

**Universitat de Lleida**

## **Economic Viability**

Made by  
*Oriol Alàs Cercós*

Delivery  
25<sup>th</sup> of May, 2022

Universitat de Lleida  
Escola Politècnica Superior  
Màster en Enginyeria Informàtica  
Technological Business Management and Entrepreneurship

**Professorate:**

Josep Escribà Garriga

# Contents

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Introduction</b>                         | <b>1</b>  |
| <b>2</b> | <b>Problem Analysis</b>                     | <b>2</b>  |
| 2.1      | Sentinel2 . . . . .                         | 2         |
| 2.2      | Related work and state-of-the-art . . . . . | 4         |
| 2.3      | Academic Datasets . . . . .                 | 23        |
| <b>3</b> | <b>Exploratory Data Analysis</b>            | <b>23</b> |
| 3.1      | Properties extraction . . . . .             | 24        |
| 3.2      | Web App . . . . .                           | 28        |
| 3.3      | Analysis . . . . .                          | 29        |
| 3.4      | Conclusions . . . . .                       | 29        |
| <b>4</b> | <b>Model architecture</b>                   | <b>29</b> |
| <b>5</b> | <b>Experiments &amp; results</b>            | <b>29</b> |
| <b>6</b> | <b>Conclusions</b>                          | <b>29</b> |
| <b>7</b> | <b>Bibliography</b>                         | <b>29</b> |

# List of Figures

|   |   |    |
|---|---|----|
| 1 | LeNet-5 architecture. . . . .   | 4  |
| 2 | Residual learning: a building block. . . . .  | 5  |
| 3 | Loss surfaces of ResNet-56 with/without skip connections. . . . .   | 5  |
| 4 | DSen2-CR model diagram. . . . .   | 6  |
| 5 | 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. . . . . | 7  |
| 6 | GAN model diagram. . . . .  | 7  |
| 7 | cGAN model diagram. . . . .   | 9  |
| 8 | Prediction results by McGAN with the synthesized cloud images . . . . .   | 10 |

|    |  |    |
|----|--|----|
| 9  | Prediction results by McGAN with the real cloud images . . . . .   | 10 |
| 10 | Comparison between the cloudy images, the outputs of McGAN and Pix2Pix-based models and the ground-truth image. . . . .              | 12 |
| 11 | Comparison between cloudy images, the outputs of McGAN versions and the ground-truth image. . . . .                                  | 12 |
| 12 | Failure cases in CloudGAN attempting to removal overly clouded images over-smoothening or completely failing. . . . .                | 15 |
| 13 | Failure cases in CloudGAN attempting to removal overly clouded images over-smoothening or completely failing. . . . .                | 15 |
| 14 | Generator $G_{\text{cloudy} \rightarrow \text{cloudfree}}$ of [1] . . . . .  | 16 |
| 15 | Generator $G_{\text{cloudy} \rightarrow \text{cloudfree}}$ of [2] . . . . .  | 16 |
| 16 | Transformer block architecture. <b>left block</b> , the encoder architecture, <b>right block</b> , the decoder architecture. . . . . | 17 |
| 17 | Multi-Head Attention block . . . . .   | 18 |
| 18 | Scaled Dot Product Attention Head . . . . .  | 19 |
| 19 | ViT architecture . . . . .   | 21 |
| 20 | L-TAE architecture . . . . .   | 22 |
| 21 | [20] model generator architecture . . . . .  | 22 |
| 22 | Laplacian variance depending on the slope used in sigmoid scaling method. . . . .  | 25 |
| 23 | Scaling methods . . . . .  | 25 |
| 24 | Sigmoid-scaled images with their RGB bands in different slopes. . . . .  | 26 |

## List of Tables

|   |   |    |
|---|---|----|
| 1 | Sentinel 2 bands . . . . .  | 2  |
| 2 | PSNR and SSIM of the single-image and multi-temporal model versions . . .   | 13 |
| 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. . . . . | 20 |

## **Acronyms**

**cGAN** Conditional Generative Adversarial Network.

**CNN** Convolutional Neural Network.

**GAN** Generative Adversarial Network.

**IR** Infra Red.

**L1C** Level-1C.

**L2A** Level-2A.

**McGAN** Multi-spectral Conditional Generative Adversarial Network.

**MLP** Multi-Layer Perceptron.

**NIR** Near Infra Red.

**NLP** Natural Language Processing.

**PSNR** Peak signal-to-noise ratio.

**RS** Remote Sensing.

**SAR** Synthetic Aperture Radar.

**SSIM** Structural Similarity Index Metric.

# 1 Introduction

Remote Sensing (RS) imagery is critical to perform challenges such climate change or natural resources management, including zone monitoring for reforestation, disaster mapping, 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 definitely needs a generic technique to detect and remove the cloudy region with an in-painting of the underlying scene.

Once set the problem, the goals of this master thesis have been:

- Search state-of-the-art solutions 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 make a comparison with the other solutions proposed.

## 2 Problem Analysis

### 2.1 Sentinel2

The analysis of the problem has been focused using Copernicus Sentinel2 constellation. Sentinel2 images are provided by two satellites, Sentinel 2A and Sentinel 2B, which orbit each other with a 180° phase shift. Generally, the acquisition of the images is more or less 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 100x100km<sup>2</sup> 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 ( $\mu\text{m}$ ) | 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.

