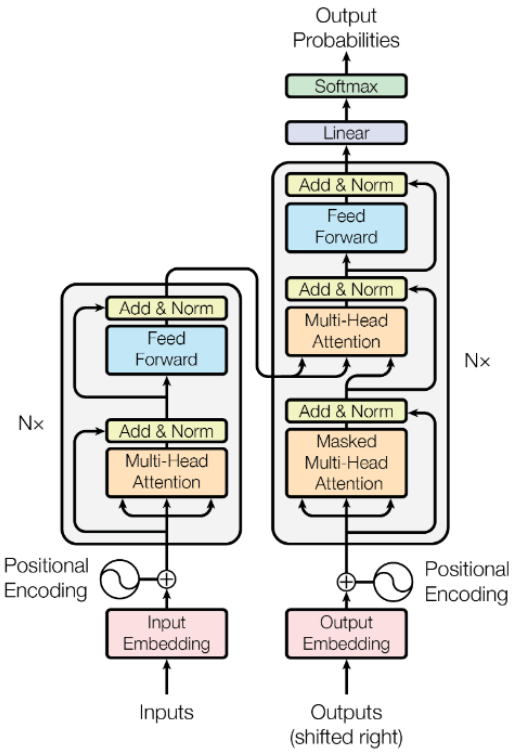


Figure 25: Transformer block architecture.
left block, the encoder architecture, **right block**, the decoder architecture.

The transformer neural network is composed by an encoder-decoder architecture much like RNN. However, the difference is that the input sequence can be passed in parallel by passing also the positional encoder zipped with, as the input might have different meaning depending on its position. Therefore, the positional encoder is a vector that gives context based on position of the element in a sentence. [19] uses the following sine and cosine function to generate this vector. For every odd step, they create the vector using the cosine function while for every even time step, they use the sine function. These functions have linear properties the model can easily learn to attend to when adding these vectors to their corresponding vector.

$$PE(pos, 2i+1) = cos(\frac{pos}{10000^{2i/dmodel}})$$
$$PE(pos, 2i) = sin(\frac{pos}{10000^{2i/dmodel}})$$

The input and the positional encoding are passed into the encoder block. The job of the encoder is to map all input sequence into abstract continuous representation that holds the learned information for that entire sequence. The encoder block has N identical encoder

layers. Each layer has two sub-layers: a multi-head attention layer and a feed forward layer. Both sub-layers have a residual connection and a layer normalization[4] next to their output vector. The residual connections helps the network to train by allowing gradients to flow directly through the network while the normalization is used to stabilize the network.

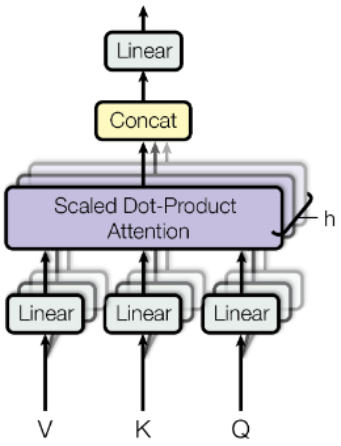


Figure 26: Multi-Head Attention block

Multi-head attention block applies a specific attention mechanism called self-attention, which allows the model to associate each individual element in the input to other elements. The attention is computed by how relevant is the i^{th} element of the sentence to other words in the same sentence. This means that it is computing what part of the sentence should the model focus. Therefore, we can understand the attention as a vector that captures the contextual relationships between elements in the sentence. To give the encoder model more representation power of the self-attention, the block is split into several blocks called heads, which can be run concurrently. The input of each head is fed into three distinct fully connected layers to create the query, key and value vectors. The idea is that the attention block must map the query against a set of keys to then present the best attention, which will be embedded to the values. In other words, the query is a vector related with what we encode, the key is a vector related with what we use as input to output and the value is the learned vector as a result of calculations but related with the input. Regarding encoder block, all three vectors are the source vector since what is wanted is to capture the attention of the elements between them.

[4]The normalization is done by each feature instead by each sample.

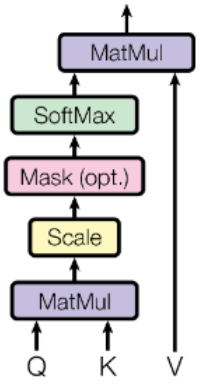


Figure 27: Scaled Dot Product Attention Head

To obtain self-attention, we will multiply the queries and the keys to obtain the score matrix, which determines how much focus should the i^{th} element be put on the other elements so each element will have a score to correspond to other elements in the time step. The matrices will be scaled by $\sqrt{d_k}$, where d_k is the dimension of the queries or the keys. This allows more stable gradients since multiplying values can have exploding effects. The softmax will turn the score matrix into a probabilistic matrix which any value is between 0 and 1 and the sum of each column or rows will be 1. A higher probability score means more focus. As it can be sensed, the scaled is also done because of the softmax and to prevent extremely small gradients. Finally, the attention score will be multiplied to the value vector and the result will be the output of the Scale Dot-Product Attention Head. Each output of the heads will be concatenated and then finally processed in a linear layer which output is the vector with encoded information on how each element should attend to all other elements in a sequence.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

As it was said before, the other sub-layer of a encoder layer is the feed-forward layer, which is a network that is applied to the attention vectors. This layer is used to turn these vectors into a form that the next encoder or decoder block can get as an input so that it gives a richer representation to the next block.

Regarding the decoder block, it has several similarities with the encoder block. They both have N identical layers and a position encoding at first of all. However, multi-head

attention layers of the decoder block have different job compared to the encoder. The decoder is auto-regressive and it takes the previous outputs from itself and the encoder output vector as inputs. This is because the encoder can use all the elements of the input sentence but the decoder can only use the previous elements $\{ j^{\text{th}} \text{ element } \mid j < i \}$ of the sentence. By not doing that, there would be no learning at all. Therefore, the first multi-head attention layer is masked so that it prevent it from being conditioned of the future tokens. In order to do that, once having the score matrix in a head, the result will be masked before calculating the softmax. Regarding the second multi-head attention layer, it can be seen that the queries and the keys are the encoder output while the values are the output of the first attention layer of the decoder, which also are the residual connection. Then, the result is passed into a feed forward layer for further processing and finally, goes to a linear layer that access a classifier and a softmax function to turn it into a probability distribution.

Table 3: Complexity per layer and number of sequential operations for different type layers. n is the sequence length, d is the representation dimension and k is the kernel size of convolutions.

Layer Type	Complexity per layer	Sequential operations
Self-Attention	$O(n^2 \cdot d)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	O(1)

Although transformers are great for sentence data such as text, its applications to computer vision problems do not fit as CNN do, since the attention has a quadratic cost. For that reason, [23] created a self-attention architecture called ViT that can perform better in some cases than CNNs reshaping the input to fit in the transformer architecture.

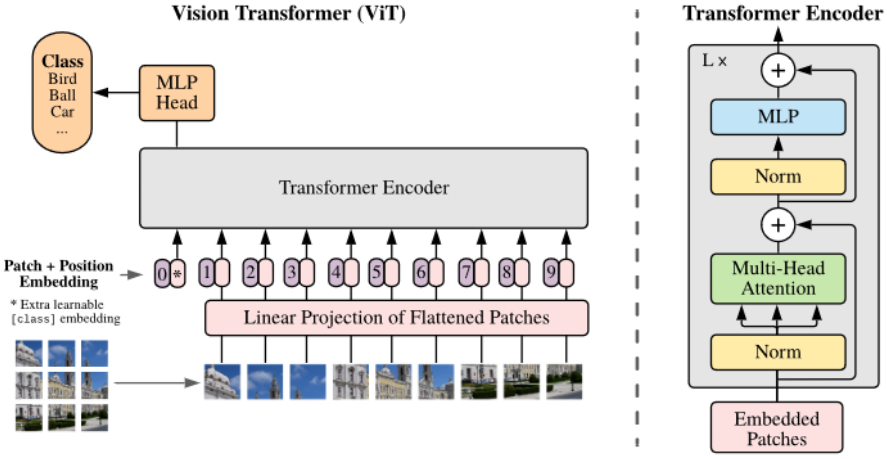


Figure 28: ViT architecture

As it has been said, transformer's input is a 1D sequence. To turn the images into sequences, the 2D images $\{x|x \in \mathbb{R}^{H\times W\times C}\}$ are reshaped into sequences of flattened 2D patches $\{x|x \in \mathbb{R}^{N\times(P^2\times C)}\}$ to be handled, where P is the resolution of each image patch and $N = \frac{HW}{P^2}$ is the number of patches. Therefore, P is the input sequence length. Once the splitting is done, the patches are mapped to D dimensions with a trainable linear projection, since the transformer encoder uses a constant latent vector of size D through all of its layers. This process is called patch embedding. The positional embedding added to the patch embedding to retain positional information and to be fed to the transformer encoder.

A more general way to understand ViT is that this architecture attempts to capture global attention of the image by putting the patches in the positional embedding while CNN captures pixel attention over a kernel, understanding better by nature the textures of an image. Nowadays, ViTs are popular models to solve computer vision problems, since they can capture better global self-attention. However, if they are not trained on huge datasets, the best option is to stick to ResNet or EfficientNet, since in this case ViTs cannot outperform the CNNs.

Regarding not only spatial but temporal computer vision challenges, L-TAE is a network architecture that employs multi-headed self-attention mechanisms to classify remote sensing time sequences. In the Temporal Attention Encoder (TAE), the channels of the temporal inputs are distributed among several attention heads operating in parallel. Each

head extracts highly-specialized temporal features which are in turn concatenated into a single representation. The proposed approach in [24] modifies the TAE by distributing the channels of the temporal inputs among several compact attention heads operating in parallel, which allows for more efficient processing of time-series data.

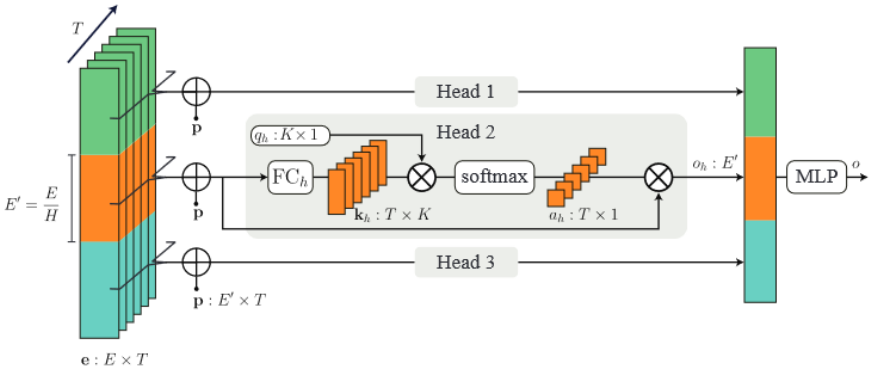


Figure 29: L-TAE architecture

The TAE is a network architecture that employs multi-headed self-attention mechanisms to classify remote sensing time sequences. The channels of the temporal inputs are distributed among several attention heads operating in parallel. Each head extracts highly-specialized temporal features which are in turn concatenated into a single representation. The proposed approach modifies the TAE by distributing the channels of the temporal inputs among several compact attention heads operating in parallel, which allows for more efficient processing of time-series data.

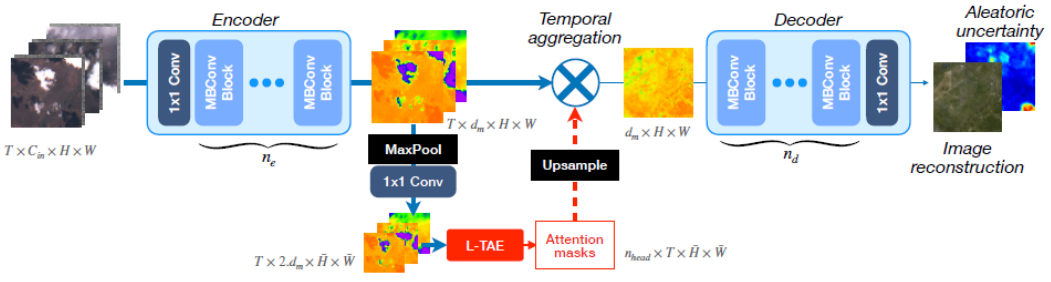


Figure 30: [3] model generator architecture

This architecture is then used in [3] to aggregate the information of temporal images without needing high computational resources. [3] proposes a multi-temporal model using [2] dataset, achieving the best results since now. Moreover, they add the paradigm of uncertainty in the field, being the first time to show how the well-calibrated predicted

uncertainties enable a precise control of the reconstruction quality in the the task of multi-temporal cloud-removal in optical satellite images.

1.3 Decision on thesis alignment

After reviewing the state-of-the-art proposals and the trend of current deep learning architectures, the objectives regarding model design process that align better in the context of this thesis are the following:

- Use SAR and Sentinel2 data in most of the model architectures since it is the most effective way to remove clouds.
- Train some models without SAR data and check the difference of it would add an interesting point to the thesis.
- Try at least one of each type of generative models in the following order: VAE, GAN and DDM.
- Do not use architectures that need temporal data since the scope of the thesis would increase twice at least.

1.4 Academic Datasets

Several public datasets for cloud removal have been proposed over the recent years. In the following section, we will review a few of them and, finally, select those who are more aligned to the thesis and that can be merged.

The first public dataset published, called `RICE`, was described in [25]. Actually, it can be considered as two datasets, since there are two quite different parts: `RICE1` and `RICE2`. While `RICE1` contains 500 pairs of images[5] of 512×512 from Google Earth, `RICE2` contains 450 sets of images from Landsat 8 OLI/TIRS, being each set three images of 512×512 px: a cloudless image and a cloudy one with its mask. Compared to the following datasets, the size of both is relatively small and the data of them is spatio-temporal homogeneous. For that reason and since they are not from Sentinel-2, they have been discarded.

[10] introduces two novel datasets contains nearly 10000 paired Sentinel-2 images of

[5] Each pair contains a cloudy and a cloudless image.

256×256 px drawn from 32270 distinct tiles worldwide. The cloud cover for each cloudy image is between 10% and 30%. It excludes the images with insufficient visible ground upon manual inspection and they restrict the percentage of images of ocean being no more than 10 % of the dataset. The other dataset contains 3130 sets of 3 cloudy images and a cloud-free image of the same region. Since it will only be used a paired images, the latter would only add 3130 pairings compared to the 10000 instances from the first dataset.

[1] also offers paired Sentinel-2 images of 256×256 px. One of the main advantages of this dataset is that it has multi-temporal and multi-spatial paired images so that there is a diversity of the season where the image is taken and the land cover. The procedure of extraction the images involves two uniform random samplings over the landmasses of the Earth and the urban areas across the globe, respectively. The quality sccore module assigns a quality score for each image by considering the likelihood it is affected either by clouds or by shadow. The images are sorted by their amount of poor pixels, and the quality scores are thresholded to determine cloud and shadow masks for each image. However, this information is not obtainable to obtain more specific metrics regarding the features of the images. Hence, an exploratory data analysis of the data must be carried out to know, for example, the common percentage of cloud of cloudy images.

[1] is the most versatile dataset regarding scene distribution, as it covers all regions of the Earth over all meteorological seasons, while most of the other datasets are restricted to fairly small areas, individual countries, or a single continent. Moreover, [1] contains SAR data of the respective patches and scenes, so any more data would need to be downloaded to train the models. A point that must be commented is that SEN12MS-CR has neighboring patches that overlap between 25 and 50 % overlap. However, the training split from the [1] cares about this issue and the validation and testing data has no overlapping regions from any of the other sets.

In addition, this dataset has compatibility with [2], which is a remote sensing data set for multi-modal multi-temporal cloud removal. This dataset is used in [3] and the data needs to be paired in order train a cloud removal image model. Moreover, a discrepancy between both data sets is that SEN12MS-CR-TS has no intersection between adjacent samples. Moreover, SEN12MS-CR-TS consists of 30 time samples for each of the

8,663 patch-wise observations, for every S1 and S2 measurement. As there is no geospatial overlap across splits between both datasets, this one can be combined for training or validation purposes, so once done the pairing, this dataset can be used for training purposes.

Another special point is the choice of using data augmentation to populate the dataset with the existing data but with some changes. More specifically, it has been thought of creating synthetic clouds using techniques, such as perlin noise, to cover or modify some regions of the images. Although it has been used in several models, it has been decided not to use it since not all the bands are equally affected. This point will be deeply explained in **??**

Finally, it has been decided to use SEN12MS-CR as the main dataset with patches from SEN12MS-CR-TS, due to their compatibility. The other datasets have been discarded since data from validation and testing could be in [10], for instance. In 4 it can be seen the main characteristics of the datasets explained and whether its data will be used for this master thesis.

Table 4: Academic datasets and its main features

Dataset	Length	# Patches	Image Size	Source	Bands	SAR	Usage
RICE1 [25]	500	500	512×512	Google Earth	RGB	No	No
RICE2 [25]	450	450	512×512	Landsat 8 OLI/TIRS	RGB	-	No
[10] (one-to-one)	32,270	32,720	256×256	Sentinel-2	13	No	No
[10] (many-to-one)	10,000	10,000	256×256	Sentinel-2	13	No	No
SEN12MS-CR [1]	541,986	122,218	256×256	Sentinel-1 & Sentinel-2	2 + 13	Yes	Yes
SEN12MS-CR-TS [2]	259,890	8,663	256×256	Sentinel-1 & Sentinel-2	2 + 13	Yes	Yes

Part II

Development

2 Exploratory Data Analysis

Once selected the datasets, an exploratory data analysis has been carried out to understand the hidden properties between bands from the images and the images themselves to gather enough information and make the dataset comprehensible so that the design and evaluation of deep learning models can be more straight-forward. Moreover, the Exploratory Data Analysis (EDA) was aimed to provide insights into potential challenges that might be encountered during the development of the model. This stage of the thesis proposed four main challenges:

- Aggregating these datasets into a unified and standardized format is essential to ensure consistency and compatibility during the subsequent stages of the deep learning model development. Although both datasets are meant to be aggregated at some point, a solution to fit the dataset schema into the actual challenge was needed.
- Extracting the properties using various techniques, including image processing algorithms and statistical analysis, without getting away from the main topic and, then, attempting not to create biases while extracting the model metrics.
- Visualizing the data can provide valuable insights but a huge amount of data extracted can be messy and create a high computational demand. Research about high order libraries to visualize the data.
- Understanding the limitations of the datasets, such as noise, class imbalance, or variations in image quality, allowing for the identification of potential biases and the formulation of suitable mitigation strategies.

During the EDA phase, various techniques were employed to extract meaningful information from the dataset. Descriptive statistics were computed to gain a preliminary understanding of the distribution, central tendency, and spread of the data across different bands. This provided an overview of the range and variability of values within each band, allowing for an initial assessment of their significance. By visually inspecting the images, patterns and trends specific to cloud formations must be discerned, aiding in the development of

strategies to tackle cloud removal effectively. Furthermore, feature engineering techniques were explored to derive additional meaningful features from the dataset. This involved transforming, aggregating, or combining existing features to capture important characteristics of the data. After each information was discovered, this steps were not considered to be sequential but concurrent phases to incrementally learn and study the data. Therefore, the following subsections must be interpreted as chapters of a story of the analysis rather than separated and independent subsections.

2.1 Properties extraction and visualization of the first dataset

At first, creating a data exploration of `SEN12MS-CR` [1] was perfect as a first stemp since it is the most aligned dataset. Once extracted and analyzed the data, merging it to the other data was thought to be more straight-forward to understand. The feature extraction of the dataset last $38h$ of computation.

As a general overview of the dataset, as mentioned before, the data is already split into training, validation and testing sets to compare between third-party models, as it can be seen in 31. For that reason, any new data can only be added to the training or validation sets.

