Universidad de la República
Facultad de Ingeniería

DeepCloud: Intra-day Satellite Prediction of Cloudiness Using Deep Learning Strategies

Memoria de proyecto presentada a la Facultad de Ingeniería de la Universidad de la República por

Ignacio Camiruaga[b], Andrés Herrera[a], Franco Mozo[a]

en cumplimiento parcial de los requerimientos para la obtención del título de Ingeniero Eléctrico[a] e Ingeniero en Computación[b].

Tutores

Rodrigo Alonso Suárez	Universidad de la República
Alberto Castro	Universidad de la República
Franco Marchesoni	École Normale Supérieure Paris-Saclay

Tribunal

Agustín Laguarda	Universidad de la República
Pablo Musé	Universidad de la República
Facundo Benavides	Universidad de la República

Montevideo
Wednesday 15th June, 2022

DeepCloud: Intra-day Satellite Prediction of Cloudiness Using Deep Learning Strategies, Ignacio Camiruaga, Andrés Herrera, Franco Mozo.

This thesis was prepared in LaTeX using the iietesis (v1.1) class.
Contains a total of 125 pages.
Compiled on Wednesday 15th June, 2022.
http://iie.fing.edu.uy/

"Be comforted, dear soul! There is always light behind the clouds."

Louisa May Alcott, "Little Women"

Acknowledgements

We would like to thank the people and organizations that made this project possible. Rodrigo Alonso Suárez and Franco Marchesoni apart from the tutoring had the initial idea for this project, and also the confidence in us to carry it out. Also, Alberto Castro found the project meaningful and decided to join us, helping in many aspects along the way.

The Solar Energy Laboratory (LES) [1] provided vital aspects such as data and infrastructure. Previous work done by members of the LES made a road for us to follow and facilitate the learning process. We were able to present our work in multiple spaces thanks to the LES, and being able to do it on behalf of the organization makes us very proud. Finally, the data used are all stored in the LES server, to which we were given access to work freely. Another organization worth mentioning is the UdelaR, for making available to us high-level education, allowing us to reach this moment in our lives.

The data-intensive experiments that were required for this project were done by using the ClusterUY infrastructure [2]. This is a high-performance national supercomputing center in Uruguay whose resources are freely available for scientific purposes.

[1] LES site: http://les.edu.uy/
[2] ClusterUY site: https://cluster.uy

To our family, friends, and professors, who gave unconditional support throughout this journey.

Abstract

This project analyzes deep learning techniques applied to satellite-based cloudiness prediction, a vital component of a solar forecasting solution. The techniques can learn from a dataset to make extrapolation into the future of a sequence of images, a process usually named satellite nowcasting. In this way, intra-day image forecasting is addressed up to 5 hours into the future, with a 10-minute periodicity. The images used are from the GOES-16 geostationary satellite, covering a large portion of southeast South America, including Uruguay, the main region of interest. The new deep learning techniques are compared against strong baselines in the field, such as the persistence and fine-tuned Cloud Motion Vectors strategies, which were previously analyzed for this region in recent studies. Several state-of-the-art architectures are implemented and evaluated over different well-known computer vision metrics as well as forecasting metrics. Our results showed the ability of deep learning models to account for complex atmospheric dynamics and make accurate predictions in a short time span. The main contribution is a deep-learning model based on the U-Net architecture that surpasses in performance all the other state-of-the-art models implemented on this dataset. The new model is presented along with detailed ablation studies and thorough evaluations, that shed light on the behavior and many potential variations of the deep learning solutions.

Contents

Chapter 1

Introduction

Solar energy is the most abundant energy source available to the planet [1], consistently exceeding experts' projections. One of the major challenges mankind faces in the 21st century is being able to exploit solar energy to generate a medium and long-term energy transition. Although its large availability, which for instance is more than 1000 times the World's energy consumption [1], its high technology maturity level, and its low prices, solar energy is still a very small portion of the World's energy mix. There are several explanations for this, including its low spatial density, its non-availability during night time and the lack of built-in storage of its main technology (solar photovoltaics), among others. This work addresses one of its drawbacks at the current state of the art: its relatively low predictability in very short time scales due to its high intermittency. This intermittency is caused by clouds, whose irregular shape, movement, development, and extinction are difficult to predict. Cloudiness prediction in very short-term lead times based on geostationary satellite images is the focus of this work, assisting the intra-day solar forecasting block up to 5 hours ahead. Efficiently solving this problem involves good forecasting. We work with data from the Solar Energy Laboratory (LES) of the Universidad de la República and provide a detailed study of deep learning methods applied to the task.