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

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

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

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

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

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

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

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

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

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

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

**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.

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

**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.

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

## 2.2 Related work and state-of-the-art

Deep learning have been a popular and efficient technique to solve challenges from satellite imagery. 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 Multi-Layer Perceptron (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 1

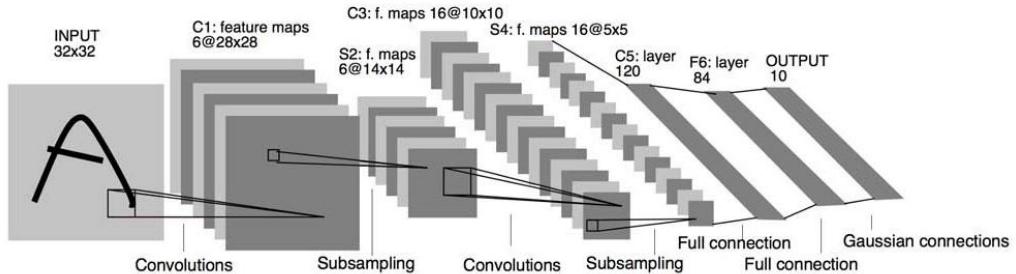


Figure 1: LeNet-5 architecture.

Being that told so, there are a lot of state-of-the-art using CNN. In [3], 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 [4].

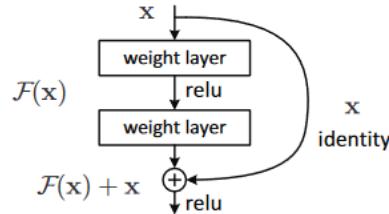


Figure 2: Residual learning: a building block.

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 3, networks without skip connections are harder to train as the loss surface is to hard to navigate. [5]

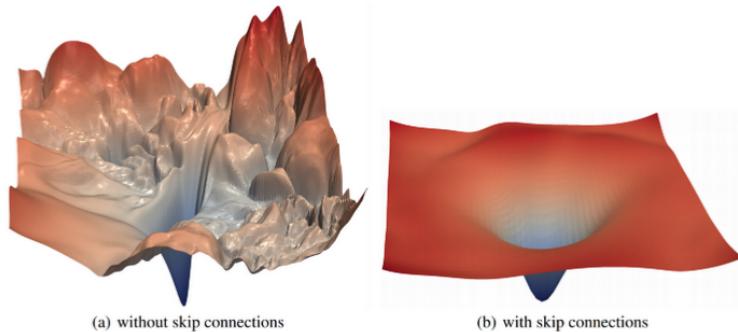


Figure 3: Loss surfaces of ResNet-56 with/without skip connections.

Similarly, in [6], 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 4. The output of DSen2-CR is a 13-channel layer, representing the thirteen bands from Sentinel2.

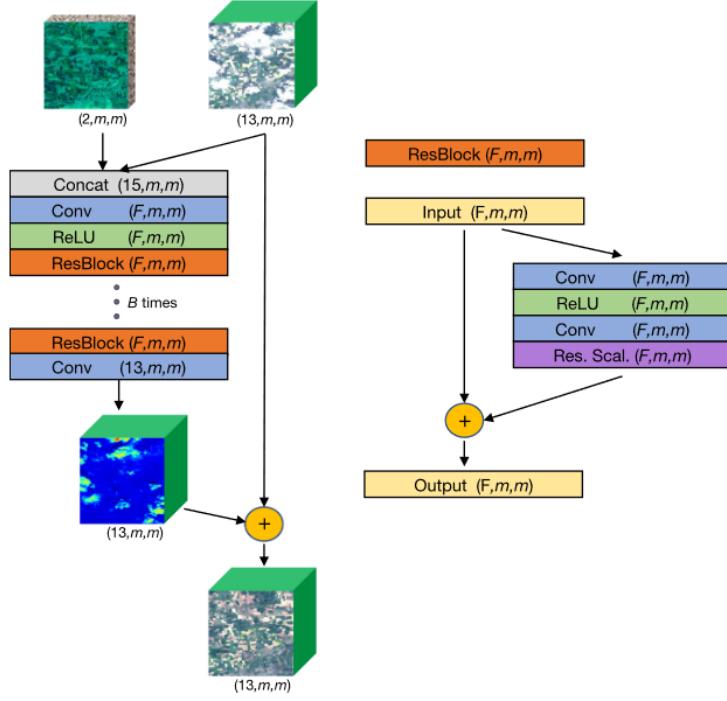


Figure 4: DSen2-CR model diagram.

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 5 that the lack of SAR data make the network less powerful.

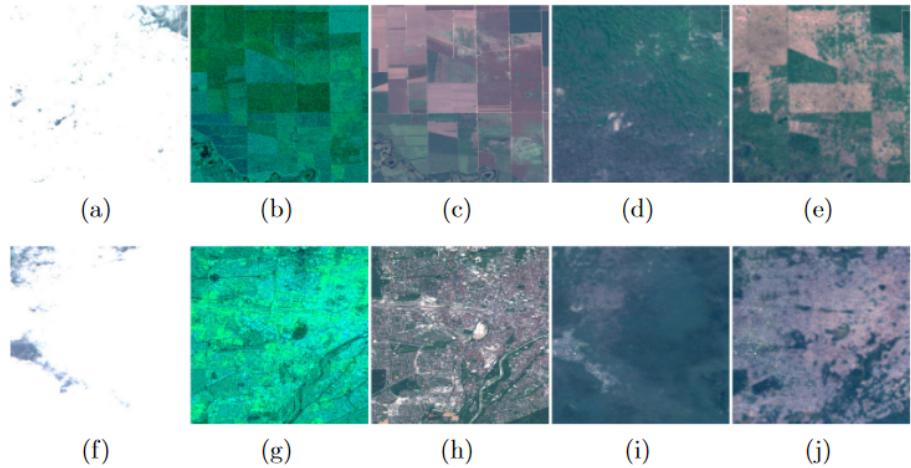


Figure 5: Examples of the influence of SAR data. Being (a, f) cloudy images, (b, g), SAR images, (c, h), the target cloudless images, (d, i), the synthesized images without SAR input data and (e, j) the predictions using SAR data as input.