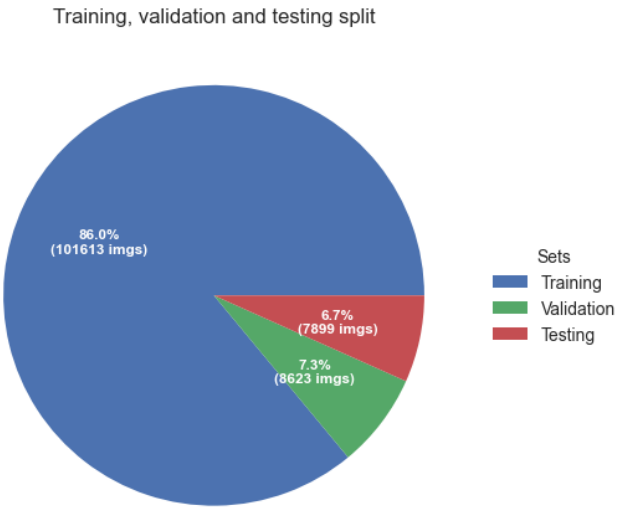


Figure 31: Training-validation-testing split sets in SEN12MS-CR.

An important step to take at this stage before processing, extracting features and visualize them was to take care of the pixel intensities in order to visualize and extract the information properly. The imagery values of each band were ranged from 0 to 10000.

Therefore, the pixels need to be normalized and scaled for a proper processing and visualization of the data. This led to a challenge since the comparison or visualization of the properties extracted would be affected. Three normalization methods were thought:

- MinMax Scaling, also known as feature scaling, is a method that linearly transforms the pixel values of an image to a specific range, typically between 0 and 1.

$$\text{Normalized value} = \frac{\text{Pixel value} - \text{Min Value}}{\text{Max value} - \text{Min Value}}$$

  Although this preserves the distribution of the pixel intensities of images for processing, this method is not proper for visualization since outliers can affect the visualization of the data, as it can be seen in 33. The max value must be the same for all the images (1000).

- Linear Stretch method. The linear stretch method is used to enhance the contrast of an image by stretching the pixel values across the a given available range. If the pixel surpasses the 98^{th} percentile, it will be 1 (or the maximum of the scaled vector). If the pixel value is under the 2^{nd}, it will be 0 (or the minimum of the scaled vector). For the other pixel values, it performs the scaling operation by using interpolation, being the maximum value the 98^{th} percentile and the 2^{nd} the minimum.

$$\text{Normalized value} = \frac{\text{Ranged pixel value} - 2^{\text{nd}}\text{ percentile}}{98^{\text{nd}}\text{ percentile} - 2^{\text{nd}}\text{ percentile}}$$

- Sigmoid after linear stretch method. The sigmoid after linear stretch method applies a sigmoid function to the linearly stretched pixel values to further enhance the contrast. In this formula, the slope k is a parameter that controls the shape and position of the sigmoid function. The sigmoid function maps the stretched pixel values to a range between 0 and 1, enhancing the contrast in the process. By increasing the slope, the range of values can be compressed, so they get more blurred and with less contrast. This can be detected by applying the laplacian variance in the image. The slower the value, the higher its blurriness, as it can be seen in 32 and 34.

$$\text{Normalized value} = \frac{1}{1 + e^{-k \cdot (\text{LinearStretch(Pixel value)})}}$$

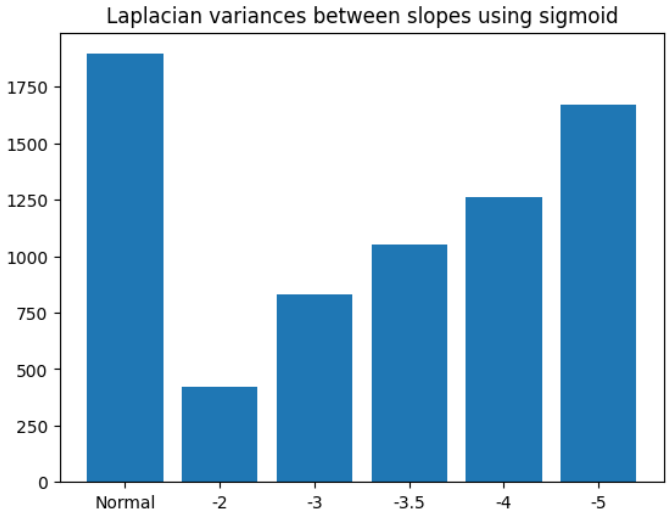


Figure 32: Laplacian variance depending on the slope used in sigmoid scaling method.

The following pictures show how the images behave after treating each scaling method:

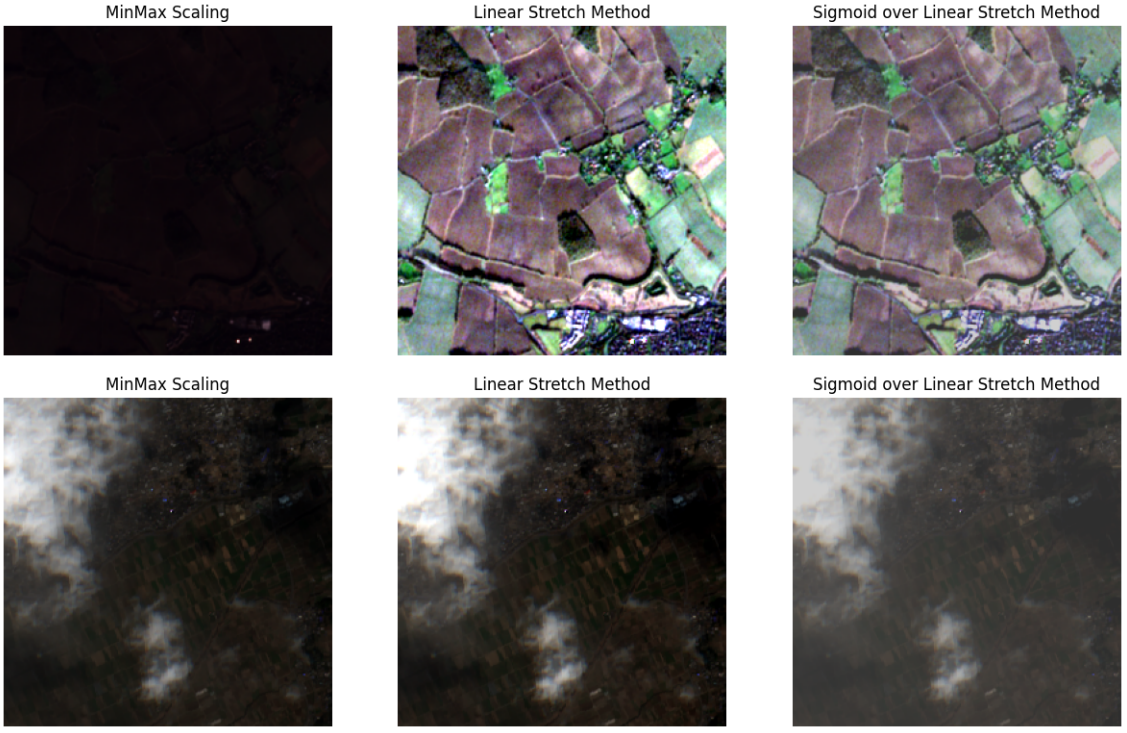


Figure 33: Scaling methods

Regarding data processing for training, the best method to use is the MinMax feature scaling, since it preserves the intensity distribution and range of the values although its image visualization is poor.

Regarding data visualization and general data extraction, although it seems in 33 that the best normalization to apply is sigmoid after the linear stretch method, we can see in 34 that this would affect the study of the properties since the data would be so compressed that the visual results could lead to confusion. Therefore, the best method to carry out while visualization is the linear stretch method, since there are some pixel intensities that are outliers due to the noise or other obstacles while capturing the images.

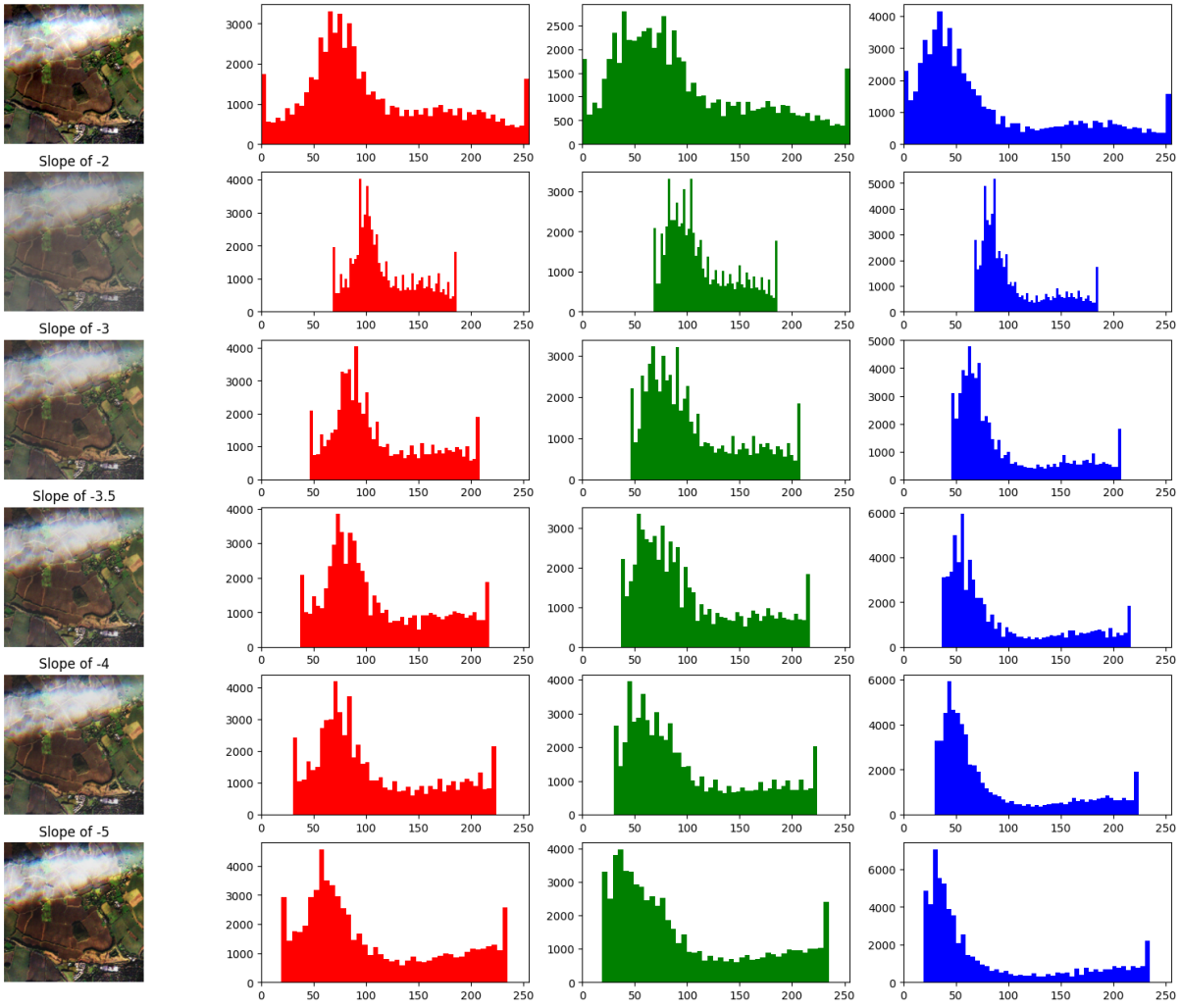


Figure 34: Sigmoid-scaled images with their RGB bands in different slopes.

The properties compared in the first dataset are classified depending on how they were extracted:

1. Band properties of each image. Exploring properties related to individual bands helped to compare two bands given an image or different samples of images given these band properties. Since the data process changes the identification of the bands,

in table 16 is detailed the translation of the bandwidths.

- Descriptive statistics such as the mean, median, std and percentiles to compute statistics for each band, providing insights into the distribution, central tendency, and spread of data values.
- Contrast. Analyzing the range and variation in pixel intensities within each band to understand the degree of contrast present. There are several ways to calculate contrast, the most popular ways are:

$$\text{Traditional contrast} = \frac{\text{std(band)}}{\text{mean(band)}}$$

$$\text{Michelson contrast} = \frac{\text{Maximum value} - \text{Minimum value}}{\text{Maximum value} + \text{Minimum value}}$$

- Blur. Assessing the level of blurriness in the images, indicating the sharpness of details captured. To extract the blurriness of the images, it has extracted the variance of the images after applying the laplacian filter.
- Kernel Density Estimation (KDE) helped to visualize the probability density function of each band, aiding in the identification of underlying patterns or modes.

2. True Color Image properties of each image. Although not all the bands captured from sentinel2 are visible in human spectra, comparing properties of the RGB bands could help us humans to understand the difference in the image captures.

   - Saturation. Examining the saturation level of colors in the TCI, which indicates the vividness or intensity of hues. To get the saturation it has been changed from RGB to HSV color space.
   - Temperature. Assessing the temperature or color balance of the TCI, providing information about the overall warmth or coolness of the image. To capture the temperature the images have been converted from RGB to CIE 1931 XYZ color space.

3. The cloud coverage of each image was captured using the library

   - Percentage of each image. This quantifies the extent to which each image is covered by clouds, enabling the assessment of potential challenges posed by

cloud interference.

- The MSE between cloud-free reference images and their corresponding cloudy versions was calculated allowing us for the evaluation of image quality degradation due to cloud cover.

4. An exploration of the indices showed in 1 helped to reveal additional characteristics or features within the images and differences between the season and regions of interest.

The analysis was ideal as a first step but hard to visualize due to the high amount of properties collected and its diversity of nature. Visualizations played a crucial role in revealing the inherent structure and characteristics of the dataset and for that reason, a first glance of the data should be done interactively. Since the variability of the cases where to compare the dataset and how extremely hard was to visualize correctly and effectively all the properties data extracted, the best decision to make was creating a web app to help the user experience of the EDA. To sum up, the goal of the app is to give a first glance of the dataset and find any imbalance.

The app is powered by Streamlit. The library is a powerful and user-friendly python library that simplifies the process of building interactive web applications for data analysis and machine learning. It provides a straightforward way to create custom web interfaces directly from Python scripts, enabling users to easily visualize and explore data, create interactive dashboards, and share their work with others.

Streamlit works by writing Python code that defines the various components of the web application, such as widgets, plots, and text. These components are then rendered in a web browser, creating an interactive user interface. The library takes care of handling the communication between the Python backend and the frontend, making it seamless for users to interact with the application. However, since the amount of data extracted from the images was huge, the efficiency of data processing while rendering the page decreases.

First of all, a sample of 100 images was extracted to visualize how is the variety of cloudy images. From 60, we can state that there are type of cloudy images:

- Overcast images. These images show complete cloud cover, obscuring the land below.

- Mostly cloud, covering more than 75 % of the land, allowing for some visibility but still showing significant cloud cover
- Partly cloud. A cloud that covers a big region of the image, between 25 % and 75 % of the land.
- Stratus clouds. Low, gray clouds covering the land like a blanket, often giving the image a dull appearance.
- Cumulus clouds. Puffy, small and white clouds dispersed over the image.
- Altostratus clouds. Clouds that there are in a high altitude and create a shadow in the land.
- Apparently clear. These are tricky cases where the cloud detection model might not be correct to identify as cloudless images. This could either mean a clear sky or a false negative from the model.

An ambiguity found in the dataset is that some data is not in any of the three set partitions, this could be due to the quality of them or the land cover changes in the region of interest, as it can be seen in *Figure 61*. Therefore, they were not included in the training set.

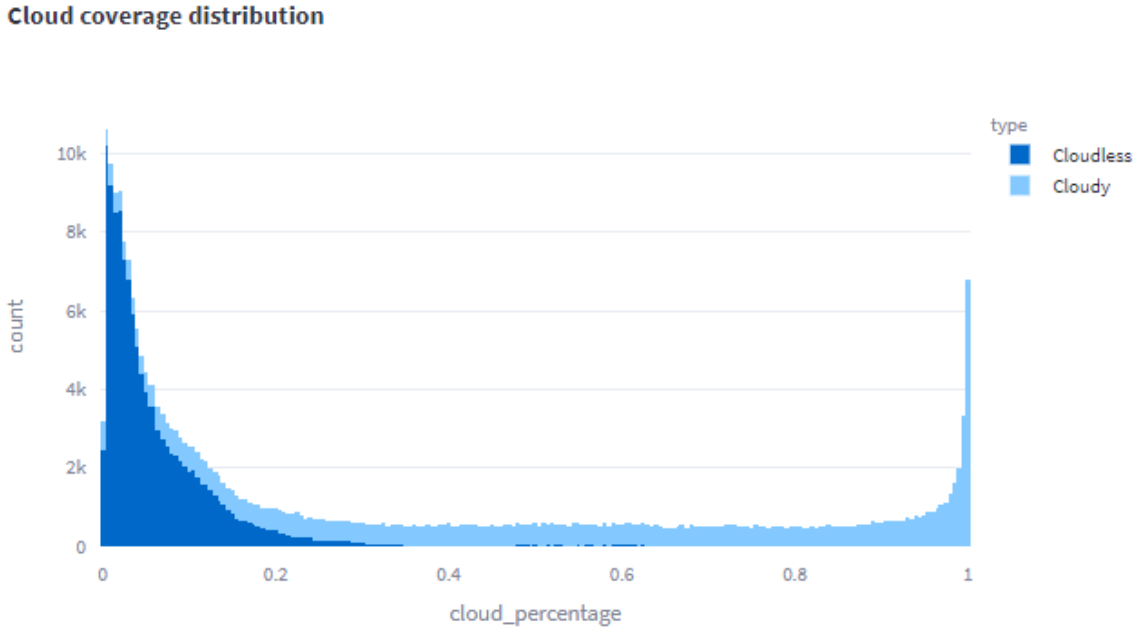


Figure 35: Distribution of the dataset regarding the cloud cover percentage area

As figure 35 shows, the dataset has a uniform constant distribution of cloudy images regarding its cloud covering area percentage without considering the peak of totally covered images. Therefore, the models trained would be less likely to remove clouds from a spcecific range or to be in a state of *mode collapse*. It has been searched for those cloudy images with low cloud covering area. The problem is because of the blurriness of the images, as it can be seen in figures 62, 63 and the scatter plot of cloudy images 36. However, since there is only 1324 blurred images[6] and its `SAR` data is ok, it might be great to have them to allow the model also turn the noise into representative images but also to replicate former models mentioned in *Section 1.2*.