The remainder of this chapter is organized as follows. Section 1.1 and Section 1.2 presents the importance and drawbacks of solar energy technologies. Section 1.3 describes the usefulness of solar energy forecasting tools and Section 1.4 provides a brief background of the state of the art, with a focus on the local R&D line. Section 1.5 gives a short introduction to the initial references on which this work is based. Then, Section 1.6 introduces the objectives of this work and how it connects with the local research. Finally, Section 1.7 and Section 1.8 provide a summary of the contributions of this work and the outline of the full documentation, respectively.

1.1 On the importance of solar energy

Solar energy is one of the different types of renewable energy that contributes to the electricity grids in many countries. What distinguishes solar from the other kinds of energies is the enormous potential it has, as the Sun delivers annually approximately a thousand times more than what it is consumed on the whole planet [1]. Uruguay is not estranged from solar energy, being one of the countries in the world which gives more relative importance to it.

The Intergovernmental Panel on Climate Change (IPCC) is the United Nations body responsible for providing assessments and updated information on climate change

for governments, policymakers, and decision-makers worldwide. In a report about Renewable Energy Sources and Climate Change Mitigation [2], a full chapter is dedicated to solar energy. In it, an exhaustive study about the viability of this type of energy source is conducted. The main takeaways are that it has the largest technical potential of all energy sources and could expand dramatically in the decades to come, as production and deployment costs tend to decrease. Another organization related to the topic is the International Energy Agency (IEA), which calculated that in the last years, solar energy was the renewable source of energy with more installations annually [3]. The IEA published during 2014 a roadmap for solar photovoltaic energy [4], mentioning that one of the main issues slowing down the growth is the prediction of the available energy.

The Uruguayan Ministry of Industry, Energy, and Mining has made an important focus on solar energy by considering it while making medium and long-term policies, planning the advancement from 2005 to 2030. One of the goals is to diversify the electric grid by incorporating renewable energy produced locally. Between 2012 and 2017, solar photovoltaic energy production had en exponential growth, going from approximately 0 MW to 242 MW produced annually [5]. The continuous growth is done not only by large generators but also by microgenerators. In July 2010, a decree [6] allowed the small consumers to install solar panels to generate energy for their consumption and sell the surplus (with some limitations) to the Uruguayan administrator of power plants and electric transmissions (UTE).

In 2017, the second version of the Uruguayan Solar Map (MSUv2) was published, giving a better knowledge of the country's solar resources with significantly higher temporal and spatial resolutions, and lower uncertainty [7]. A solar map is useful for determining if a particular region receives enough solar energy to make an investment in infrastructure and be profitable. As the results show, Uruguay is a country rich in solar resources, so it is of great interest to be able to exploit these resources. Even more, considering that the geographic location is not rich on fossil fuels such as oil and natural gas, or uranium deposits.

1.2 Drawbacks of solar energy

Solar energy has some well-known drawbacks. These are the focus of state-of-the-art research. First, it can not produce energy on-demand with firm capacity as it is driven by meteorological conditions. This is why solar is sometimes combined with storage solutions. Second, the energy is available in cycles: for a given site, there is no solar energy during the night and it peaks during midday assuming clear-sky or average conditions. Lastly, the energy to be generated has high intermittency, whose effects on the rest of the electrical system need to be mitigated.