Improvements have been found using Generative Adversarial Network (GAN) [7], which is a model architecture that belongs to the set of generative models. 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.

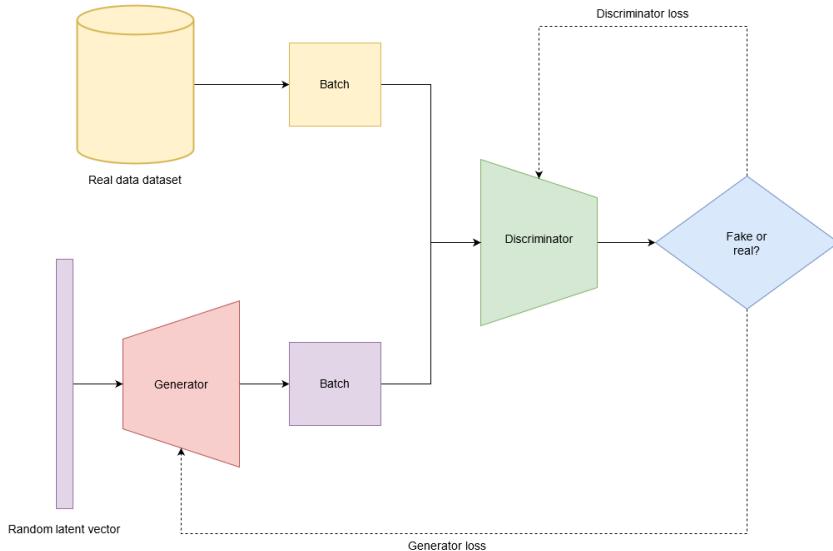


Figure 6: GAN model diagram.

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:

$$\begin{aligned} Loss_D(D) &= E_x[\log(D(x))] \\ Loss_G(G) &= E_y[\log(1 - D(G(y)))] \\ Loss_{GAN}(G, D) &= Loss_D(D) + Loss_G(G) \end{aligned}$$

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 7, 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.

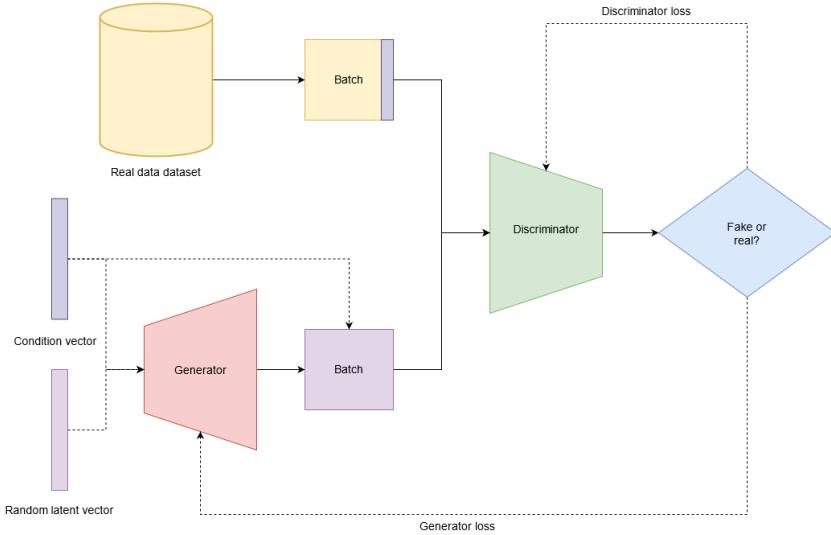


Figure 7: cGAN model diagram.

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 [8], they proposed a McGAN trained using RGB and NIR cloud-free bands as input and synthetic RGB 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 possess 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 8 and 9