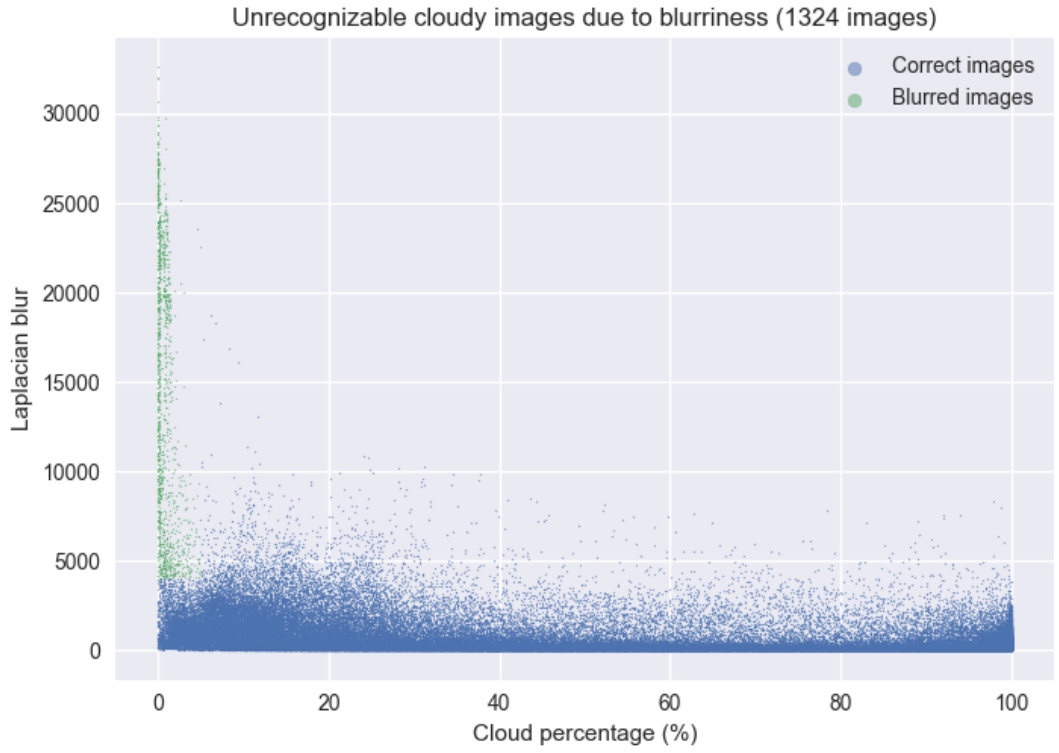


Figure 36: Sample of images labeled as cloudy but with low cloud cover percentage area.

Regarding cloudless images, it can be seen in *Figure 35* that they follow a heavy tail distribution, meaning that:

a) our cloud detector model detects more false positives images

b) there is a misleading tagging process in the dateset

[6]They represent less than a 1 % of the total data.

Moreover, some cloudless images have a high cloud area percentage. In this case, it is due to the haze, which our cloud detector identify these pixels as cloud but the acadamic dataset did not tag them as that, as it can be seen in *Figure 64*.

That having been said, it is more than intuitive that blurriness affect the processing and evaluation of the dataset, although it is itself a characteristic of the dataset. In 62 it can be seen a sample of blurred images to demonstrate its gravity when it happens. A grosso modo, the blurriness in S2 can be caused [7] by

- atmospheric conditions, such as haze, aerosols, and clouds itself, since they absorb light reducing their clarity.
- satellite motion, if the satellite or the objects on the ground are moving rapidly during image capture, this can also be due to an atmospheric condition or mechanical vibrations.
- data processing. For example, some occurrences of yellow pixels over extra brights clouds have been identified when looking at the true color image visualization.

This effects do not only cause blurriness to the images but also particular changes in their saturation and temperature, as it can be seen in *Figures 37 and 38*.

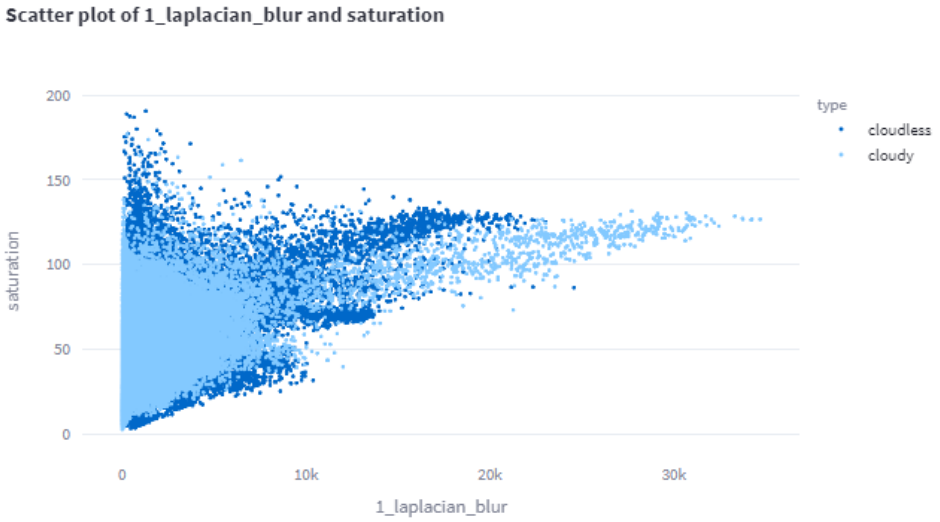


Figure 37: Scatter plot of the laplacian variance in B2 and the saturation of the true color image.

[7] All these errors and more are listed in [26]

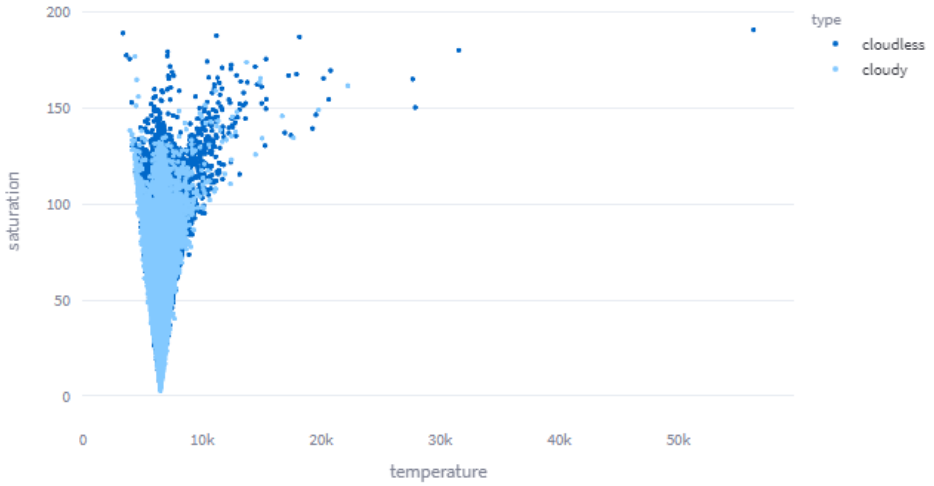


Figure 38: Scatter plot of the temperature and the saturation of the true color image.

Moreover, Sentinel-2 includes a thermal infrared band (B10) that can be used to measure land surface temperature. High-temperature variations on the Earth's surface can lead to variations in thermal emissions, which may affect the quality of images in this band. We can see the difference of the bluriness in B1 and B10 regarding the saturation of the true color image checking *Figures 37 and 39*.

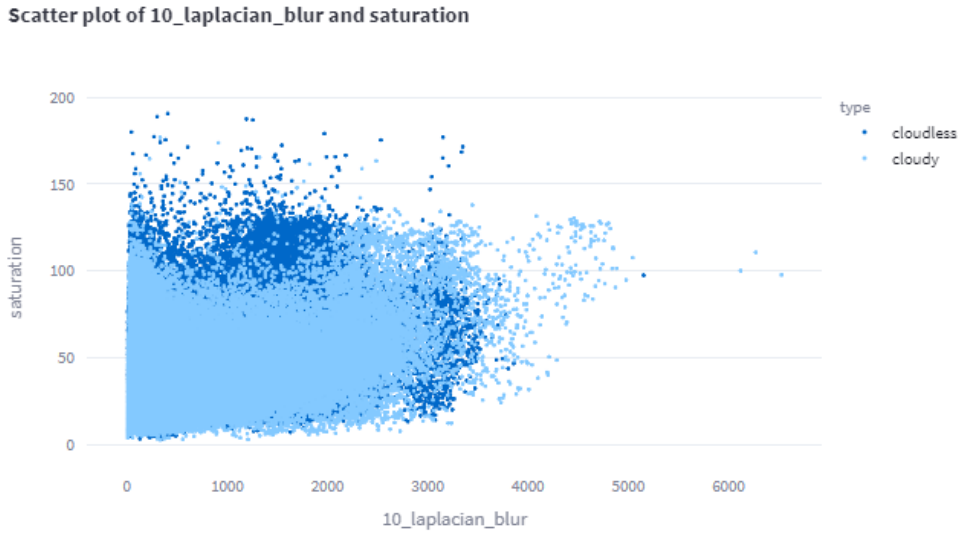


Figure 39: Scatter plot of the laplacian variance in B10 and the saturation of the true color image.

Taking that into account, it has been decided how the B10 is affected by clouds comparing it with the other bands. First of all, it has been demonstrated that the bands

are not affected equally to the clouds 40, being the true color images[8], B8[9] and B10. Therefore, it has been decided not to employ perlin noise to create synthetic images with clouds, since this would lead to a bias in generating images.

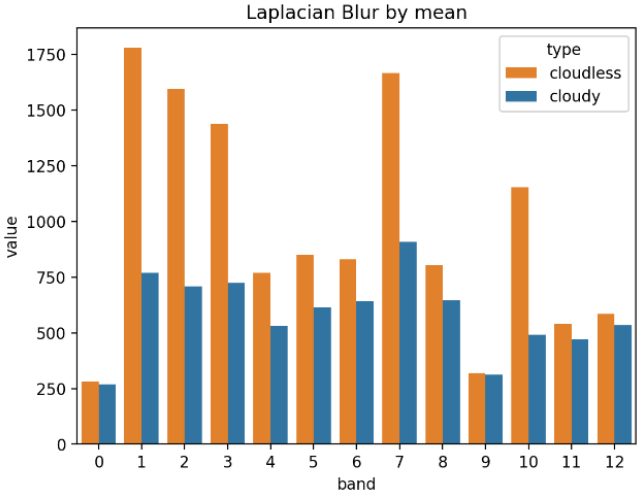


Figure 40: Bar plot of the mean value in each band of the laplacian variance.

Although this similarity in the TCI bands and B10, the blurriness does not behave equally regarding other properties from the image, e.g. the contrast, as it can be seen in the comparison of figure 66.

Moreover, in figure 41 it is clear that there is a high correlation between the region affected by clouds and the values of B10, meaning that cloud regions cause high values of B10. However, not otherwise, as it shows 65.

[8]B2, B2 and B3, showed as 1, 2 and 3 in 40 respectively
[9]Showed as 7 in 40.

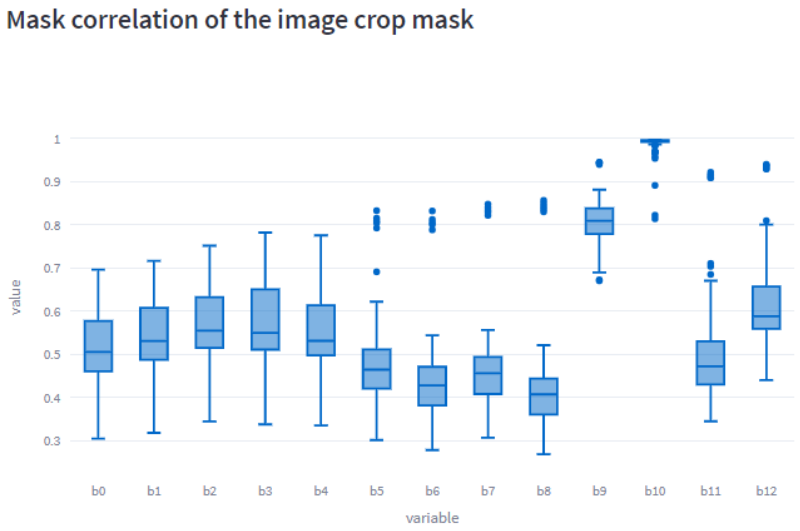


Figure 41: Correlation of the region with clouds and the values of the band, by mean from each image.

The cloudy crop correlation did lead to ask how B10 behaves in cloudless and cloudy image respectively. By visualizing the kernel density estimation of B10 42 and the other ones[10], it is totally clear that B10 is the most corrupted and affected bandwidth by the presence of clouds. Therefore, a single metric to check the difference between the B10 of synthetic data and cloudless data could be helpful to discern the effectiveness of the models.

[10]For instance, figure 67

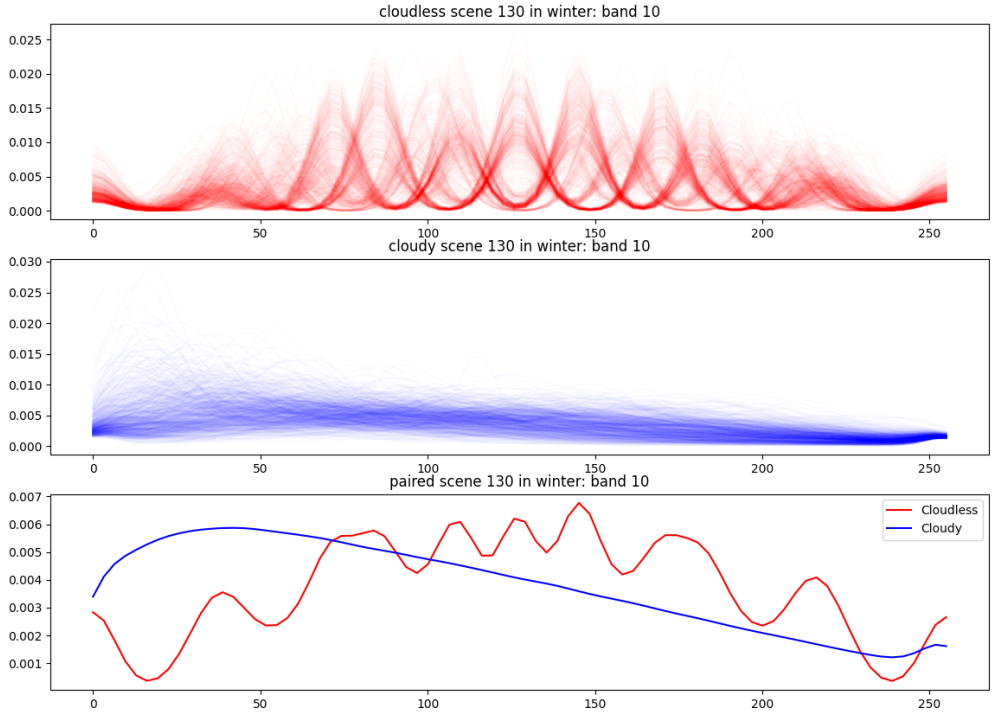


Figure 42: Kernel density estimation of B10 in cloudy and cloudless the patches of a specific scene (130)

Finally, it has been decided to include the blurry images in the training set to make the model robust against such real-world imperfections. Features to capture blurriness, saturation, and temperature could be engineered and fed into the model to improve its predictive power. For example, make the blurriness, the contrast and the temperature be in concordance from the target set. Given the high sensitivity of B10 to cloud presence, incorporating a metric that compares the B10 values between synthetic and cloudless data, as suggested, should be useful to evaluate the model's performance specifically in terms of its handling of cloud-related features. Given that different bands react differently to clouds, a multi-modal neural network could be beneficial. Separate branches of the network could specialize in learning features from different bands and then combine them for final predictions.

2.2 Data merge strategy

As it has been said before, part of SEN12MS-CR-TS [2] will be included in the original dataset SEN12MS-CR [1]. However, since there is no paired images in the dataset, some adjustments should be carried out. Overall, there is a 55 % of cloudy images. Hence, inherently, a pairing of the dataset could be done just by choosing the 45 % cloudless images and the ones left. However, a bias would be produced since we would be inject 14 % pairings of the same patch.

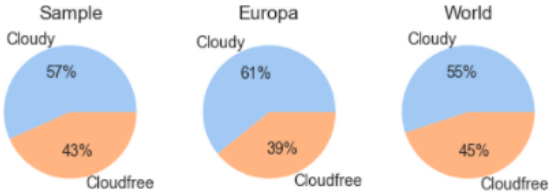


Figure 43: Cloudy images percentage in SEN12MS-CR-TS.

To know better which kind of pairing I was dealing with, a visualization of a sample of 10 patches from any time interval was done to check have a simple overview of a patch given the 30 intervals, as it is depicted in figure 68.

To classify which images are cloudy and cloudless, it has been defined a limit of the mean of the cloud percentage area in [1] from the former exploratory data analysis. This is shown in figure 44, given the patch shown in figure 68.

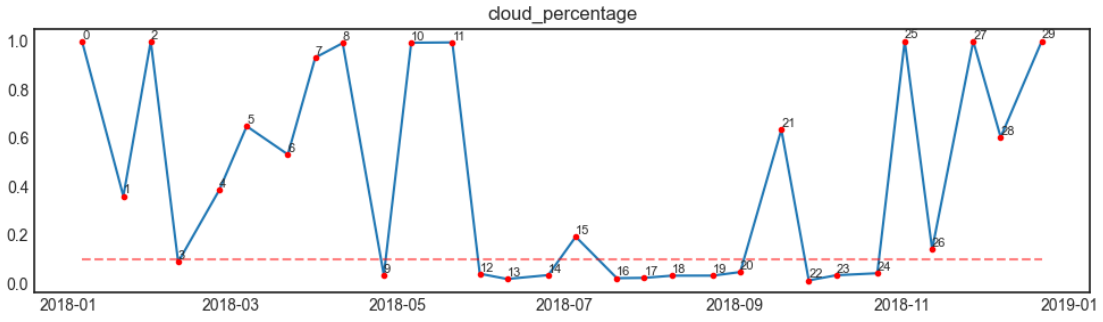


Figure 44: Patch exploration in all intervals.

It is notable to mention how similar are the classification from the patch shown and the cloud percentage distribution of the entire dataset, as it is depicted in 45. This is due to the range of image capture of each time interval, since no time instant[11] from any patch

[11]Time instant and time interval are used indistinctively in that case.

is more than 6 days away from any other patch given the time instant id.

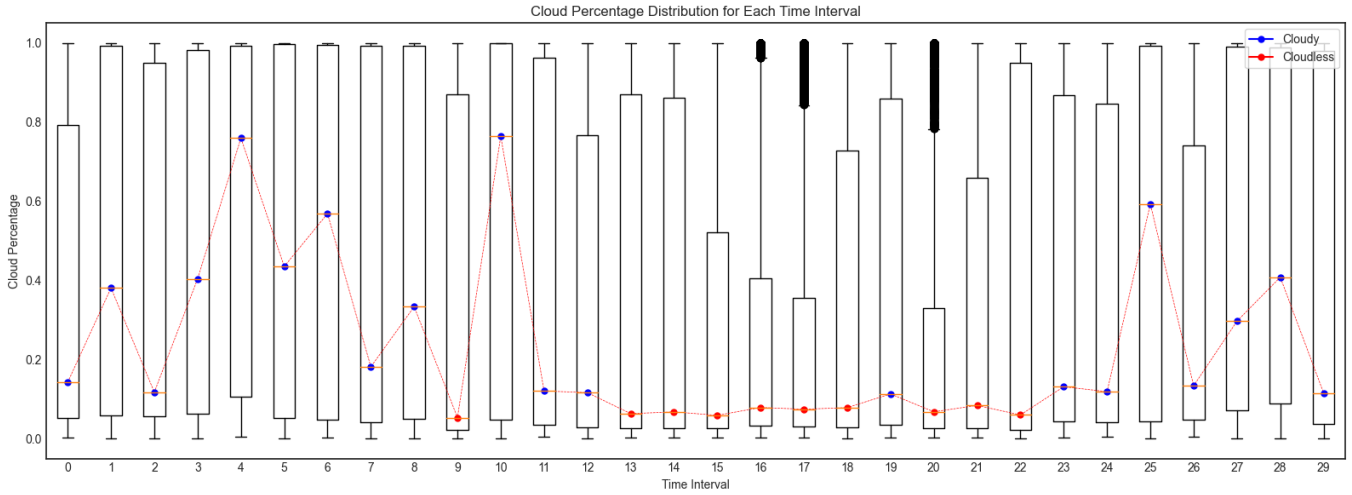


Figure 45: Cloud percentage distribution in all intervals and its possible classification given the median.