Of the three drawbacks that were mentioned, only the sensitivity to atmospherical conditions introduces uncertainty into the energy dispatch and planning. This happens because clouds, ashes, and aerosols can generate significant changes in solar irradiance over photovoltaic power stations. Even small clouds can cause large drops in the global horizontal irradiance (GHI) on a given site. The sensitivity to clouds makes the solar generation reproduce the short-term variability of the atmosphere, introducing uncertainty in its current and future availability. This uncertainty is present at many different timescales, from the year level to some minutes-ahead predictions.

1.3 Motivation behind solar forecasting

The need for reduced uncertainty makes good forecasts a key component of efficient solar solutions. This facilitates the integration of solar energy into electricity grids. Having a reliable prediction of the solar resource availability that the region will have during the day allows for better decisions regarding energy dispatch. This reduces costs associated with the variability, supply/demand balance, and backup generation. Thus, good forecasts help preventing economic losses and enable to establish better sell/buy prices in the electricity market.

In particular, intra-day solar forecasting is relevant for load following mechanisms and electricity market transactions applications. Usually, forecasts for horizons longer than intra-day rely on Numerical Weather Prediction (NWP) models, but these are not the best option when doing intra-day forecasts. One reason is that they take a substantial amount of time and computing power to run, so intra-day forecasts are not frequently updated [8]. The need for some hours to yield a forecast jeopardizes the forecast itself: a forecast provided at time t is using the information available at time $t - r$, where r is the inference time. In other words, NWP methods do not take advantage of the most recent (thus relevant) information but use instead the information available at the time when the inference run started. Because of the previously mentioned points, predictions based on satellite images are the preferred method for intra-day solar forecasting, for a five hours horizon.

1.4 Solar energy forecasting in Uruguay

For the reasons mentioned in Section 1.3, the research on intra-day solar forecasting, conducted by members of the LES, focuses on satellite image-based methods. Recent works study Cloud Motion Vectors (CMV) techniques to generate predictions. A master's thesis dissertation [9] works on the two steps needed to predict GHI. The first is predicting the satellite image, for which it studies CMV performance, and the second step is using that image to predict the photovoltaic energy production for several power plants. A more recent work [10] also studies both steps but makes an emphasis on the first step by comparing different optical flow algorithms to calculate the CMV. The current operating system in the LES servers [11], which provides real-time intra-day predictions to ADME, is based on CMV techniques, and is of public access [1].

1.5 Baseline related works

Other works share the same focus as the one presented in this documentation. Specifically, two articles [12] [13] try different deep learning models for solar and precipitation forecasting respectively. These articles were taken as a first reference, although the satellite information and aims of the works are not the same. A paper was published at an advanced stage of this project which presented a state-of-the-art architecture for solar irradiance short-term forecasting, named IrradianceNet. Due to its high relevance, this architecture was studied in this project. A much more detailed description of the related work is provided in Chapter 3 with up-to-date papers on the field.

[1] GHI predictions by the LES: http://les.edu.uy/online/pronostico.php

1.6 Objectives

This project aims to expand the current satellite cloudiness forecasting R&D line in Uruguay through developing deep learning methods. By doing so, the results should answer whether these methods have similar performance or even out-perform the current solution the LES is providing. It is expected to test many state-of-the-art deep learning models to find the most suitable in terms of accuracy and applicability to a real-time solution.

1.7 Contributions

The detailed research presented and the results obtained are considered to be a contribution to the area of intra-day solar forecasting. This might be the only work with a deep learning approach that uses high-resolution images of size 1024×1024 over the South-American region and with a very high 10 minutes cadence. Numerous ablation studies conducted provide a roadmap for future research in satellite cloudiness forecasting and related problems. The result is a new modification to a well-known architecture such as the U-Net, which is capable of outperforming the baseline models and other state-of-the-art deep learning models. This variation of the U-Net has a low number of parameters, which translates into short inference times. A study on perceptual metrics shed light on the flaws that convey the oversimplification of complex systems such as human vision, and the need for further research. For the latter, a first approximation on the use of GANs on satellite images provides a starting point. Another major contribution is the training and evaluation of IrradianceNet, a state-of-the-art model, on a dataset with images four times the size and higher spatial resolution compared to the one used in the original publication. This yields a better understanding of its capabilities and limitations, mainly focused on the patch-based approach.