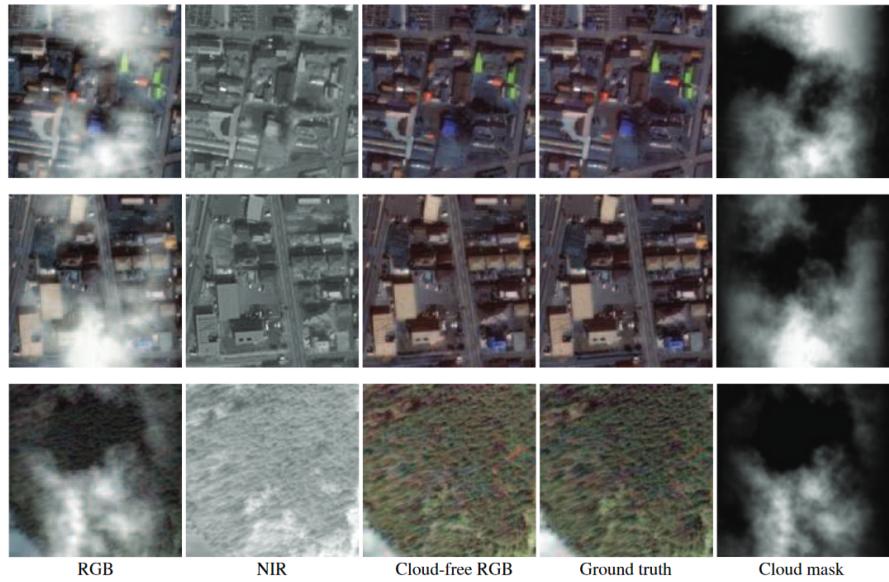


Figure 8: Prediction results by McGAN with the synthesized cloud images

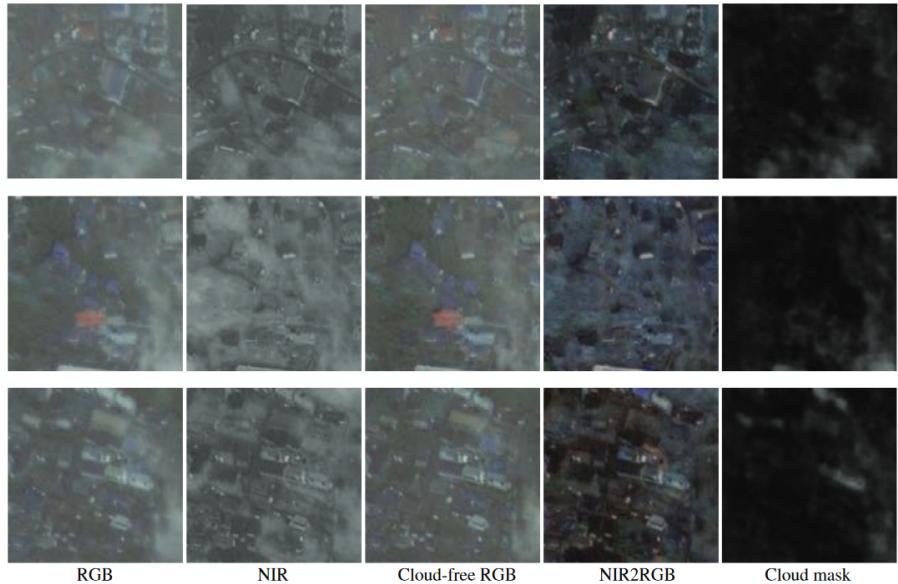


Figure 9: Prediction results by McGAN with the real cloud images

To overcome the need of synthesized data, [9] 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 [10] 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 [11]. 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 10 and 11. 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.

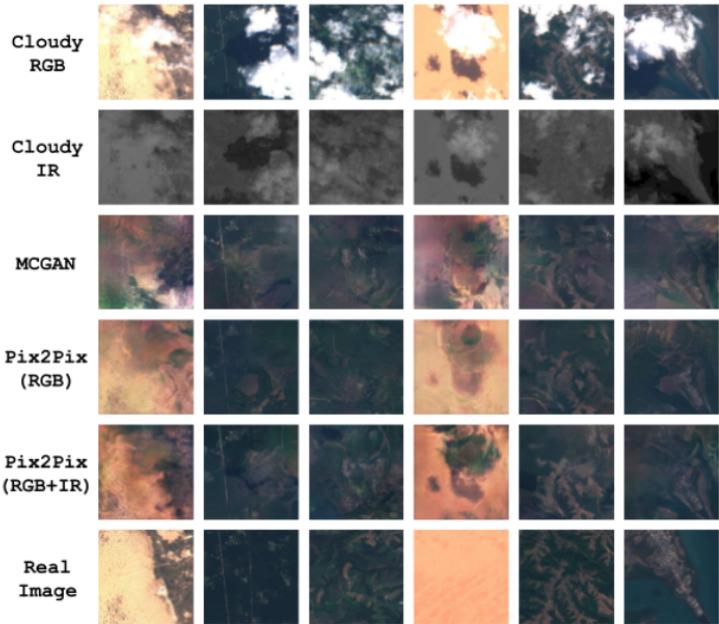


Figure 10: Comparison between the cloudy images, the outputs of McGAN and Pix2Pix-based models and the ground-truth image.

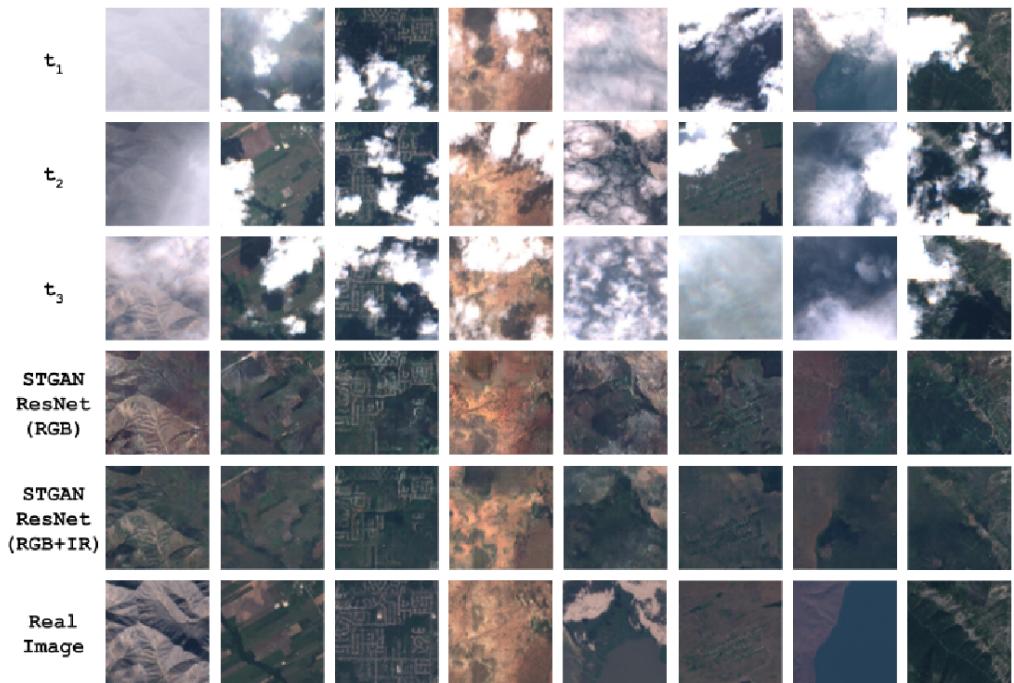


Figure 11: Comparison between cloudy images, the outputs of McGAN versions and the ground-truth image.