Additionally, each band and index has been compared to the cloud percentage of the given time instance to check if some anomalies were found. In fact, it has been discovered that when an image got blurred, the indices `MSI`, `NDWI` and `NDSI` are more affected than the others. Actualy, all these indices are created by bandwidths B3 and B8 and they are about the presence of moisture, water, snow respectively. Therefore, the anomaly found in the B3 and B8 can be just depicting a storm than showing a discordancy between the output of the cloud model and the image itself. This phenomenon has also been aligned with the behaviour of SEN12MS-CR, although was not presence in all the imagery.

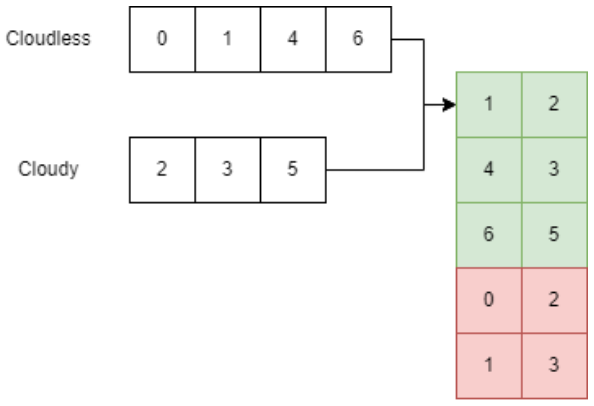


Figure 46: Pairing algorithm.

Once analysed the dataset and found that they have more or less the same qualities and characteristics from base dataset, it has been developed an algorithm to pair the images.

Considering that each patch has two vectors, one at a time when the images are cloudy and another when they are cloud-free, a resultant vector is created from the possible pairs ordered by the minimum distance of these moments in time. In this way, the modification of the terrain will be minimal. Moreover, instead of doing of every vector, it has been split the patch into the seasons, so the resulting number of pairings found is `30097`. Hence, the data grows up to $(101613 + 30097) = 131710$ pairings, increasing a 29.16% more. Nevertheless, since it is added some bias to the dataset, a comparison between models trained with only SEN12MS-CR and with the data augmentation should be carried out.

3 Model training

3.1 Technologies

The choice of tools, frameworks, and libraries plays a critical role in facilitating research, ensuring reproducibility, optimizing model performance and a great user experience for the metrics visualization. This section delves into the core technological stack employed in the master's thesis for the model training pipeline. [27]

Regarding the architecture design and backend for the training, it has been decided to be `PyTorch`. It is an open-source deep learning platform that offers a flexible way to define and train neural networks. Its flexible way to define and train neural network, its idiomatically code written in python and the experience of mine in this library were the main reasons why `PyTorch` was chosen. Moreover, the library carries with a lot of internal and third-party libraries to help developers to build more advanced and deep techniques. In our case, `Torchvision`, offered a collection of model architectures, and image transformations for computer vision that make easier to work with image data by providing this essential utilities. Other third party libraries were used such as `torch-sssim` to build some specific losses and metrics. Another library in the `PyTorch` ecosystem is `diffusers`, which integrates `PyTorch` as a backend for designing and developing DDM.

Managing the machine learning lifecycle and ensuring that the experiments are not just cutting-edge but also traceable and reproducible were also objectives to take into account the runs visualization. `Tensorboard` was the first library thought. Nevertheless, although it is enough to see the results of the current run, it does not inherent explicitly the reproduction of models and an efficient interface. Otherwise, `MLFlow` includes tools for tracking experiments, packaging code into reproducible runs, and sharing and deploying models, so, finally, it became the one used for tracking the model behaviour and manage the models learning lifecycle.

Other libraries were used such as `Scikit-image`, which is a collection of algorithms for image processing built on top of the Python scientific stack. It's an extension of `Scikit-learn`, a popular machine learning library in Python. `Scikit-image` provides versatile tools for image processing and is interoperable with other Python scientific libraries.

So as to save the configuration of the models and a more automatic way to gridsearch the models and hyper-parameters of the models, it has been created a library of training, which by reading a configuration file in `yaml`, it gets the parameters of the models, the information of the dataset and their properties, and the settings of the hyperparameters.

The training runs and the experiments have been executed in a machine. The machine specifications are shown in the following table 5.

Although this part was a bit challenging, the most difficult part was stabilizing the training. Due to the experience of large batches in GANs, it has been preferred not to use batches[12]. In that way, the training was easy to see when the discriminator outdoes and when fails to discern. In this regard, `MLFlow` proved instrumental worthwhile in discerning the performance of the discriminator throughout each turn of the loop. To achieve this, metrics such as the fake score and the real score were generated both before and after discriminator training. This approach allowed for a clear visualization of the discriminator's progress, showing the gains and losses at each batch iteration.

The real score, sometimes referred to as the "realness score" or "real data score," is the output of the discriminator when it evaluates real data samples from the training dataset. Conversely, the fake score represents the output of the discriminator when it assesses generated data samples produced by the generator. It indicates how convincingly the discriminator believes that the input is fake or generated data. The goal is to achieve a balance where the generator produces data that is indistinguishable from real data, resulting in both high real and fake scores, and making it challenging for the discriminator to differentiate between real and fake samples.

Although keeping a small batch size helped sometimes, when the batch size is that small, the learning from the inference done increases. To prevent overfitting in the network during the initial epoch and to manage the magnitude of the weights, the learning rate of the optimizers, particularly that of the discriminator, was significantly reduced. This precaution was taken because, at the beginning, the discriminator can easily distinguish between real and fake data samples. Although the learning rate has been reduced dramatically, the

[12]The batch size has been set to 1.

weight decay is higher than generator's, since it needs some way to outdo it.

Finally, to assist the generative model in its initial iterations, it is added the MSE to the loss function. This addition enabled the generative model to orient itself, enhance training stability, and maintain a clear focus on addressing the core problem.

Table 5: Specifications of the machine

Attribute	Value
CPU	32 Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz
RAM	512 GB
GPU	2 NVIDIA GeForce RTX 3090

3.2 Losses and metrics

As it has been detailed during all the memory of the master's thesis that the process of training a model essentially revolves around the optimization of a particular function. Essentially, the suitable loss function is not only hard to come up but also it might be impossible since any function has their payoff. In this section, a review of the metrics and losses will be carried out to understand their advantages and disadvantages when applying them. However, those function loss described before such as the GAN (equation 1), VAE (equation 4) or DDM (equation 5) losses.

The MSE Mean Squarred Error loss computes the mean of the squared differences between the target values and the predicted values. It is the most common used when using autoencoders. It is given by

$$MSE(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

One of the main advantages of this loss is that squaring the differences magnifies the errors, making the model more sensitive to data points that are far from the predicted values. It is also often used in regression problems and highly sensitive to outliers due to the squarring term. However, when it comes to generative image models, this loss tend to produce some noise in the synthetic images.

The L1 loss or MAE Mean Absolute Error loss computes the mean of the absolute differences between the target values and the predicted values. It is used when aiming to

reduce the noise from the output image. It is given by

$$L1(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

It is less sensitive to outliers compared to MSE loss because it doesn't square the differences.

The SAM is a widely used algorithm for comparing spectral similarity between two spectra. It calculates the angle between two spectra, treating them as vectors in a space with as many dimensions as there are bands. The smaller the angle, the more similar the two spectra.

$$SAM(y, \hat{y}) = \arccos(\frac{y \times \hat{y}}{||y|| \cdot ||\hat{y}||})$$

A loss or metric used only in cloud removal state of the art is the Cloud-Adaptive Regularized Loss (CARL). It is a custom loss function, typically used when there is the cloud mask. This loss function is constructed to handle areas in satellite images that are clear or cloudy differently. This metric focuses on minimizing the differences between the clear regions of the input image and the synthetic image. For that reason, for clear regions, it is used the MAE between the output and the input image, and, for cloudy images, it is used the MAE between the output and the target image. Additionally, there's a penalty term added to the loss which is simply the mean absolute error between the predicted values and the true target values, weighted by a factor. Normally, $w = 1$.

$$\begin{aligned} CARL(x, y, \hat{y}, mask_{cloud}) = \quad & \frac{\sum_{i=1}^{n} (A - mask_{cloud}) \times |(x_i - \hat{y}_i)|}{n} + \\ & + \frac{\sum_{i=1}^{n} mask_{cloud} \times |(y_i - \hat{y}_i)|}{n} + \\ & + w \cdot \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n} \end{aligned}$$

$$\text{being} \quad A = \begin{pmatrix} 1 & 1 & ... & 1 \\ 1 & 1 & ... & 1 \\ ... & ... & ... & ... \\ 1 & 1 & ... & 1 \end{pmatrix}$$

In essence, the CARL loss aims to:

- Penalize large differences in clear areas between predictions and cloudy inputs.
- Penalize large differences in cloudy or shadowed areas between predictions and true

values.

- Apply an overall penalty for differences between predictions and true values across the entire image.

Normally, no classifications loss functions would be used in generative models. Although, since GANs, more specifically, discriminators, need to discern between real and fake images, it is notable to mention the Binary Cross Entropy (BCE) loss, whcih, actually, is regularly used in binary classification. A grosso modo BCE calculates the dissimilarity between the true distribution of labels and the predicted probability distribution. In other words, it doesn't just look at the raw value of the prediction but considers it as a probability distribution.

$$BCE(x, y) = -(y \cdot log(x) + (1 - y) \cdot log(1 - x))$$

A lower BCE indicates that the predicted probabilities are closer to the true labels, while a higher BCE suggests a larger discrepancy. Since it is measuring probabilities, it is compulsary to output a value between 0 and 1.

When referring to image processing and computer vision, there are metrics to measure the similarity between two images. The PSNR is a metric commonly used to measure the quality of reconstructed or compressed images in comparison to a reference image. It is particularly popular in the fields of image and video compression, as it gives an objective measure of the reconstruction quality. The PSNR is derived from the MSE between the reference and the reconstructed image.

$$PSNR(y, \hat{y}) = 10 \cdot log_{10} \frac{(MAX_I^2)}{MSE(y, \hat{y})}$$

where MAX_I is the maximum possible pixel value of the image, in our case, 1. The PSNR is usually expressed in decibels (dB). A higher PSNR indicates better reconstruction quality, implying that the reconstructed image is closer to the reference image. Conversely, a lower PSNR suggests that the image quality has deteriorated. It's worth noting that while PSNR is a straightforward and widely-used metric, it doesn't always align perfectly with human perception. In some cases, images with higher PSNR may appear of lower quality to human viewers, and vice-versa.

Another metric used in image processing is the SSIM. It was developed to provide a more intuitive quality metric than traditional methods like MSE or PSNR. While MSE and PSNR are based on pixel-wise differences, SSIM considers changes in structural information, luminance, and texture.

The SSIM index is defined for two windows $x \in \text{img}_1$ and $y \in \text{img}_1$. The formula for SSIM is given by:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

where:

- μ_x and μ_y are the average values for windows x and y respectively.
- σ_x^2 and σ_y^2 are the variances of windows x and y respectively.
- $\sigma_x y$ is the covariance between both windows.
- C_1 and C_2 are constants to stabilize the division against weak denominators. They are typically set as $C_1 = (k_1 L)^2$ and $C_2 = (k_2 L)^2$ where L is the range of the pixel values. In our case, $L = 1$. On the other hand, k_1 and k_2 are small constants, which are commonly set to $k_1 = 0.01$ and $k_2 = 0.03$.

The SSIM index can be computed for every window in the image, and the results can be averaged to produce a single SSIM value for the entire image. If the valuyes of the metric is 1, it indicates that the two images are identical, being the upper bound of the index. Values further than 1 indicate less similarity.

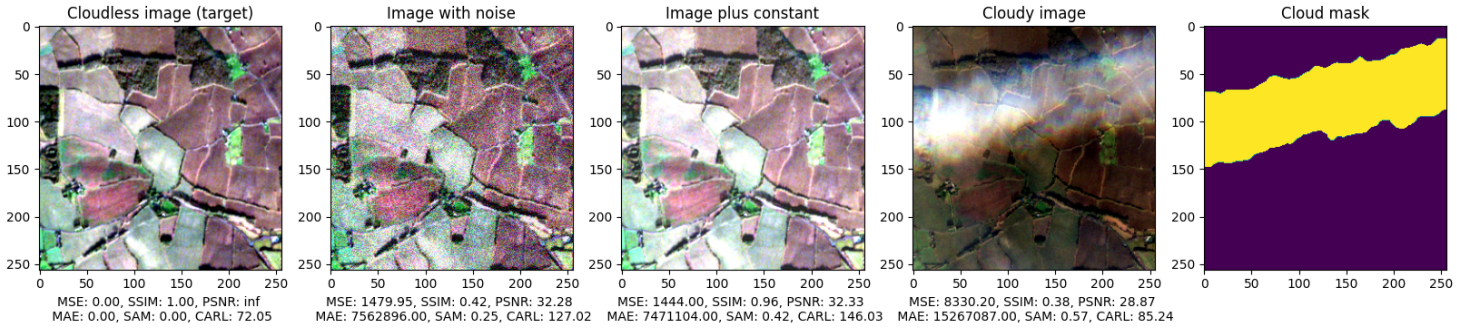


Figure 47: Comparison of all the metrics ($w = 1$).

To sum up, it has been done a comparison of these metrics between a cloudy and a cloudless image. To help the comparison, it has modified the cloudless image to create two possible outputs from the model: one with noise and one adding a constant. In that case, it is notable that SSIM puts more attention to the structural information and MSE, SAM or CARL check pixel-wise differences. It is worth noting how CARL benefits cloudy images than those who are modified. That is because this function depends on the cloud mask and how different are the cloudless and the cloudy image. Taking that into account, CARL needs to be used more like a loss function, where the model would try to optimize not only the cloud removal but also the minimum differences in clear regions than a metric to check how good is the model trying to remove the clouds without considering its consequences. Nevertheless, we must take into consideration that the intensity values from the pixels are from 0 to 256, if they were normalized, CARL results would be between $[0, (2 + w)]$.

Table 6: Comparison of all the metrics.

Image	MSE	SSIM	PSNR	MAE	SAM	CARL(w=1)
Cloudless (target)	0	1	inf	0	0	72.05
Noisy image	0.02	0.42	-31.70	7563250	0.25	126.92
Image plus constant	0.02	0.96	-31.60	7471104	0.42	146.03
Cloudy image	0.13	0.38	-38.52	15267087	0.57	85.24

3.3 Models architecture

Regarding the models, we can classify each model trained by its main idea behind their design:

- Simple CNN. Due to its simplicity compared to the other architectures, its primary purpose was to determine the level of challenge posed by the dataset.
- Residual neural networks. Inspired by [7], it has been developed residual blocks to fit in CNN architectures to check the behaviour of the input data and the possible lack of SAR data.
- Autoencoders not only reduces the dimensions of the input data but also it has a new kind of layers called *Convolutional Transposed* layers. This models were designed to know how dimensionality reduction affects to the data.
- VAE. While traditional autoencoders might not fully capture the diverse nuances of the data, contrasting them with likelihood-based models can provide valuable insights

into their strengths and limitations.

- GAN have the potential to outperform standard CNNs, primarily due to the integration of the discriminator component. The primary focus here will be on preventing mode collapse and ensuring the stability of both the generator and discriminator networks within the GAN architecture.

- Finally, a compromising DDMs will be explained.

3.3.1 Convolutional Neural Networks

The training of the simple CNN was the initial step in assessing the dataset's complexity. This basic CNN architecture consisted of eight convolutional layers, each followed by a batch normalization layer and the Rectified Lineal Unit (ReLU) activation function.

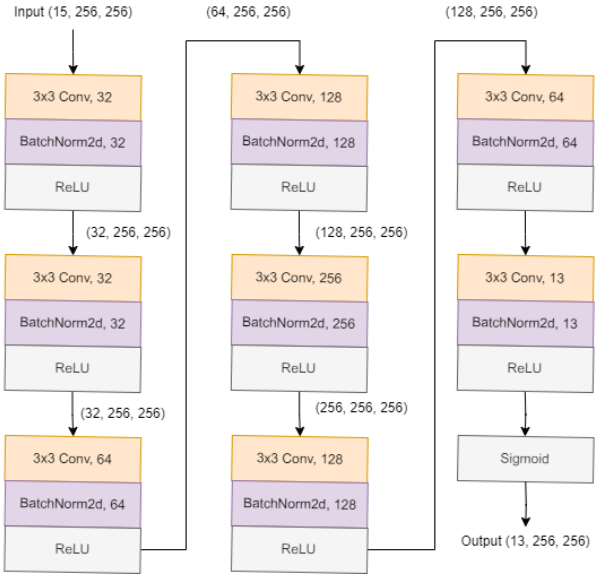


Figure 48: Carla model

The purpose of this straightforward CNN, called `Carla`, was to provide a baseline understanding of how challenging it was to learn from the dataset with only `SEN12MS-CR`. The model has been trained using MSE loss and with a max-min scaling. By analyzing the performance, we could gauge the inherent complexity of the data and make informed decisions about whether more advanced architectures were necessary. By training this model, it has been demonstrated that small models cannot adequately address the complexity of this problem. This topic will be further discussed in the experiments and results section.

To push forward with our study in generative models and building on the work of [7], it has been developed a model which uses similarly residual blocks of stride 1 to capture the behaviour of SAR data, as it shows the Figure 49.

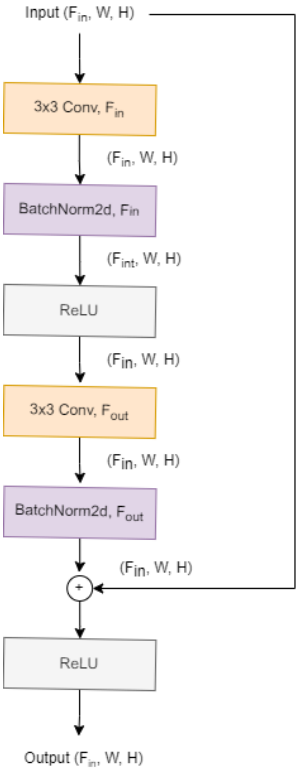


Figure 49: Residual block from `Regina`

What significantly differs between the current model and the inspired model is the number of parameters, which has been reduced to 50% of the referenced model. Firstly, to reduce parameter dimensionality and prevent gradient-related issues, the number of channels was systematically reduced. While [7] always uses 256 filters in each convolution, the filters of the convolutional layers in `Regina` are added incrementally and then, decrementally. Moreover, the number of blocks was decreased from 16 to 10. To compensate this, for this omission and enhance the model's performance, a sigmoid function was introduced in the final layer to constrain the pixel values within a specific range.

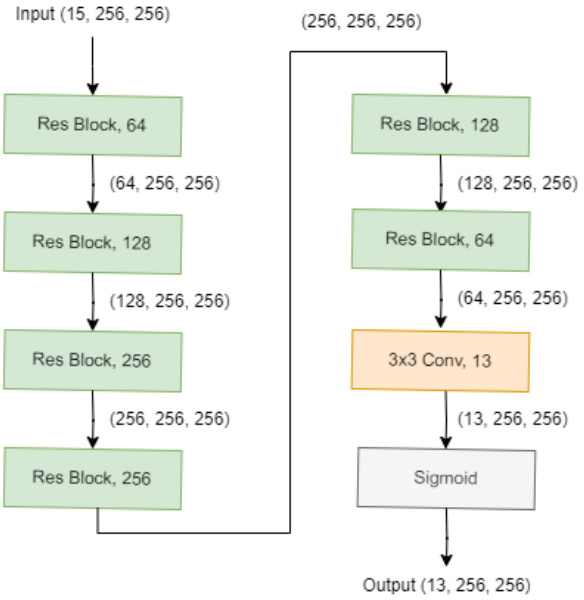


Figure 50: Regina architecture

Since the use of SAR data has also been a prominent topic in the state of the art and to assess whether the inclusion of augmented data enhances the model's performance, multiple training sessions of `Regina` were conducted, comparing sessions with and without augmented data.