1.8 Detailed outline

This project's documentation is organized as follows. In Chapter 2, we present the dataset composed of GOES-16 visible channel images. These data were preprocessed with occasional inpainting and a normalization step that accounts for the Sun's position. Afterward, specific filters were applied to remove outliers and unnecessary data. The splitting of the remaining data into training, validation, and test datasets is described in the same chapter.

Chapter 3 presents an in-depth review of related work, comprising both the solar forecasting field and the deep learning literature. For the latter, the different models are analyzed and the context of the works is compared with ours. This chapter also introduces the baseline models and deep learning models tested in Chapter 4, along with the metrics used to evaluate the performance of the mentioned models.

Chapter 4 describes the implementation with necessary modifications, training, and selection of models. The experiments conducted throughout the project are presented with their results and an explanation in this chapter.

The selected models and the most relevant findings are evaluated over a dataset of previously unseen images, yielding the results we present in Chapter 5. Chapter 6 includes the main conclusions of our work based on the results, a recommendation on future steps if this project was to continue, and a group statement about the experience doing the project.

Chapter 2

Database Description and Preprocessing

In this chapter we describe the data to be used in this project. The data are geostationary satellite images registering the visible channel of a southeast region of South America every 10 minutes. An exploratory analysis of the data is presented along with the preprocessing pipeline. A conclusion made from the analysis is the need to reduce the size of the images so three distinct datasets are generated, named MVD, URU, and REGION3. These datasets are split into sub-datasets for proper model training and later testing. Working with data so big in size (in the order of terabytes) required careful handling, which is described in the last section.

2.1 Description and analysis

2.1.1 LES Database

The LES has internal databases that include GHI ground measurements in Uruguay and other countries, satellite images from different satellites of the Geostationary Operational Environmental Satellite (GOES) series, and other data such as photovoltaic power output, temperature, etc. In particular, the LES has developed and administrates the only satellite image infrastructure in the country that can receive, process, and store the high volume of the GOES series. The reception is done in real-time and assists in the generation of several solar radiation satellite services that the lab provides on its servers.

The data used in this work are composed of visible channel images taken by the GOES-16 satellite, a meteorological geostationary satellite administrated by the US's National Oceanic and Atmospheric Administration (NOAA). This satellite is part of the novel generation of NOAA's geostationary satellites, providing enhanced images' space and time resolution, and new spectral bands. It is located at the GOES-East position (-75°W meridian) and is fully operational since January 2018. The images are made public by NOAA free of cost using different means, including satellite reception and several web services.

The images are taken in Full-Disk mode, meaning they cover the whole face of the Earth that the satellite is able to see, as seen in Figure 2.1. These kinds of images are captured and made available every ten minutes since the year 2019. The LES downloads these images in almost real-time (with a 15-20 minutes delay) and processes

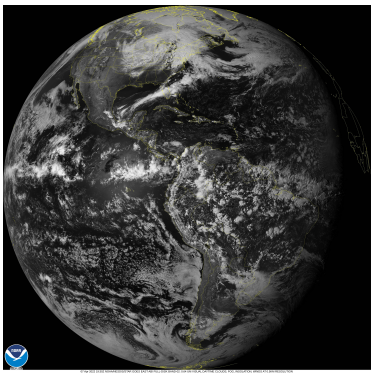


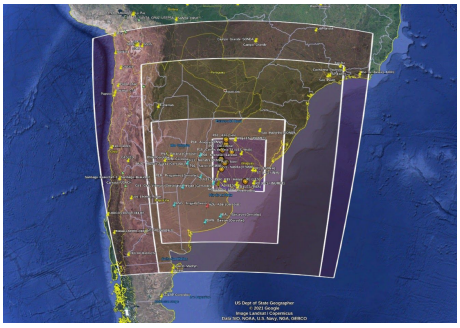
Figure 2.1: Full-Disk image of the Band 2 channel, taken by the satellite GOES-16. Figure taken from GOES Image Viewer.

the images to map the irregular space acquisition of the original images to a regular equally-spaced grid. Due to a limited storage capacity, only Full-Disk original images and some crops of the processed images are saved in the LES database, containing a great part of the South American region. These crops' regions are shown in Figure 2.2 (a), being the largest the one that was used in this work, as shown in Figure 2.2 (b). For better visualization, this second image is constructed by calculating the median value pixel-wise using all the images from a single day considered to have few clouds with the objective to show a clearer picture of the region. As we are going to use 10-minutes images of the new GOES-East satellite, dealing with images of 2020 only was considered enough data.