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 [12] is a CycleGAN [13] 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:

$$Y_A : \{y \mid y \text{ is a synthesized cloud obscured image}\}$$

$$Y_B : \{y \mid y \text{ is a synthesized cloud-free image}\}$$

$$Y = Y_A \cup Y_B$$

$$X_A \cap Y_A = \emptyset, \quad X_B \cap Y_B = \emptyset$$

The transformation functions from each generator can be expressed as:

$$G_A : \text{Generates } y \in Y_B \text{ from } z \in Z_A, \quad Z_A = X_A \cup Y_A$$

$$G_A : Z_A \longrightarrow Y_B$$

$$G_B : \text{Generates } y \in Y_A \text{ from } z \in Z_B, \quad Z_B = X_B \cup Y_B$$

$$G_B : Z_B \longrightarrow Y_A$$

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  $\lambda$  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 [12] 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 12 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.

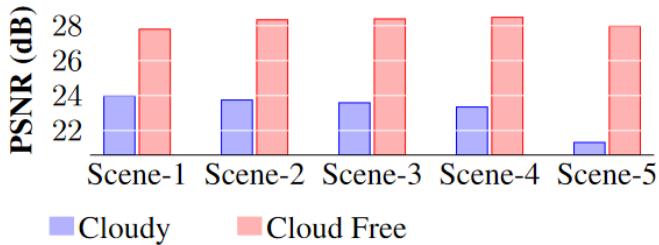


Figure 12: Failure cases in CloudGAN attempting to removal overly clouded images over-smoothening or completely failing.

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 13.

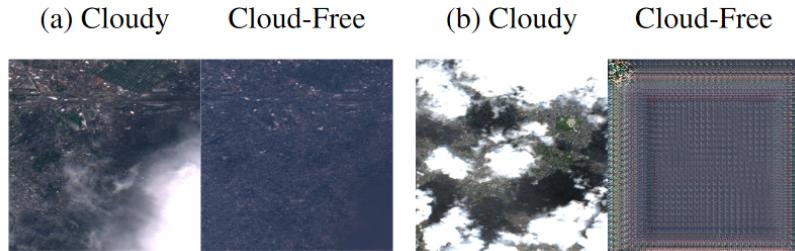


Figure 13: Failure cases in CloudGAN attempting to removal overly clouded images over-smoothening or completely failing.

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 14.

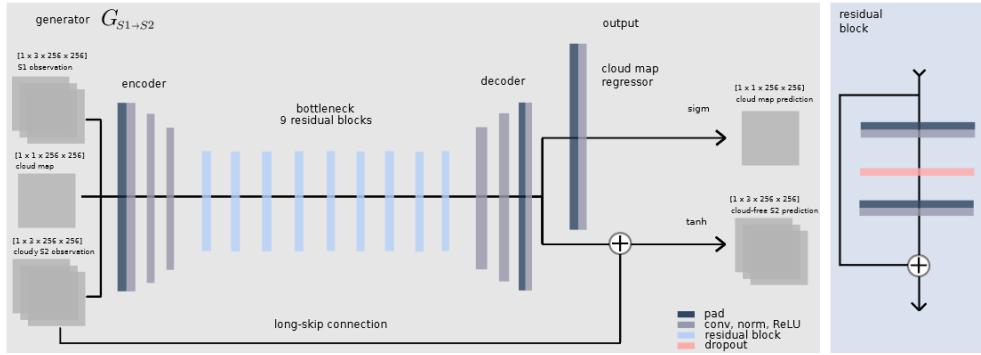


Figure 14: Generator  $G_{\text{cloudy} \rightarrow \text{cloudfree}}$  of [1]

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 MAE, RMSE, PSNR, SSIM and SAM were used to evaluate the generators.

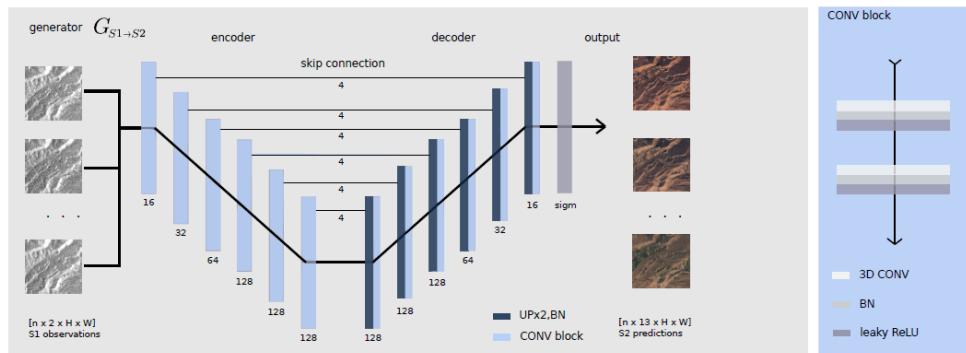


Figure 15: Generator  $G_{\text{cloudy} \rightarrow \text{cloudfree}}$  of [2]

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.

CNN have worked very well for cloud removal, but latest and disruptive state-of-the-art deep learning attention-based architectures [14] uncover new paths to achieve remarkable improvements and results. It has been demonstrated that transformers can excellently overcome challenges such Natural Language Processing (NLP) [15], Text-To-Image Generation [16] or Image Completion [17] with large datasets, great model size and enough compute.