3.3.2 Autoencoders

After creating residual models with skip connections, the introduction of autoencoders was proposed to explore how their training and results compared to other models. Unlike the previous models designed, these autoencoders incorporated the use of transposed convolutional layers, which means they generated a kernel based on an input value, as opposed to assigning a value based on a kernel.

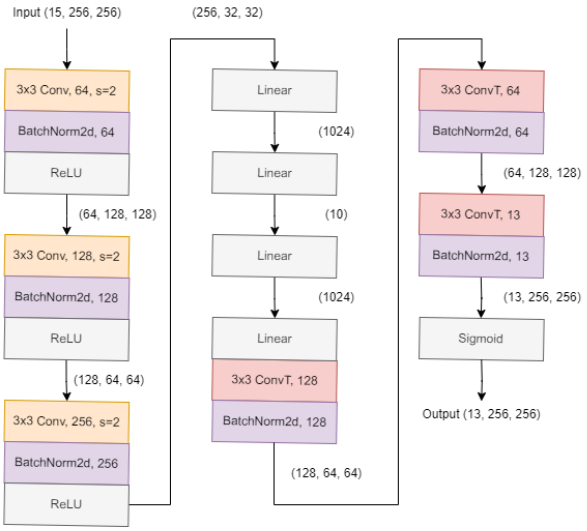


Figure 51: Anna architecture

The initial model, named `Anna`, was developed to assess the capabilities of de-convolutions for dimensionality reduction. This reduction, which was carried out by striding the convolutions, was also pursued with the aim of exploring the data distribution within a latent space. Similarly, `Veronica` was created by solely modifying the linear layers in the middle with GlobalAveragePool layers, enabling the determination of the output.

The reason behind introducing these transposed convolutional layers at this stage of development was straightforward. The use of deconvolutions allows upsampling and creating new filters at the same time, without involving a new use in memory.

Although the residual blocks in the models had a higher level of complexity and there is no transposed residual convolutional block, when dealing with images having spatial dimensions of `256x256`, employing autoencoders for dimensionality reduction, latent space exploration, and unsupervised classification might not yield satisfactory results due to a suboptimal model design that results in significant loss during reconstruction. In this sense, the concept of incorporating skip connections was conceived to retain more information in the model.

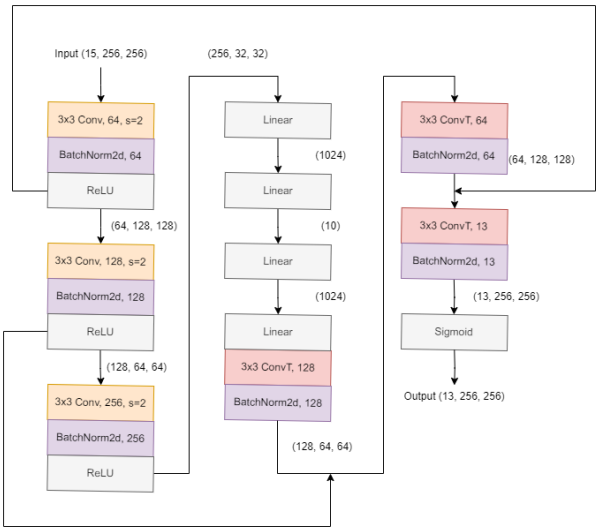


Figure 52: AnnaSkip architecture

Although it will be shown an improvement with the introduction of skip connections, it still fell short of comparing favorably with other models. Thus, it became apparent that the issue might be rooted in the dimensionality reduction process. To address this limitation, more flexible models were considered, such as the `U-Net` architecture, where the dimension was not restricted to a one-dimensional vector but comprised different filters with reduced spatial dimensions. This led to the creation of the `Ausias` model, which, while not converting the image into a 1D vector to analyze its characteristics, yielded better performance metrics.

3.3.3 Generative Adversarial Networks

GANs have been a majority of the work entailed during the research of cloud removal. It is notable that adversarial training have been a significant popular focus of our research in cloud removal. So, therefore, it's worth emphasizing that the use of adversarial training has been pivotal in our approach.

Although it seems obvious, the most difficult part of training a GAN is the instability of the training loop, since two neural networks are trying to outdo each other. Finding the right balance between the two networks and achieving convergence is challenging.

Hence, instead of contemplating an architecture where the generative model might struggle to comprehend the dataset, the approach was to use an architecture that demonstrated

effectiveness with the existing dataset, which is `Regina`. Subsequently, an exhaustive research have been pursued for getting the most suitable discriminative models. This involved adding and removing layers, fine-tuning parameters of the optimizers and implementing strategies, which will be elaborated upon below. Estimating the architecture was straightforward, since it is recommendable to start using the same amount of parameter of the generative network. Also, it is a good practice to use LeakyReLU instead of ReLU in order to make the discriminator less smart than the generator. In a standard ReLU function, the output is zero for any input less than or equal to zero and linear (equal to the input) for positive values. LeakyReLU introduces a small, non-zero gradient for negative input values, instead of being completely zero.

3.3.4 Denoising Diffusion Models

Finally, the last section involves discussing diffusion models. Given a lack of domain expertise and uncertainty regarding how this dataset might respond to diffusion models, an effort was made to find a library that would simplify the design process. While it's true that `Regina` could have been employed as a model, since the only requirement is to use the same spatial dimensions. However, it was deemed unsuitable for this purpose when trained as a denoising model due to perceived limitations in its complexity. Instead, the chosen model was a variant of the UNet with residual downsampling blocks. At each downsampling step, the model incorporates attention mechanisms similar to how ViT would employ the Scale Dot Product Attention in its attention heads.

The inclusion of spatial self-attention in the downsampling block allows the model to capture long-range dependencies in the image. Traditional convolutional layers have a limited receptive field, meaning they can only capture information from nearby spatial locations. With spatial self-attention, the model can consider information from all parts of the image, making it more powerful and potentially improving its performance on tasks like segmentation.

It's worth noting that attention mechanisms, especially in large models or images, can be computationally intensive. However, their ability to capture long-range dependencies and focus on relevant parts of the input often leads to improved performance in many tasks.

4 Experiments and results

In regards to experiments and results, it is essential to address the impact of using a sigmoid activation in the final layer and normalizing the data after being fit into the model. These factors have led to suboptimal model training, despite initial expectations. Consequently, the decision has been made to split the experiments and results section into two parts. Initially, the focus will be on hyperparameter decisions and comparisons across various datasets and metrics. For those models that were able to be retrained enough to display the metrics correctly and see their comparisons, the table has been changed. Subsequently, a separate section will present the results obtained with the current models, highlighting their limitations and proposing potential solutions to address these issues.

4.1 Challenges with data normalization

Normalizing data is a crucial step in data preprocessing, especially in the context of computer vision, when it is useful to know the imagery mean and deviation. It ensures that data is consistent across the dataset, making it easier to compare and analyze different data points. Many optimizers converge faster when data is normalized, as it ensures features have similar scales, making optimization landscapes more symmetric.

In this case, the data exhibited a wide range from 0 to 10,000. At that time, it was strongly emphasized that this preprocessing step was vital because excessively preprocessing the cloud-free and cloud-covered images could result in an unrealistic model. To address this concern, normalizing the data was considered. Nevertheless, performing min-max scaling did not yield favorable outcomes for several reasons:

- The output data values became very constrained, preventing neural networks from making substantial adjustments, resulting in a persistently low function loss.
- It was expected that metrics like MSE, and MAE would register smaller values due to the scaling. However, it was not anticipated that the parameters SSIM metric should also be changed. This implies that achieving an SSIM metric around 1 with such small output values is not equivalent to achieving the same SSIM if the data range is not checked. Applying these metrics in such a small range has meant that they cannot provide enough information to the models.

At this point, it became apparent that utilizing the sigmoid function plus the normalization was not as effective as initially thought. Due to the wide range of values, applying normalization did not significantly impact the loss function, specifically in this case, the MSE score, when dealing with very small differences once the training had already taken some epochs. After addressing the issue, two methods for preventing this problem have been identified:

- Clipping: This method involves limiting the range of values in an image. In [7], they employ clipping to constrain the values within the range of 0 to 2000.
- Scaling Sigmoid Activation: It is indeed advantageous to use a sigmoid activation as the final function since it constrains the neural network's output. Instead of removing the sigmoid, it is recommended to keep it and adjust its final output value by scaling it according to the range of values present in the dataset.

Both options can be implemented concurrently. Nevertheless, the decision between utilizing clipping or employing a sigmoid has been deferred to future iterations of the project.

4.2 Results

To start this section, we will address the issue of underfitting. To observe the progression of the networks, we initially employed simpler models, such as `Carla` or `Anna`. Subsequently, we incorporated similar structures and introduced new ones, gradually increasing the complexity, as seen in the `Regina` and `Ausias` models, respectively.

Regarding the `Carla` model, it is evident that it lacks the capacity to further minimize its loss function. Also, there are instances when the validation dataset exhibits slight fluctuations. This is a clear case of underfitting. The occurrence of underfitting as early as the first epoch is primarily attributed to the use of a very low batch size. A smaller batch size implies that the model is processing more precise but limited information at each step, leading to a more optimized backpropagation of tensors for each individual case.

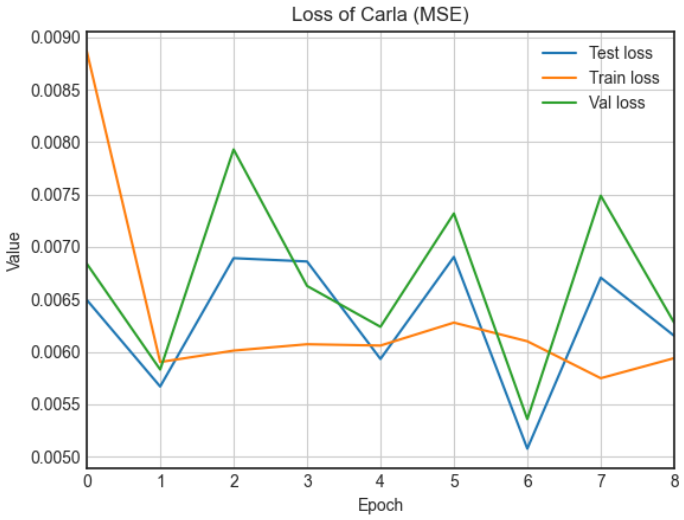


Figure 53: Loss of the Carla best run.

When it comes to the utilization of residual blocks, there is a significant difference in the loss between `Carla` and `Regina` in Figure 7, despite the fact that the normalized values are not accurate.

Table 7: Comparison of metrics between `Carla` and original `Regina` (with data normalization)

Model	MSE	PSNR
Regina	0.252	27.57
Carla	0.603	10.21

The table 8 demonstrates that SAR data has substantial impact on the metrics, making them slightly more challenging. On the other hand, using data augmentation does not improve significantly the scores. This marginal changes in augmenting the data could be attributed to the dataset's inherent variability, which allows for the detection of complex correlations between the spectral bands. Although the augmented data was about 30 %, it seems it does not incorporate value to the dataset.

Table 8: Comparison of metrics between variants of `Regina` (with normalized data).

SAR	Augmented	MSE	PSNR	SSIM
No	No	0.23	6.38	0.15
No	Yes	0.15	9.28	0.10
Yes	No	0.002	27.57	0.783
Yes	Yes	0.005	23.01	0.32

It is noteworthy that the `Anna` and `Veronica` models, which involve the transition

from 2D data to 1D data and then back to 2D, encounter greater difficulty in establishing correlations among their neurons. Conversely, models that incorporate a bottleneck but maintain the data in a 2D format do not experience as much difficulty during training. Additionally, to further enhance the Ausias model, a convolutional layer was integrated after each de-convolution, serving as an analogy to process these layers and unify different kernels generated based on input values. In the end, the model known as AusiasConv emerged as the top-performing autoencoder within its category.

Table 9: Comparison of metrics of the autoencoders (without normalizing data).

Model	MSE	PSNR	SSIM	CARL
Veronica	69285	-0.32	-3.93	4354.43
Anna	64432	0.04	-1.93	3320.87
AnnaSkip	27430	2.443	-1.32	954.32
Ausias	9423.32	8.38	0.01	**293.32**
AusiasConv	**6485.80**	**10.011**	**0.08**	302.88

Furthermore, a training comparison was conducted using both the MSE and MAE loss functions, as the calculation methods for these results share significant similarities. It can be observed in 14 that models trained with the MAE loss function yielded poorer results and exhibited considerably more instability compared to those trained with MSE, since the model weights were far more inbalanced and they could output `NaN`s results.

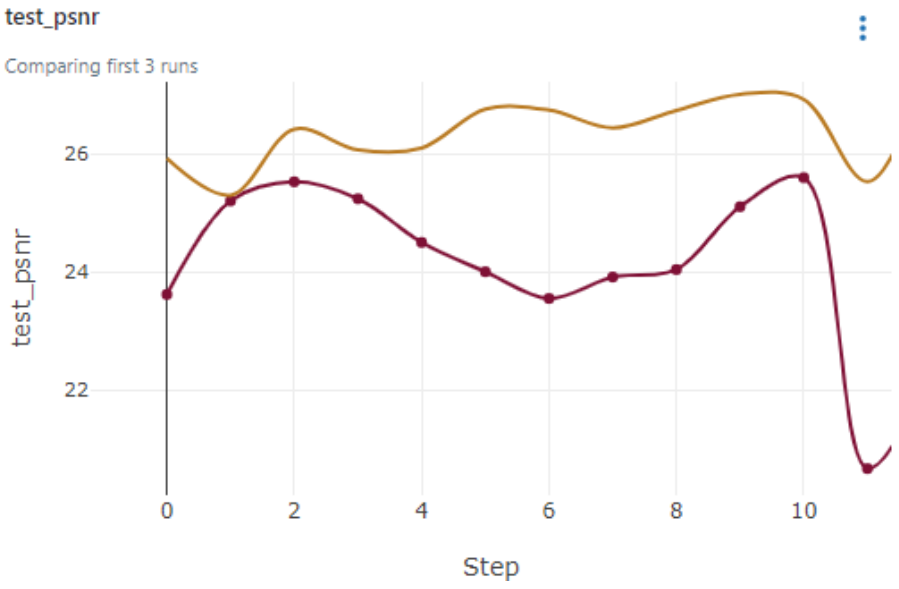


Figure 54: Comparison of test PSNR with L1 (red) and MSE (orange) of AusiasConv, normalized data.

Table 10: Comparison of loss functions in Ausias model (normalized data).

Model	Loss Function	MSE	PSNR	SSIM
AusiasConv	L1	0.0058	22.3	-2.3
AusiasConv	MSE	**0.003**	**25.22**	**0.08**

Regarding GAN training, while designing the discriminator's architecture presented challenges, the most demanding aspect was fine-tuning the hyperparameters to strike a balance between the two models in this adversarial setup. Initially, it was found that when using Adam as the optimizer, the beta values needed to be adjusted differently compared to a basic classification challenge scenario. Furthermore, the learning rate of the discriminator was lowered to minimize its learning during the initial iterations. Consequently, the weight decay for the discriminator was increased slightly to counteract its growing ability to detect changes in the generator.

Table 11: Hyper parameters of RegiGAN

Hyper-parameter	Generator	Discriminator
Optimizer	Adam	Adam
Learning rate	10^-5	10^-8
Weight decay	$5 \cdot 10^-5$	$1 \cdot 10^-4$
Beta A	0.5	0.5
Beta B	0.999	0.999

Additionally, to assess the dynamics between the two networks, various parameters were introduced, including the accuracy of the discriminator, which ideally should hover around 0.5, and the scores for fake and real inputs, where both scores should also aim for 0.5. The "fake score" and the "real score" are terms commonly used in the context of GAN to assess the discriminator's performance. These scores represent how well the discriminator distinguishes between real and fake data samples during the training process. The real score is the output of the discriminator when it evaluates real data samples from the training dataset. Conversely, the fake score represents the output of the discriminator when it assesses generated (fake) data samples produced by the generator. These metrics played a crucial role in stabilizing the model. Actually, this metrics needed to be calculated in each batch, since any particular and remarkable change in the losses could led to desequilibrum.

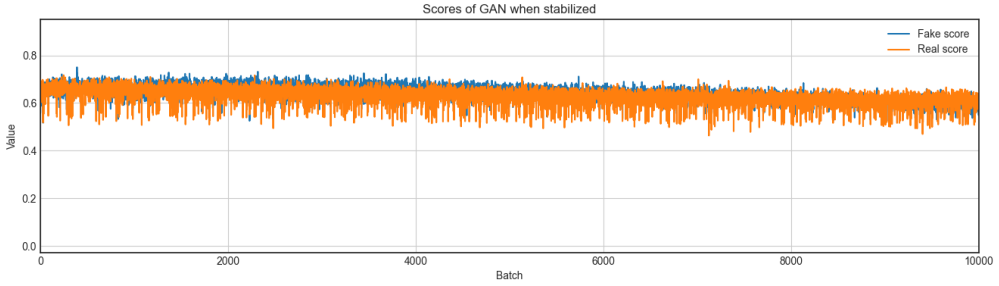


Figure 55: Stabilized training

In the final stage, to assist the generative model in its initial iterations, the MSE loss function was introduced. This addition enabled the generative model to establish its bearings, enhance training stability, and maintain a clear focus on addressing the core problem.

It can be seen in Figure 56 how the generator was able to perform better metrics than the `Regina`, although it is missing being able to see these metrics with the unnormalized data.

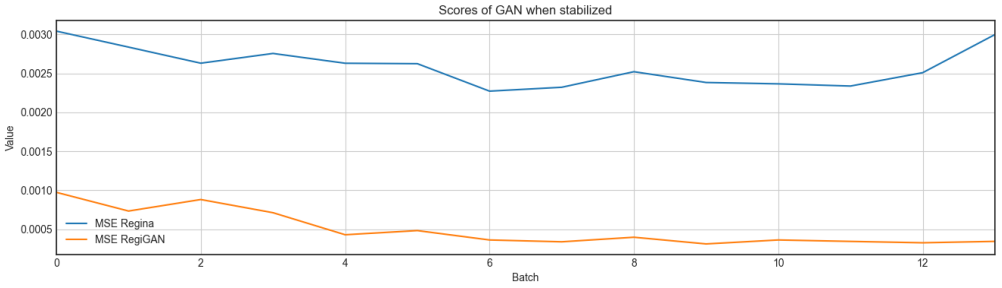


Figure 56: Comparison of MSE between RegiGAN and Regina.

Table 12: Comparison of loss functions between RegiGAN and Regina.

Model	MSE	PSNR	SSIM
Regina	**5867.52**	10.60	**0.49**
AusiasConv	6485.80	10.011	0.08

Regrettably, the diffusion model could not be deployed due to its prohibitively expensive training process and the inability to stabilize the model. Consequently, no metrics can be presented for the test data.

Table 13: Comparison of the scores of the models

Model	MSE	PSNR	SSIM
Regina (ours)	5867.52	10.60	0.49
AusiasConv (ours)	6485.80	10.011	0.08
RegiGAN (ours)	5863.467308044434	10.449258527964542	0.642
DSen2-CR	-	27.76	0.874
UnCRtainTS	-	28.90	0.88

In summary, this section sheds light on several crucial observations although the results are not quite good enough. Firstly, it becomes evident that transposed convolutions introduce a significant degree of imbalance into the network, resulting in training instability. Notably, the training of an image-to-image network presents a considerably higher level of imbalance compared to the more conventional tasks of classification or regression. This heightened imbalance necessitates a more pronounced fine-tuning process, a factor that was initially underestimated.