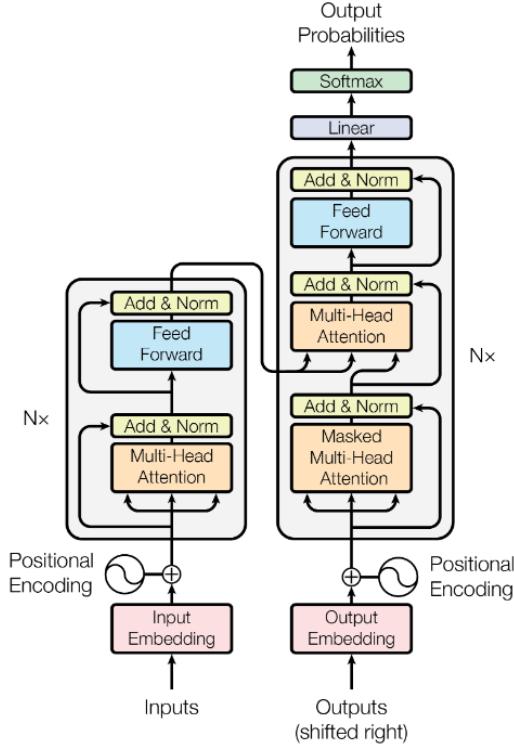


Figure 16: Transformer block architecture.  
**left block**, the encoder architecture, **right block**, the decoder architecture.

The transformer neural network is composed by an encoder-decoder architecture much like Recurrent Neural Network (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. [14] 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.

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

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/dmodel}}\right)$$

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<sup>1</sup> 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.

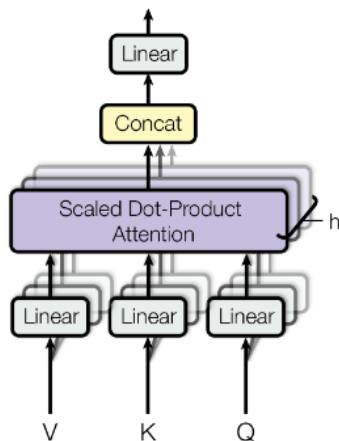


Figure 17: Multi-Head Attention block

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  $i^{\text{th}}$  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

---

<sup>1</sup>The normalization is done by each feature instead by each sample.

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.

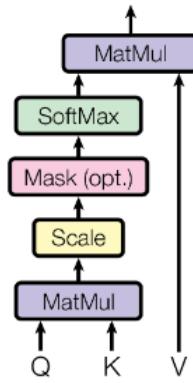


Figure 18: Scaled Dot Product Attention Head

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.

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

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, [18] 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.

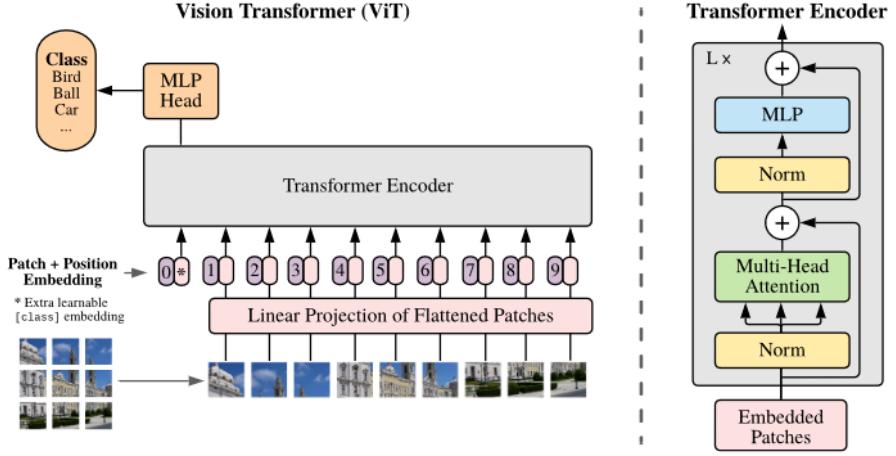


Figure 19: ViT architecture

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 image generation, it is proposed the ViTGAN with a Vanilla-ViT discriminator and a ViT-based generator, although a mixture of deconvolutional layers and global receptive fields from ViT might perform better.

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 [19] 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.

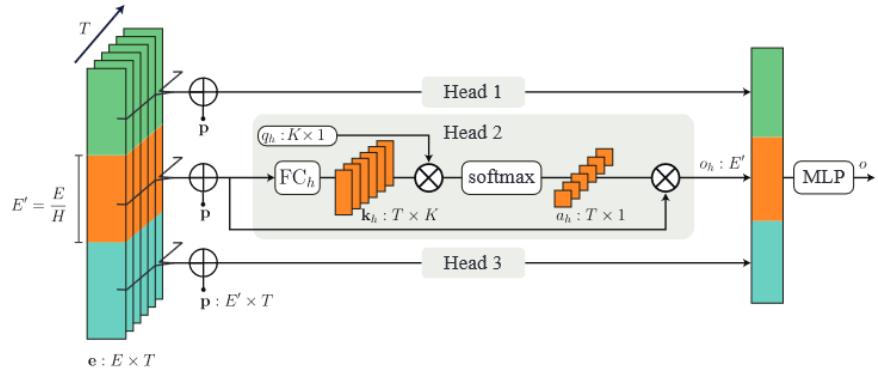


Figure 20: L-TAE architecture

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.

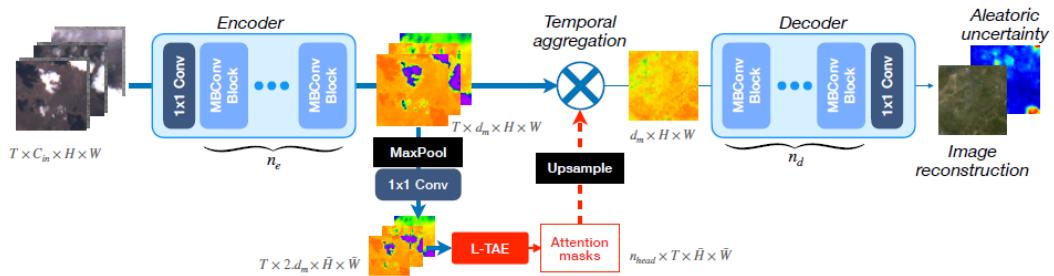


Figure 21: [20] model generator architecture

This architecture is then used in [20] to aggregate the information of temporal images without needing high computational resources. [20] 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.

### 2.3 Academic Datasets

## 3 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 EDA was aimed to provide insights into potential challenges that might be encountered during the development of the model.

The analysis proposes four 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 they 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 attempting not to create biases while extracting the data.
- Visualizing the data can provide valuable insights but a huge amount of data extracted can be messy and create a high computation demand.
- Understanding the limitations of the dataset, such as noise, class imbalance, or variations in image quality, allowed 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.

### 3.1 Properties extraction

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 I did not want to modify the images to the point that comparison and 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.

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

- 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<sup>th</sup> percentile, it will be 1 (or the maximum of the scaled vector). If the pixel value is under the 2<sup>nd</sup>, 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<sup>th</sup> percentile and the 2<sup>nd</sup> the minimum.

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

- 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 seen in 22 and 24.

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

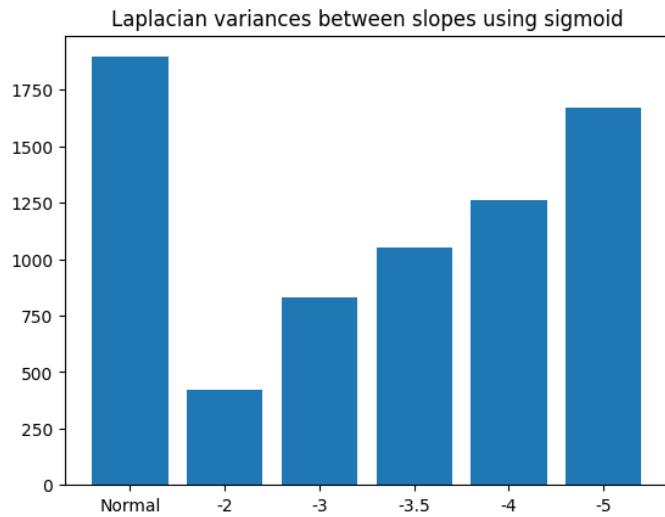


Figure 22: Laplacian variance depending on the slope used in sigmoid scaling method.

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

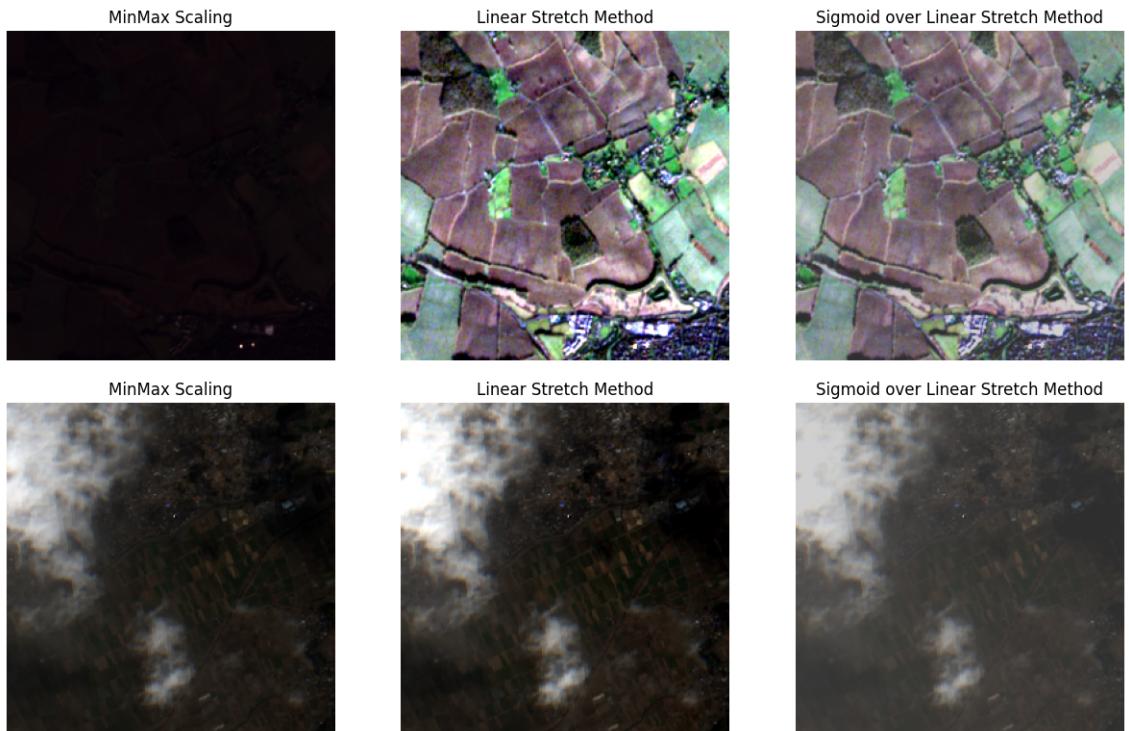


Figure 23: Scaling methods

Although it seems in 23 that the best normalization is to apply is sigmoid after the linear strech method, we can see in 24 that this would affect the study of the properties since the data would be so compressed that the analytical results would not be as precise as it could. Therefore, the best method to apply 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.