In general, it appears that the project may have not utilized sufficiently powerful neural networks. This decision was guided by the intention to proceed methodically and incrementally. Unfortunately, this cautious approach has resulted in delays. Consequently, there is a pressing need to consider the development of new models that are more aligned with achieving a practical solution.

The issue of normalization has exacerbated this situation. Regrettably, it was only recognized at a later stage that the models were not effectively learning, which contributed to the setbacks in the project's progress.

Furthermore, it becomes apparent that the utilization of autoencoders and VAE may have limitations when applied to larger images or when left in a latent space towards the end of the network architecture. It could be beneficial to explore alternative approaches, such as reconstructing images separately, one without clouds and another with clouds, to assess the network's behavior more effectively.

Additionally, there was an initial suspicion regarding the computational efficiency of diffusion models, but the extent of their time-consuming nature exceeded expectations, highlighting the need for careful consideration and optimization in their implementation.

4.3 Experiments

In this section, we will showcase the results of the top three models. Regrettably, due to normalization issues, we were unable to carry out tests involving bands 9. Furthermore, it's worth highlighting that both autoencoders and Variational Autoencoders (VAE) yielded unsatisfactory results. Consequently, displaying features and characteristics of the latent vector or the output image without achieving an accurate image reconstruction would not be meaningful.

Table 14: Comparison of the scores of the models

Model	MSE	PSNR	SSIM
Regina	5867.52	10.60	0.49
AusiasConv	6485.80	10.011	0.08
RegiGAN	5863.467308044434	10.449258527964542	0.642

To further analyze and compare the performance of the three leading models, a series of visualizations were carried out. It's interesting to note that `RegiGAN` doesn't excel in terms of metrics as `Regina` does, yet it demonstrates a noticeable advantage in image quality. This observation prompts two questions:

- The consideration of a more robust loss function, as the current error approximation may not directly translate into qualitative enhancements.
- The possibility of errors in the metrics or image processing methods, which could be affecting the assessment of model performance.

Furthermore, it has been demonstrated that these models lack the necessary complexity and require additional training time, as the results obtained so far have not been satisfactory.

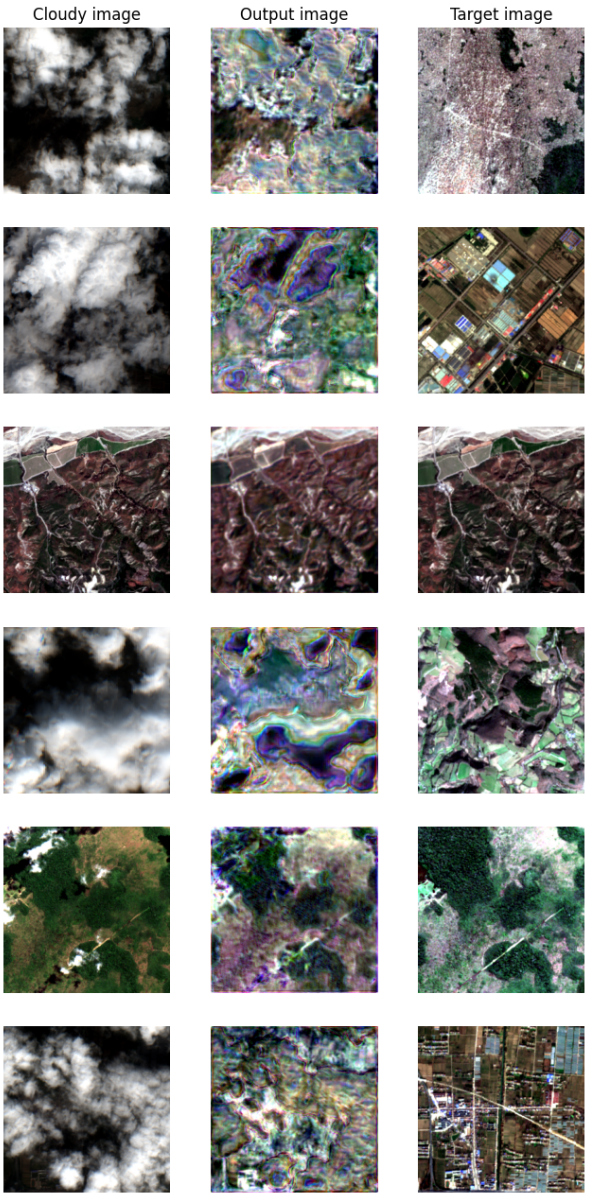


Figure 57: Sample of cloudy (left) and cloudless (right) images along with `Regina`'s synthetic.

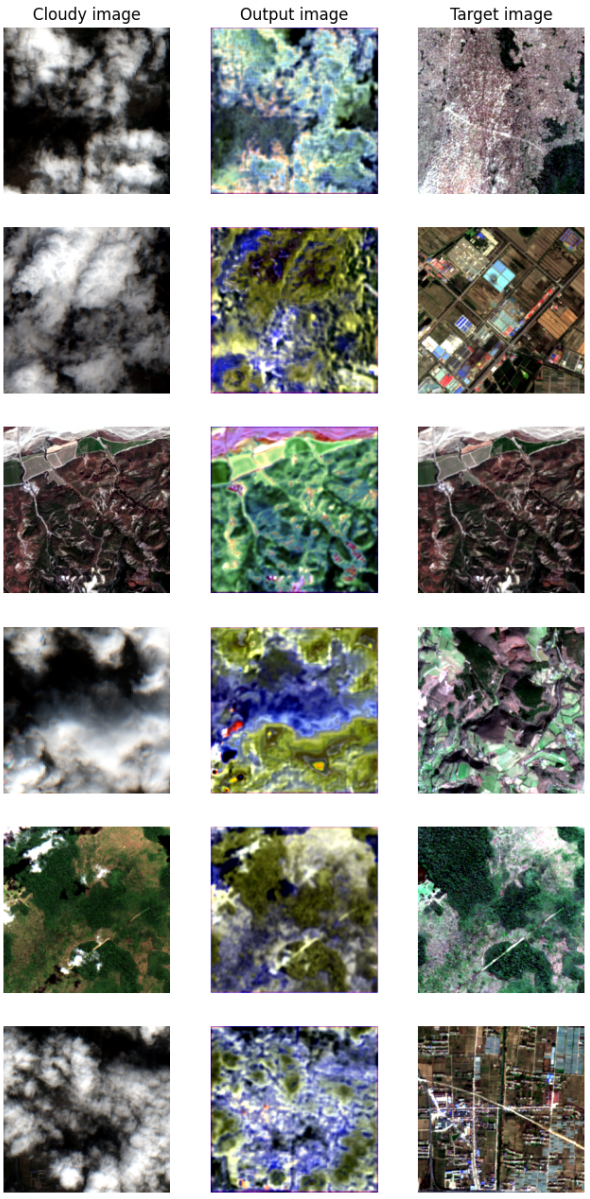


Figure 58: Sample of cloudy (left) and cloudless (right) images along with `Ausias`'s synthetic.

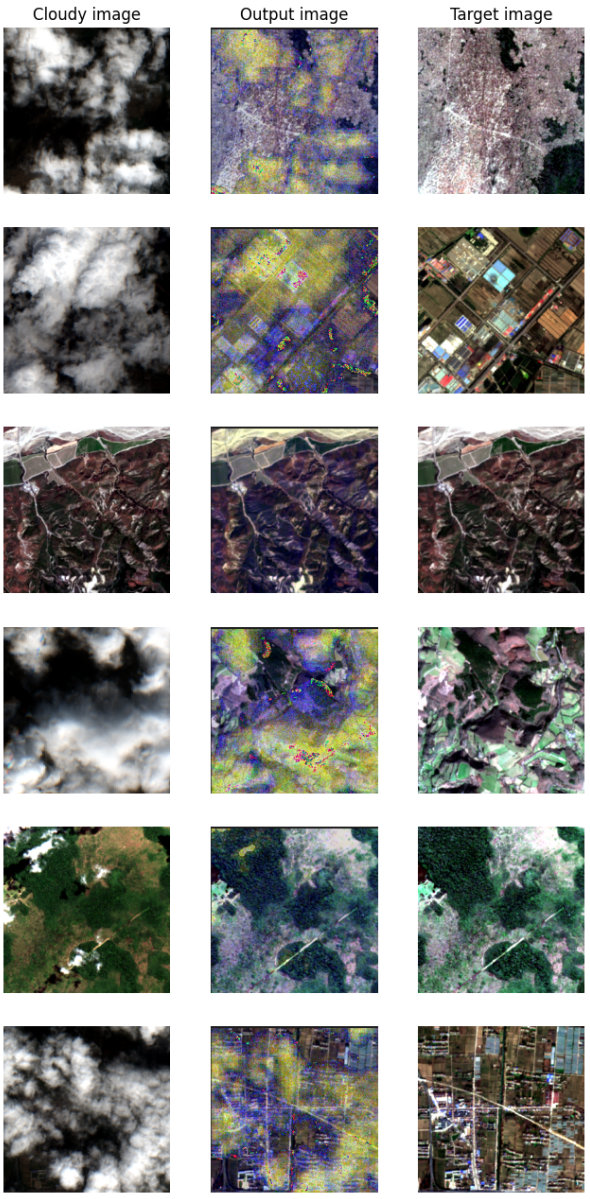


Figure 59: Sample of cloudy (left) and cloudless (right) images along with `RegiGAN`'s synthetic.

5 Future directions

The realm of generative modeling offers a vast spectrum of possibilities, and as exploration deepens, the horizons of what can be achieved continue to expand. Recognizing past challenges and learning from them will be crucial in charting the path forward.

A primary focus of the subsequent phase of this thesis will be to zero in on a singular network type. This decision emerges from the prior explorative phase, where various model architectures were extensively navigated. Concentrating on a specific network will allow for a more profound and nuanced understanding of its capabilities and limitations.

Furthermore, the choice of loss function is pivotal in the training process. The current losses, either due to their lack of specificity or computational intensity, may not be optimal. Future endeavors should emphasize identifying and implementing a more efficient and relevant loss function, ensuring not only enhanced training efficacy but also greater model precision.

Visualization plays an indispensable role in understanding the intricacies of data. A particularly intriguing observation was the significance of the ninth band in the reconstruction process. Delving deeper into such findings could unearth valuable insights and guide model refinement.

Lastly, as the project advances, leveraging the well-structured codebase will be advantageous. It would be prudent to add images in the a dedicated platform for training and tracking. The platform needs to facilitate real-time monitoring of the training process, enabling timely inferences from saved checkpoints.

In terms of visualizations and experiments, there has been lacking comparing models utilizing remote sensing with both real and synthetic images. This approach would allow us to assess whether it enhances the accuracy and effectiveness of the model.

In essence, as the future unfolds, a blend of specialization, efficient computational strategies, and insightful visual analyses will be paramount in propelling the research to new heights.

Economic Analysis

While the project primarily centers around research, it's essential to consider that applied research incurs costs. Consequently, a basic economic analysis has been carried out to assess the expenses incurred thus far.

In the economic analysis, we assumed that an individual with a master's degree would cost the company approximately 70€/h. Given that the work was conducted on a part-time basis, a total of 430 hours were spent, equating to 30100€. On the other hand, the GPU was used for a total of 182 hours. Given that the average price is around €2 [28] per hour, €364 was spent in total. Considering that machine learning or deep learning projects typically cost between 60000€ to 100000€, the project remains profitable. However, since there is job to be done the required time for the project would decrease its profitability. Assuming it will be needed one entire month more, the expenditure could amount to a total of €42,140.

Table 15: Cost of the project

	Value	Conversion	Total (€)
PM	2.5 PM	$70\frac{€}{h}$	30100
GPU	182 hours	$2\frac{€}{h}$	364
Total			30464
Predicted PM	2.5 PM	$70\frac{€}{h}$	12040
Total			42140

Part III

Conclusions

In summary, the introduction and results suggest that while RS imagery and deep learning hold great promise, there is a need for more powerful neural networks, improved data normalization, and consideration of alternative approaches to overcome cloud coverage challenges.

The project aimed to search for and analyze state-of-the-art approaches and frameworks in the field of remote sensing and deep learning. This exploration was intended to identify novel methods that could lead to improvements and superior results in cloud removal from satellite imagery.

Technically, the project's cautious approach of using less powerful neural networks for methodical development has led to delays. There is a clear need to reconsider this approach and explore more potent neural network models to expedite progress. Some challenges were exacerbated by issues with data normalization. Recognizing these issues at a later stage hindered model learning and contributed to setbacks. Addressing normalization problems is crucial for better results. The project found that diffusion models were more time-consuming than initially suspected. This highlights the importance of careful consideration and optimization in the implementation of such models to make them computationally efficient.

Professionally, I must admit that the Master's thesis has provided me with extensive knowledge about generative images. However, there is still a considerable distance to cover before I can feel entirely at ease with this type of problem-solving and approach it with greater ease. On the positive side, I now feel more capable and prepared to rigorously adhere to project timelines, even if it requires prioritizing certain aspects over others. This experience has taught me the importance of identifying what truly matters and having the ability to emphasize or de-emphasize elements based on their relevance to the project's ultimate goals. In essence, this project has been a valuable learning experience, helping me establish effective boundaries when working on complex tasks.

References

[1] Patrick Ebel, Andrea Meraner, Michael Schmitt, and Xiao Xiang Zhu. Multisensor Data Fusion for Cloud Removal in Global and All-Season Sentinel-2 Imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 2020.

[2] Patrick Ebel, Yajin Xu, Michael Schmitt, and Xiao Xiang Zhu. SEN12MS-CR-TS: A Remote Sensing Data Set for Multi-modal Multi-temporal Cloud Removal. *IEEE Transactions on Geoscience and Remote Sensing*, 2022.

[3] Patrick Ebel, Vivien Sainte Fare Garnot, Michael Schmitt, Jan Dirk Wegner, and Xiao Xiang Zhu. Uncrtaints: Uncertainty quantification for cloud removal in optical satellite time series. *arXiv preprint arXiv:2103.01902*, 2021.

[4] Charis Lanaras, José Bioucas-Dias, Silvano Galliani, Emmanuel Baltsavias, and Konrad Schindler. Super-resolution of sentinel-2 images: Learning a globally applicable deep neural network. *ISPRS Journal of Photogrammetry and Remote Sensing*, 146:305–319, 2018.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[6] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[7] Andrea Meraner, Patrick Ebel, Xiao Xiang Zhu, and Michael Schmitt. Cloud removal in sentinel-2 imagery using a deep residual neural network and sar-optical data fusion. *ISPRS Journal of Photogrammetry and Remote Sensing*, 166:333 – 346, 2020.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[9] Kenji Enomoto, Ken Sakurada, Weimin Wang, Hiroshi Fukui, Masashi Matsuoka, Ryosuke Nakamura, and Nobuo Kawaguchi. Filmy cloud removal on satellite imagery

with multispectral conditional generative adversarial nets. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1533–1541, 2017.

[10] Vishnu Sarukkai, Anirudh Jain, Burak Uzkent, and Stefano Ermon. Cloud removal in satellite images using spatiotemporal generative networks. *arXiv preprint arXiv:1912.06838*, 2019.

[11] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017.

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. cite arxiv:1505.04597Comment: conditionally accepted at MICCAI 2015.

[13] Praveer Singh and Nikos Komodakis. Cloud-gan: Cloud removal for sentinel-2 imagery using a cyclic consistent generative adversarial networks. In *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 1772–1775, 2018.

[14] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

[15] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends in Machine Learning*, 2019.

[16] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

[17] W. Feller. On the theory of stochastic processes, with particular reference to applications. *Proceedings of the First Berkeley Symposium on Mathematical Statistics and Probability (August, 1945 and January, 1946)*, pages pages 403–432, Aug 1945. Referenced by: MathSciNet [MR0027980].

[18] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman

Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 8780–8794. Curran Associates, Inc., 2021.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010, 2017.

[20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. cite arxiv:2005.14165Comment: 40+32 pages.

[21] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8821–8831. PMLR, 18–24 Jul 2021.

[22] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1691–1703. PMLR, 13–18 Jul 2020.

[23] Alexander Kolesnikov, Alexey Dosovitskiy, Dirk Weissenborn, Georg Heigold, Jakob Uszkoreit, Lucas Beyer, Matthias Minderer, Mostafa Dehghani, Neil Houlsby, Sylvain Gelly, Thomas Unterthiner, and Xiaohua Zhai. An image is worth 16x16 words: Transformers for image recognition at scale. 2021.

[24] Vincent Sainte Fare Garnot and Loic Landrieu. Lightweight temporal self-attention for classifying satellite images time series. *arXiv preprint arXiv:2103.01157*, 2021.

[25] Xiaoke Wang Yang Wang Xian Sun Kun Fu Daoyu Lin, Guangluan Xu. A Remote Sensing Image Dataset for Cloud Removal. 2019.

[26] ESA. Sentinel-2 anomalies. `https://s2anomalies.acri.fr/anomalies`.

[27] O. Alàs. Cloud Removal. `https://github.com/Oriolac/cloud-removal`, 2022-2023.

[28] GPU pricing. `https://www.runpod.io/gpu-instance/pricing`, 2022-2023.

Appendices

A Exploratory Data Analysis

Table 16: Bandwidths translation between Sentinel2 and the datasets [1] & [2].

Band Sentinel2	Name	Band ID
B1	Coastal aerosol	0
B2	Blue	1
B3	Green	2
B4	Red	3
B5	Vegetation Red Edge	4
B6	Vegetation Red Edge	5
B7	Vegetation Red Edge	6
B8	NIR	7
B8A	Vegetation Red Edge	8
B9	Water Vapour	9
B10	SWIR-Cirrus	10
B11	SWIR	11
B12	SWIR	12

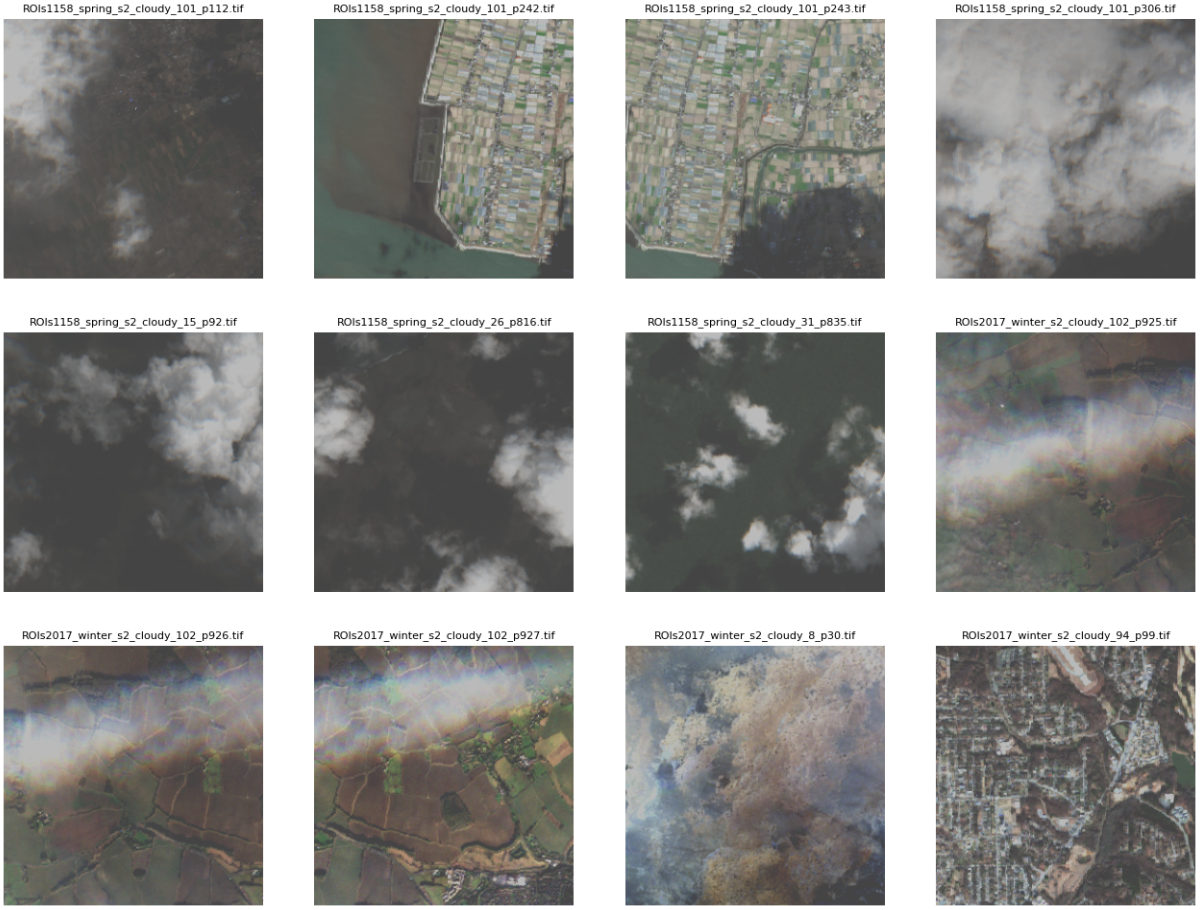


Figure 60: 12 images of the sample showing the diversity of cloudy images.

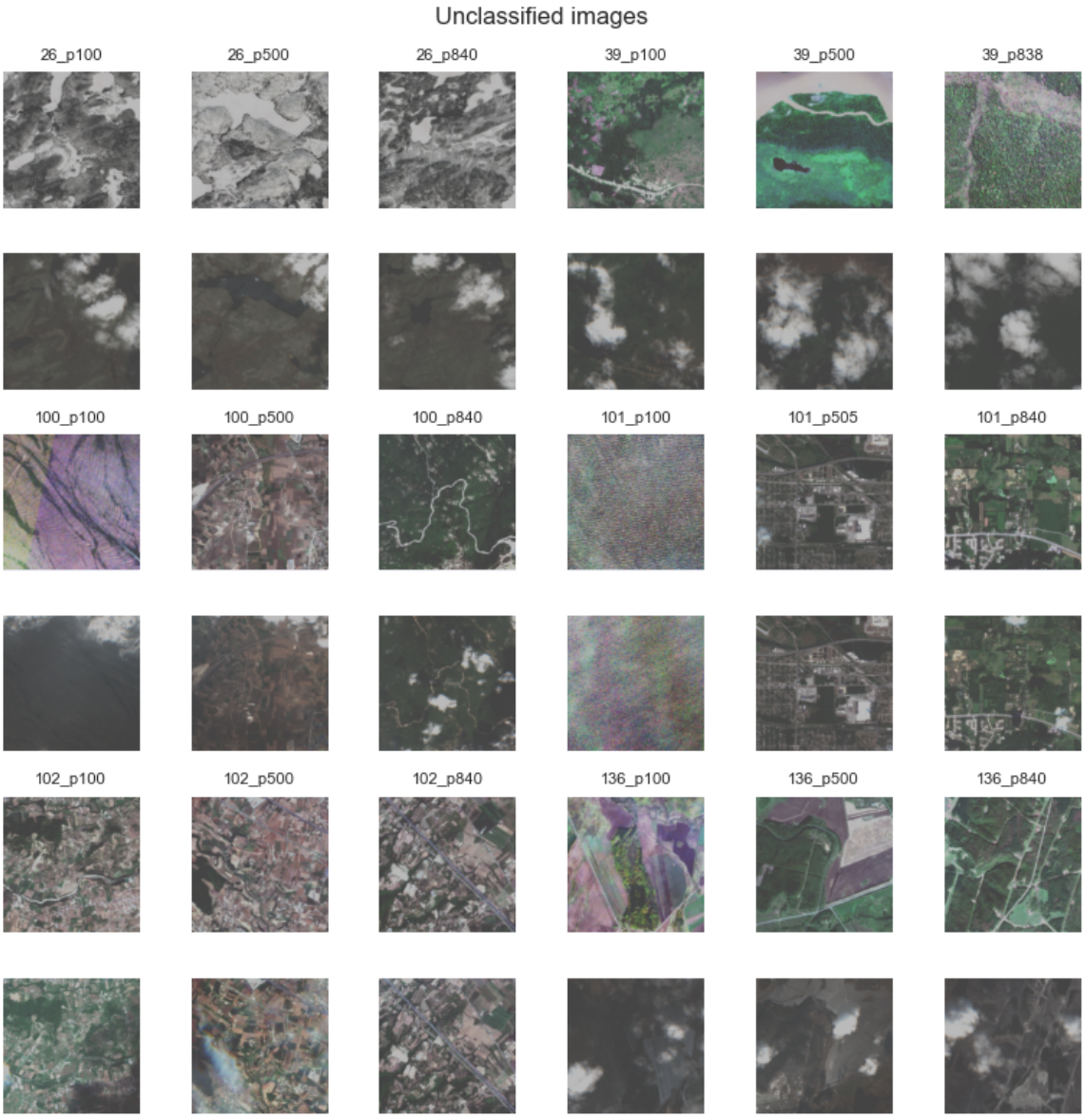


Figure 61: Unclassified images

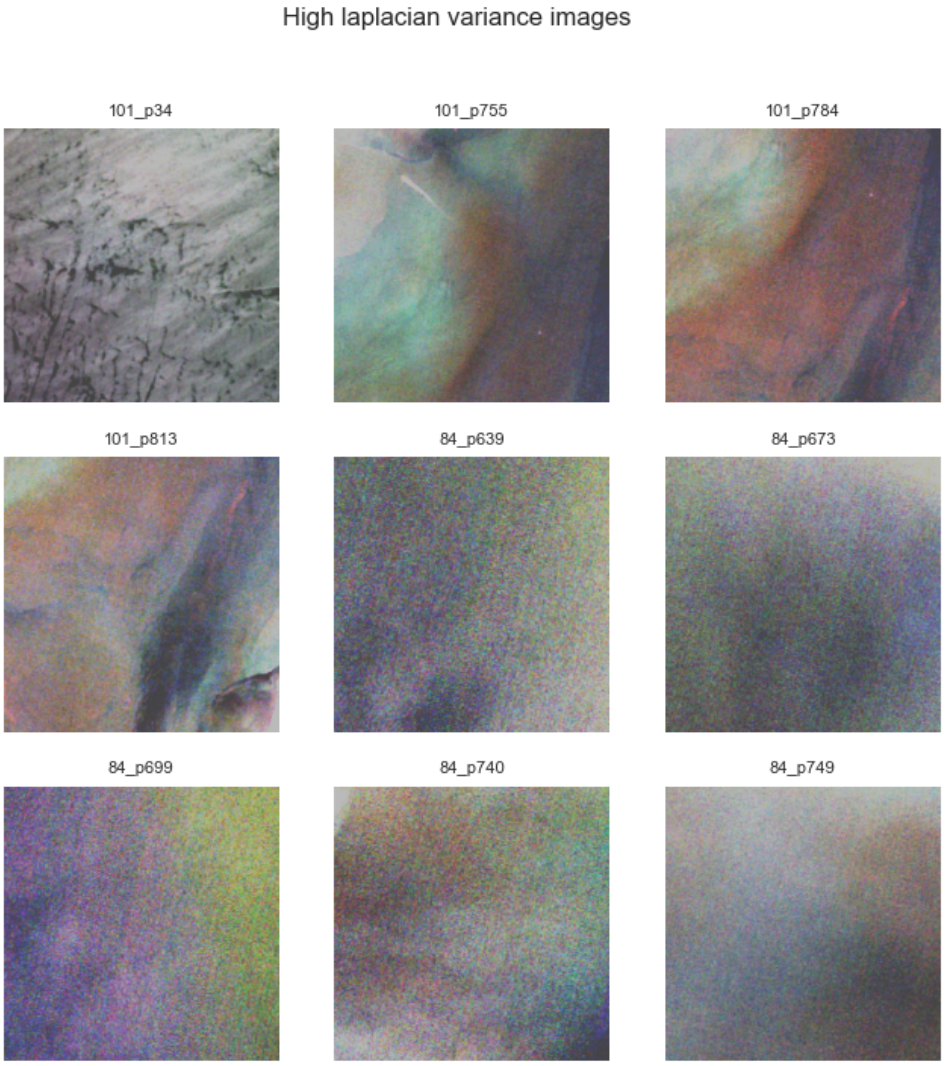


Figure 62: Sample of images with high laplacian variance, blurriness.

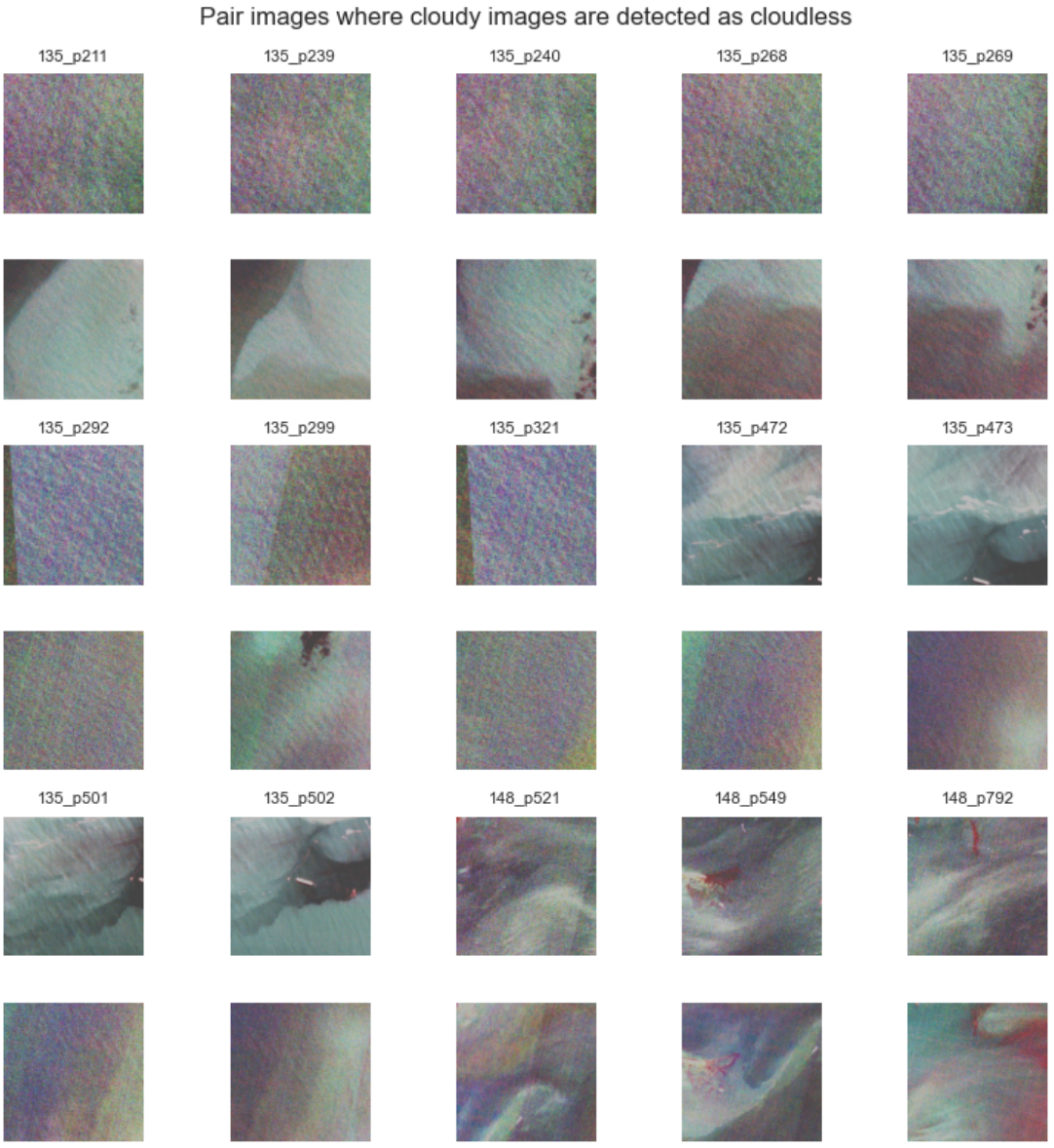


Figure 63: Sample of images labeled as cloudy but with low cloud cover percentage area.

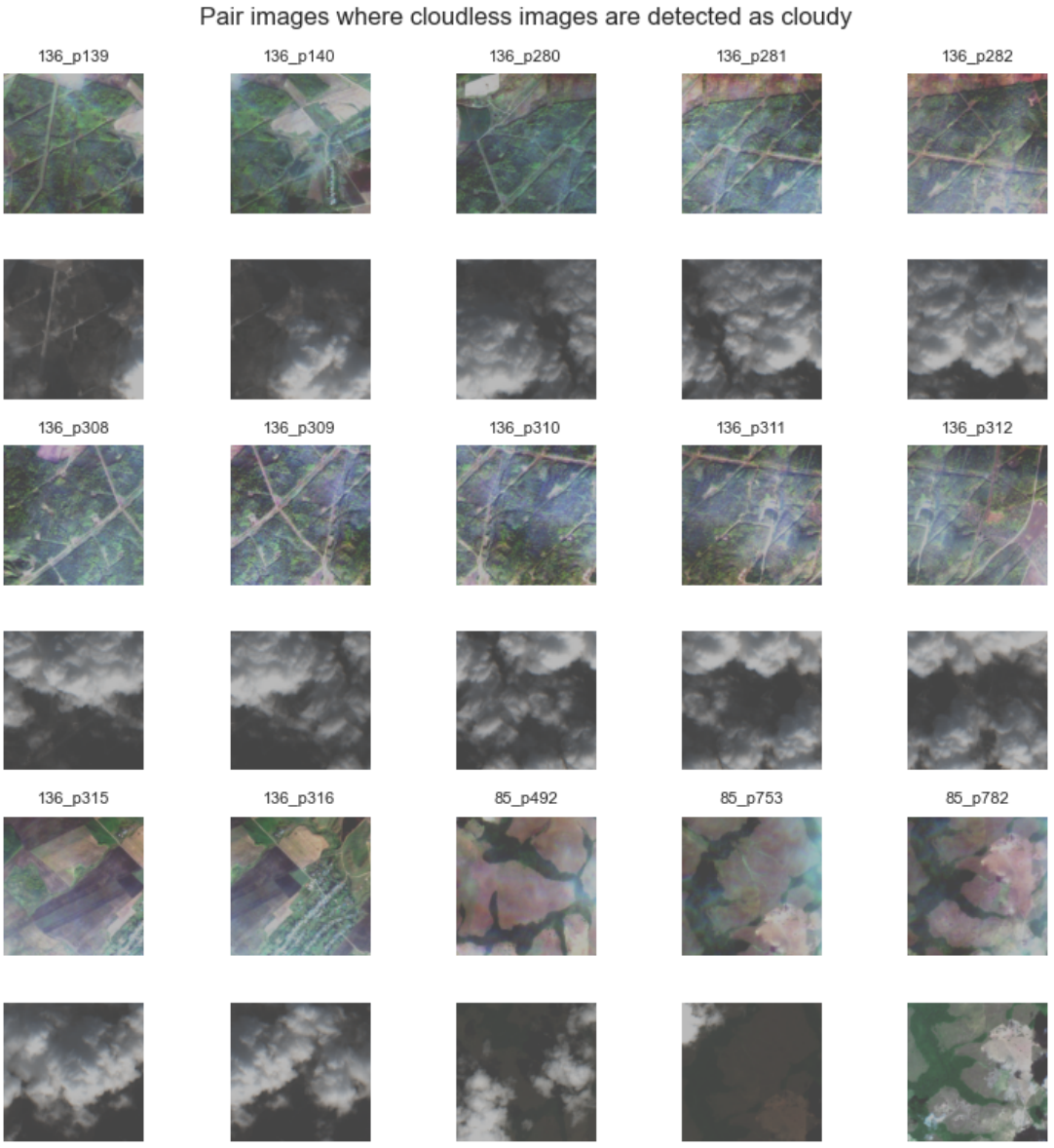


Figure 64: Sample of images labeled as cloudy but with low cloud cover percentage area.

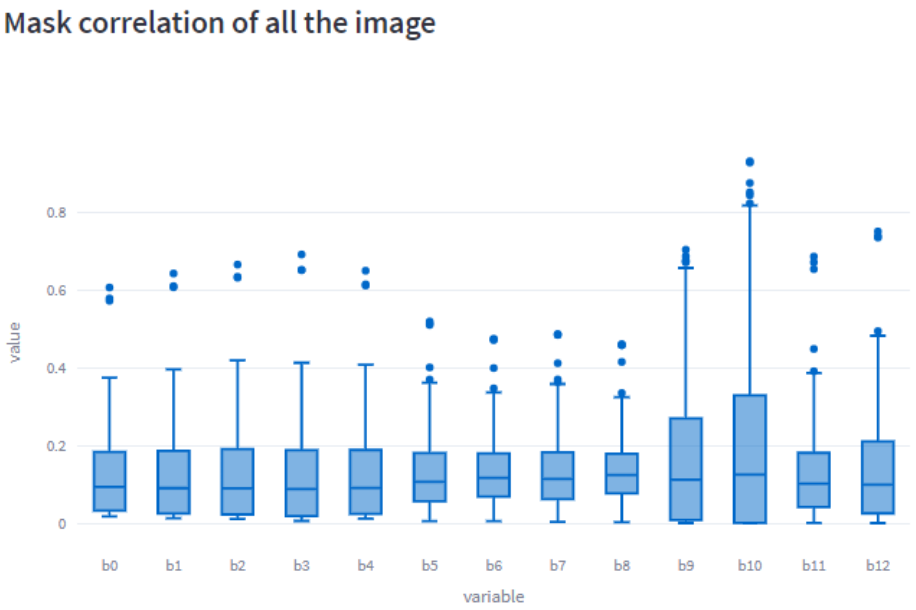
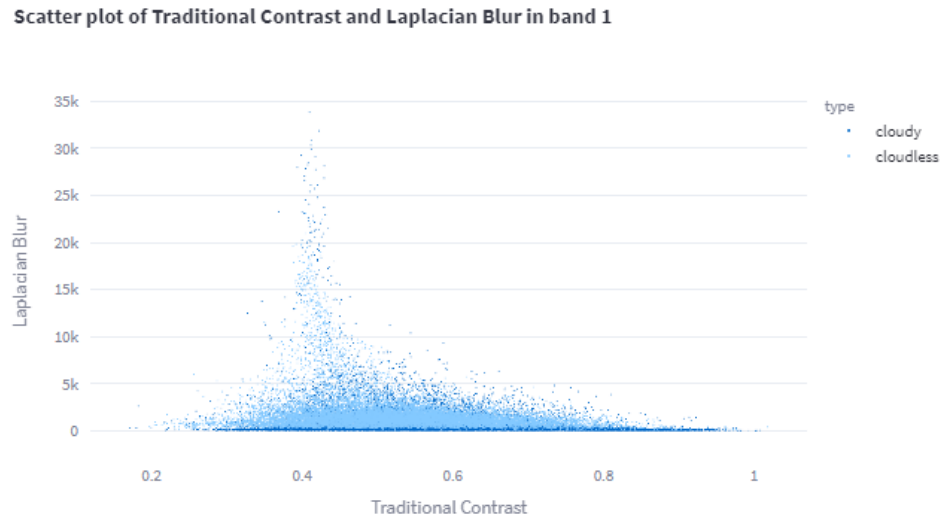
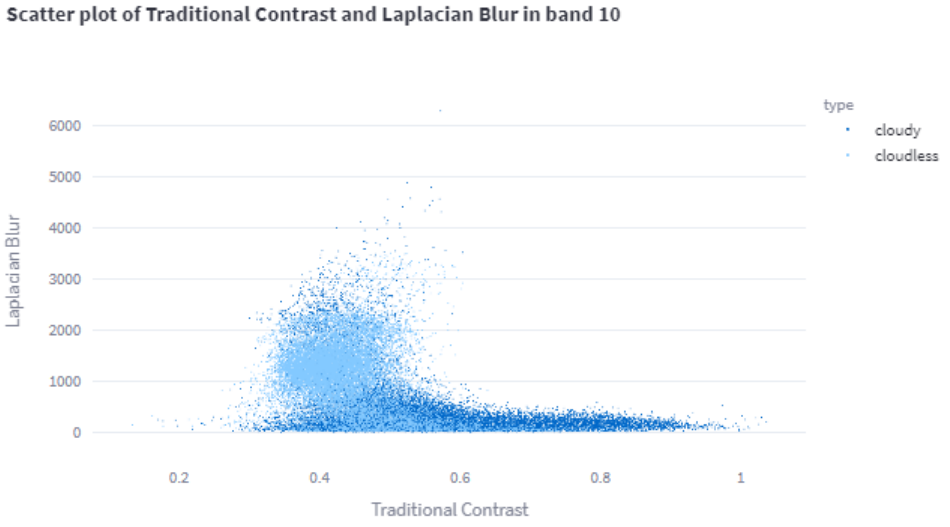


Figure 65: Correlation of the image and the values of the band, by mean from each image.