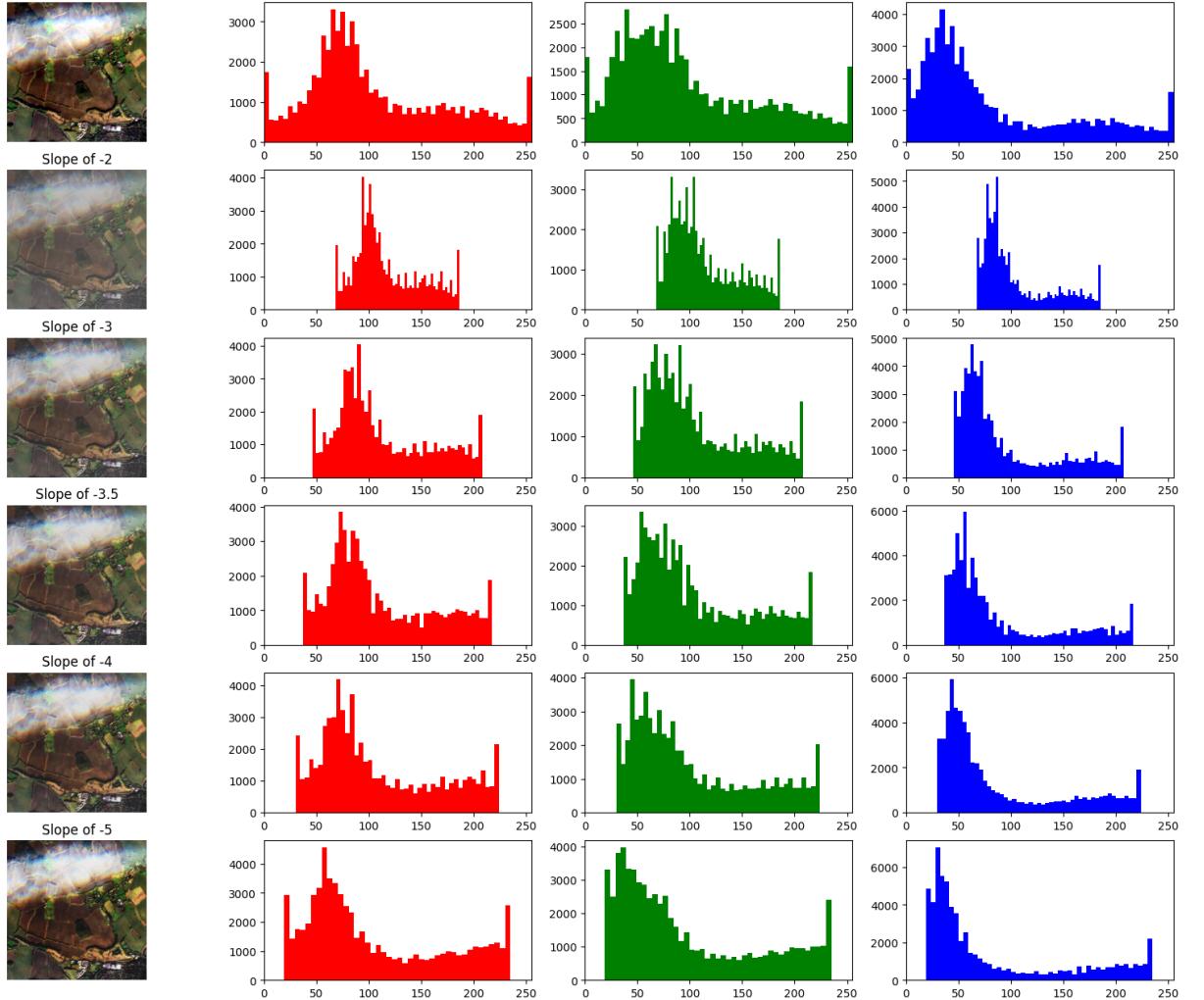


Figure 24: Sigmoid-scaled images with their RGB bands in different slopes.

The properties compared 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.
  - 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). KDE helped to visualize the probability density function of each band, aiding in the identification of underlying patterns or modes.
2. True Color Image (TCI) 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.

### 3.2 Web App

Visualizations played a crucial role in revealing the inherent structure and characteristics of the dataset. By visually inspecting the images, patterns and trends specific to cloud formations were discerned, aiding in the development of strategies to tackle cloud removal effectively. Furthermore, feature engineering techniques related to computer vision and, specifically, satellite imagery 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. 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.

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 allowing developers to write Python code that defines the various components of their web application, such as widgets, plots, and text. These components are then rendered in a web browser, creating an intuitive and 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.

### 3.3 Analysis

### 3.4 Conclusions

## 4 Model architecture

## 5 Experiments & results

## 6 Conclusions

## 7 Bibliography

## 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] 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.
- [4] 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.
- [5] 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.

- [6] 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.
- [7] 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.
- [8] 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.
- [9] Vishnu Sarukkai, Anirudh Jain, Burak Uzkent, and Stefano Ermon. Cloud removal in satellite images using spatiotemporal generative networks. *arXiv preprint arXiv:1912.06838*, 2019.
- [10] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017.
- [11] 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.
- [12] 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.
- [13] 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.
- [14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz 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.

- [15] 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.
- [16] 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.
- [17] 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.
- [18] 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.
- [19] Vincent Sainte Fare Garnot and Loic Landrieu. Lightweight temporal self-attention for classifying satellite images time series. *arXiv preprint arXiv:2103.01157*, 2021.
- [20] 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.