(a) Scatter plot of the contrast and the laplacian variance of B1



(b) Scatter plot of the contrast and the laplacian variance of B10

Figure 66: Comparison between B1 and B10 regarding blurriness and contrast.

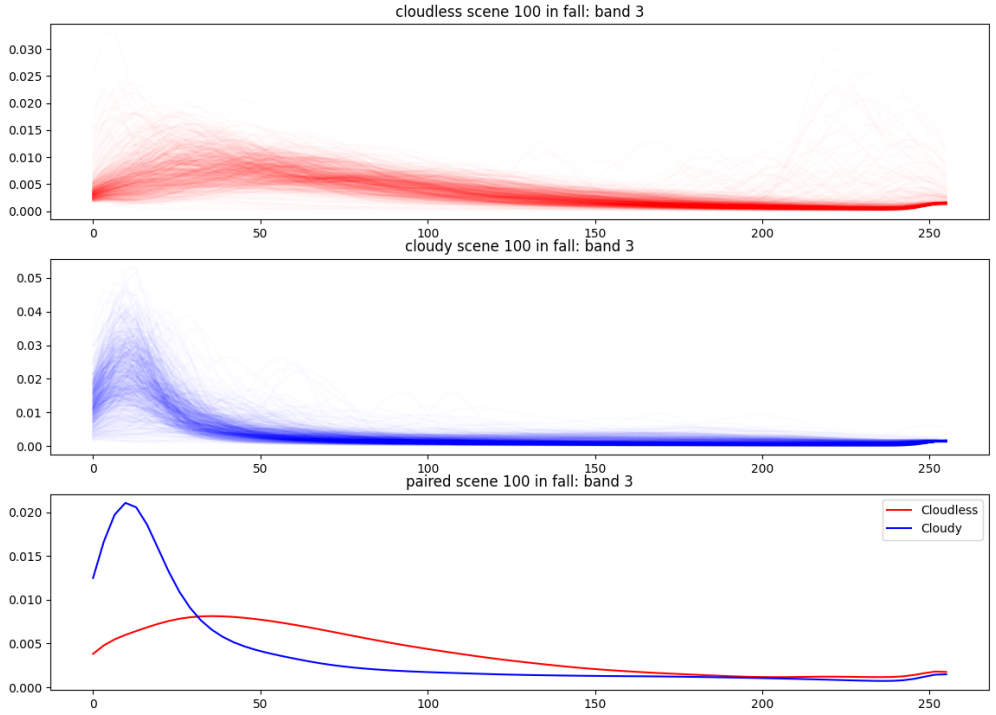


Figure 67: Kernel density estimation of B3 in cloudy and cloudless the patches of a specific scene (100)

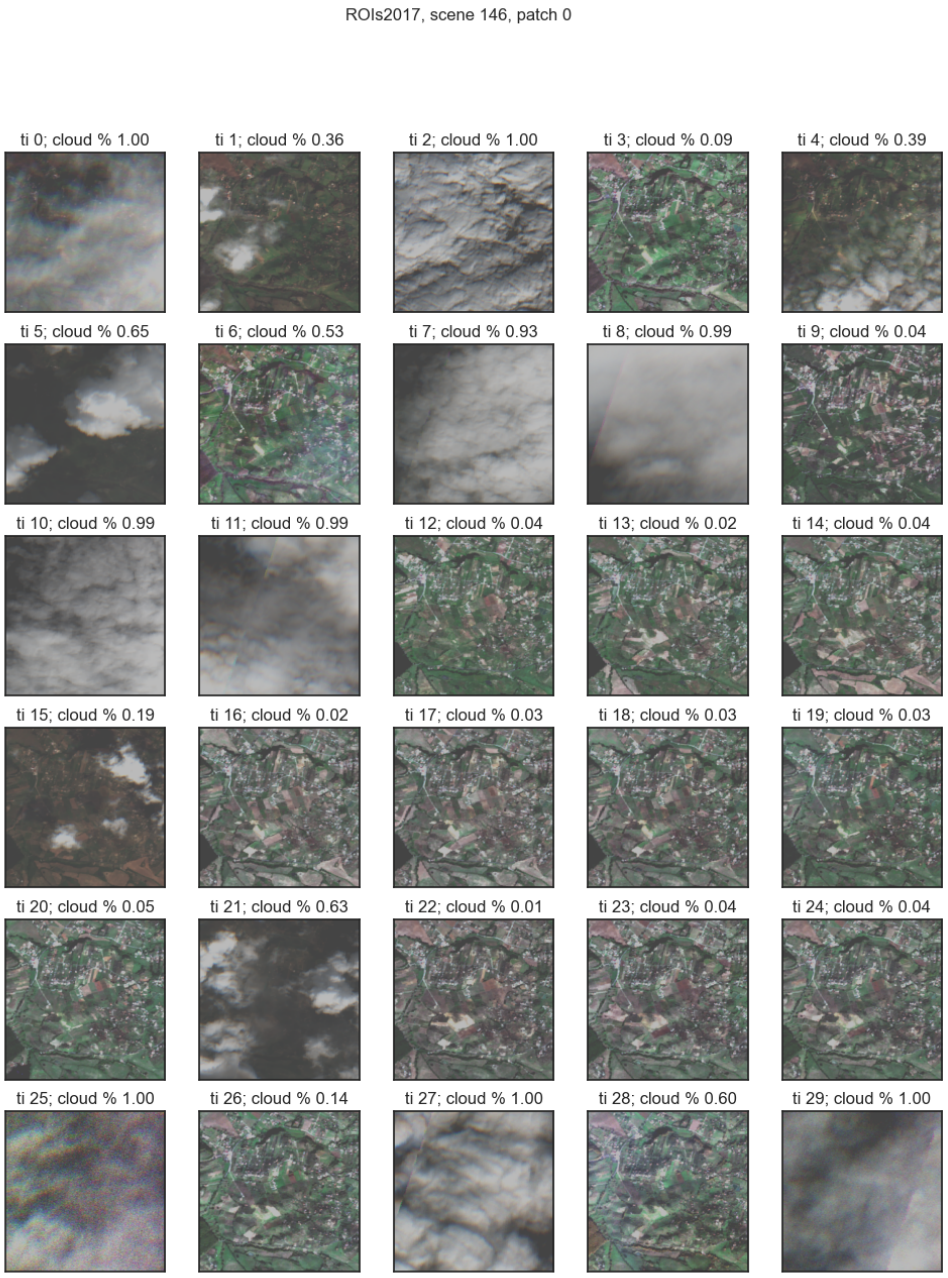


Figure 68: Patch exploration in all intervals.

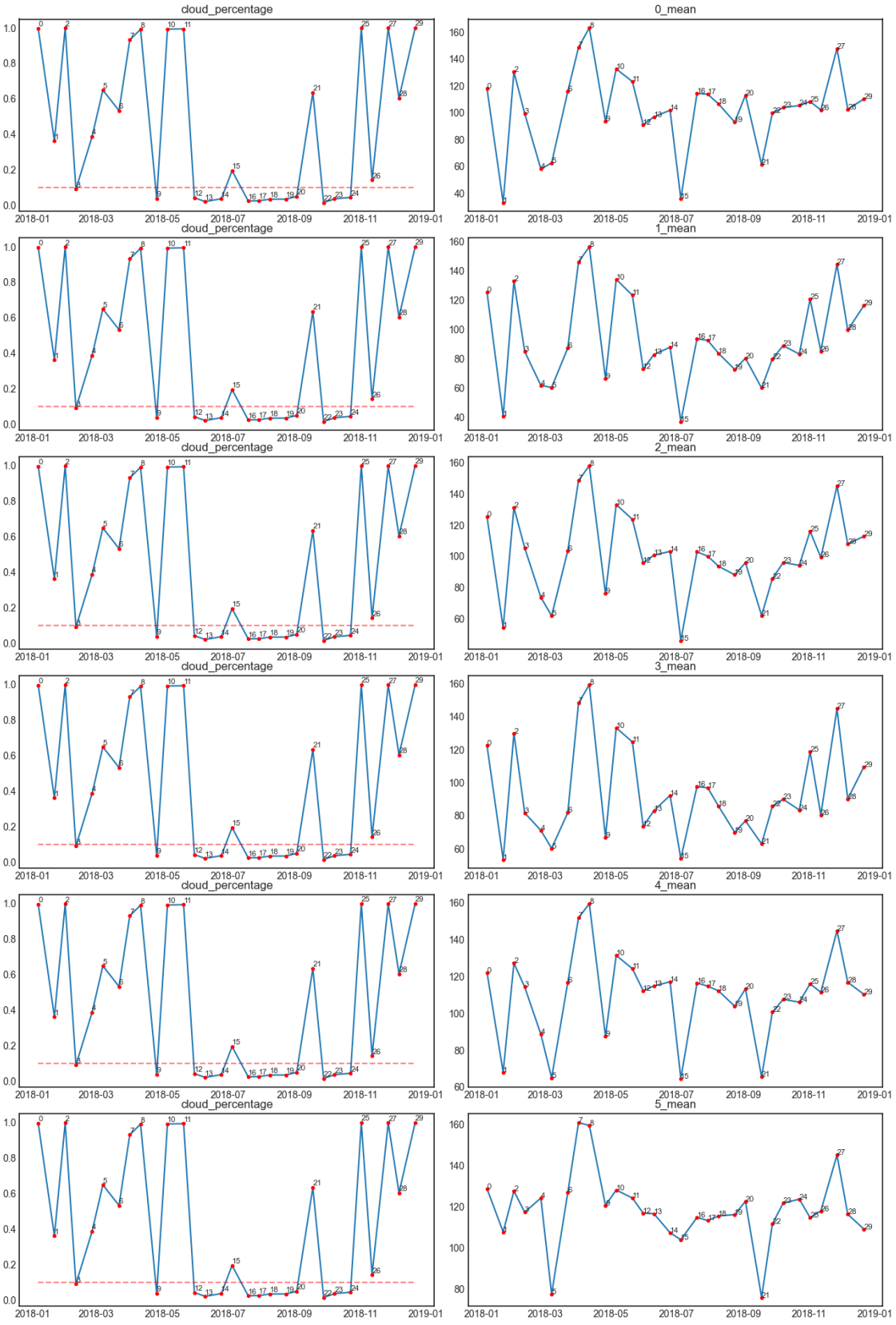


Figure 69: Cloud percentage and band mean value in all time instances (1).

Figure 70: Cloud percentage and band mean value in all time instances (2).

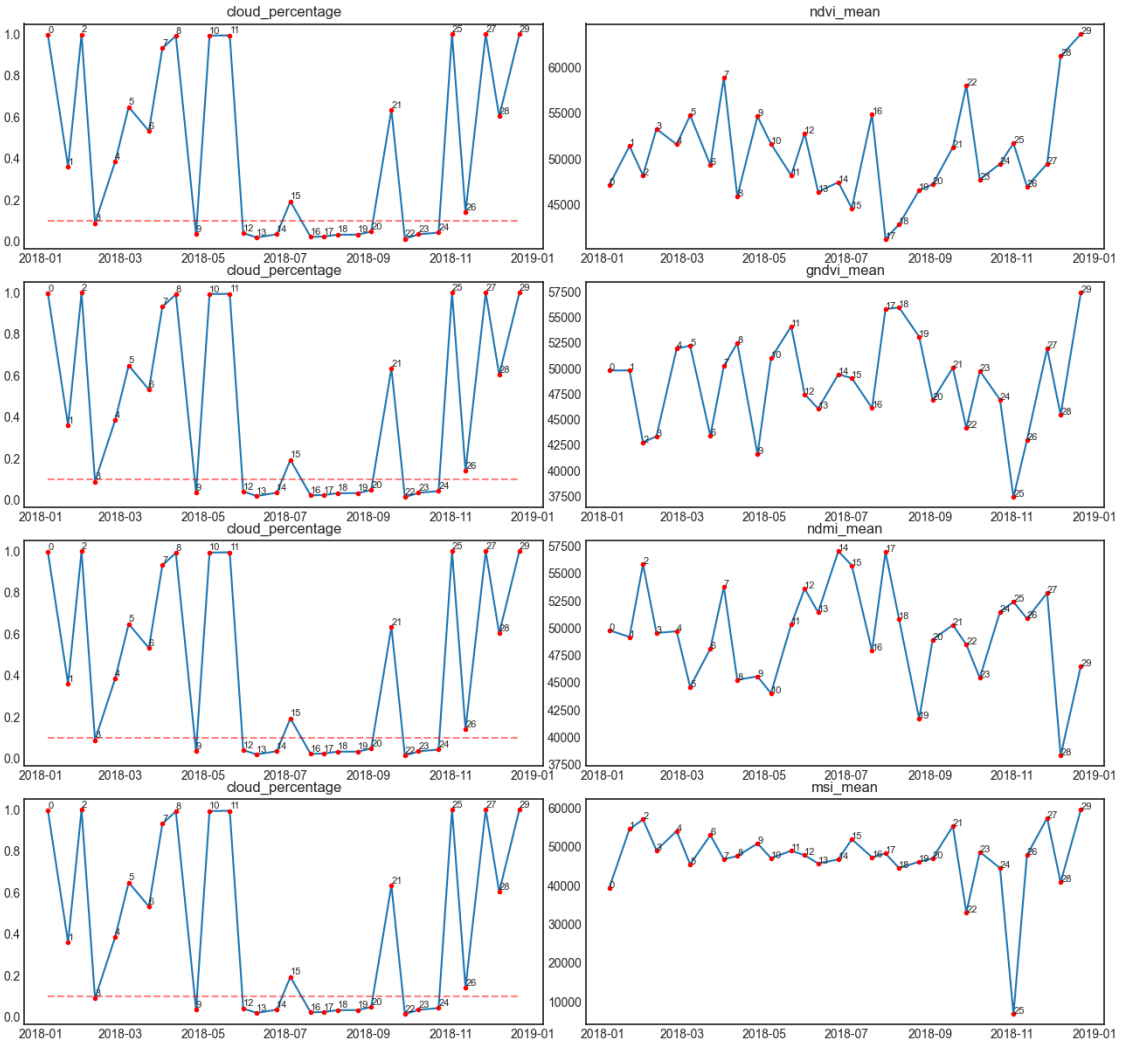


Figure 71: Cloud percentage and index mean value in all time instances (1).

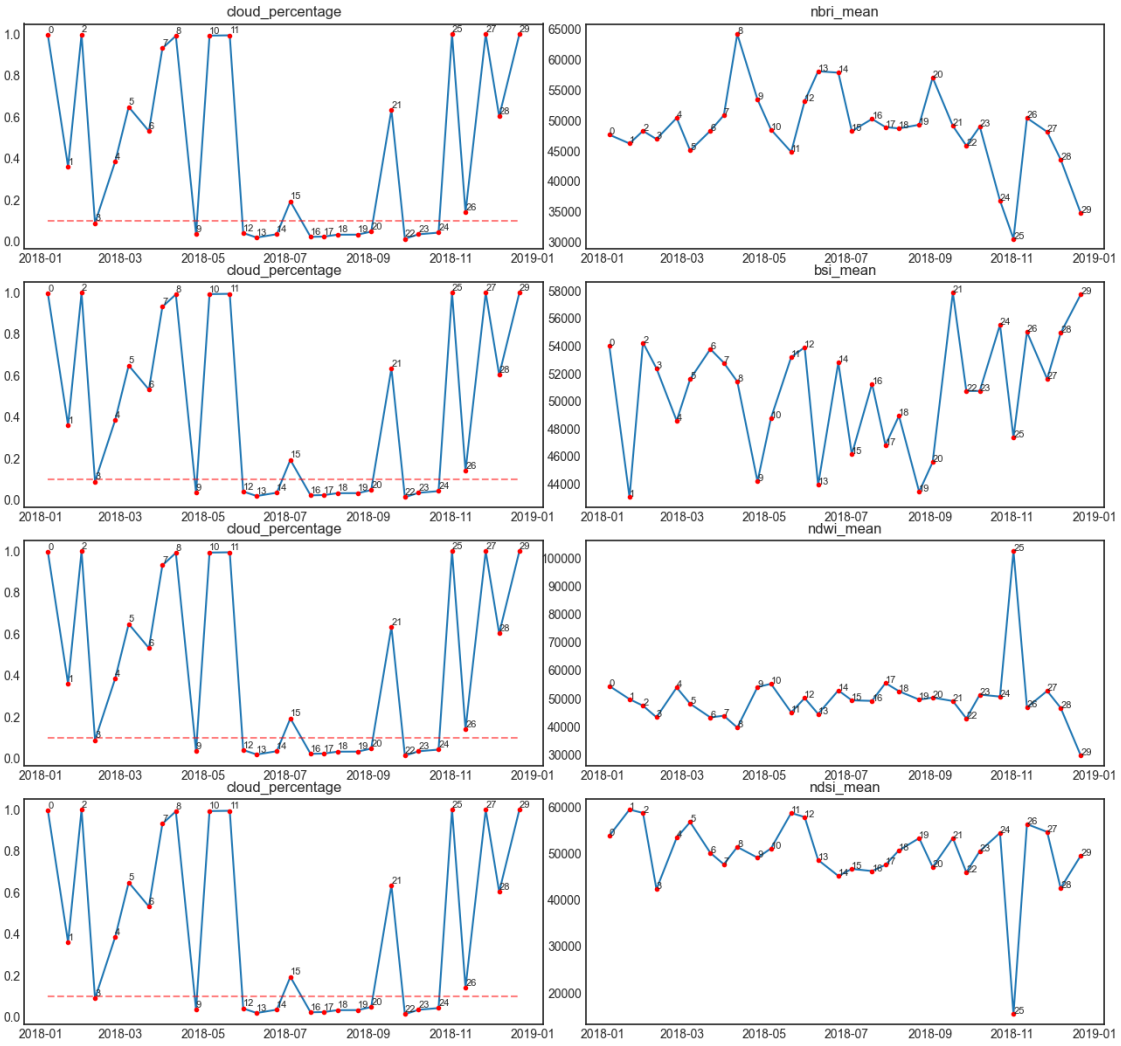


Figure 72: Cloud percentage and index mean value in all time instances (2).