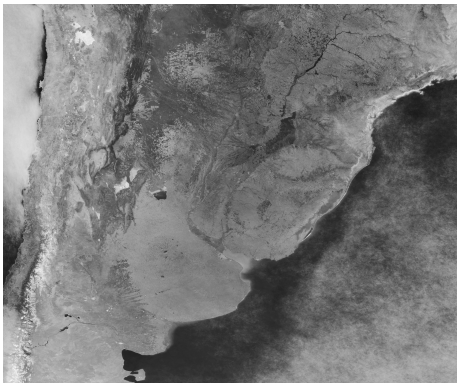
The reason why the visible channel is useful to detect clouds, is that clouds usually reflect more light than the ground, appearing brighter in the image. This does not hold for high albedo terrain, e.g. areas containing snow or salt. An example of the appearance of a cloud on an image is shown in Figure 2.3.

The LES has some default processing from which images and validity masks are obtained, along with region-specific metadata. Expecting an image every ten minutes, for each day the number of images should be equal to $6 \times 24 = 144$, some days the total number is less because of operational issues with the satellite. The images from the visible channel are 2501×3001 pixels large. The range of the Earth's surface covered by this image crop is 25° in latitude and 30°in longitude. Given the number of pixels, it is determined that each pixel has a spatial resolution of $0.01° \times 0.01°$ degrees. Making an approximation that 1° expands through approximately 110 km both in latitude and longitude, it gives that each pixel represents on average a size of the land of 1100 m $\times$ 1100 m. This approximation gives an idea of the pixel size, although it is not taking into detail the non-spherical shape of the Earth nor the fact that the region is not rectangular. Indeed, pixels closer to the equator and the satellite's nadir correspond to a smaller surface than those further away from them.

Complementary to the data offered by the LES, an elevation map of the region [14] was used for some experiments. In Figure 2.4, the most outstanding area is the Andes mountains, reaching heights of approximately 6000 meters above sea level. Note that the eastern side of the South-American continent shown in Figure 2.4 is referred to as Pampa Húmeda, and presents moderate to low heights, e.g. not surpassing 515 meters inside Uruguay. The majority of the Pampa Húmeda is considered warm, temperate, humid, and with hot summers [10]. This region is characterized by a challenging mesoscale convective system [15] [16], and intermediate short-term variability of the



(a) The biggest marked area corresponds to the South American region covered by the cropped images stored in the LES database.



(b) Image generated by calculating the median value pixel-wise using all the images from a single day considered to have few clouds. The objective is to show a clearer picture of the region.

Figure 2.2: Region Covered by the images stored in the LES database.

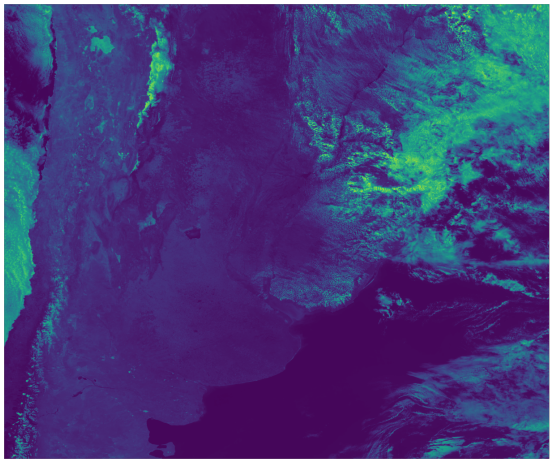


Figure 2.3: Example of a processed image. Corresponding to October 31, 2020 at 17:20 (UTC-0).

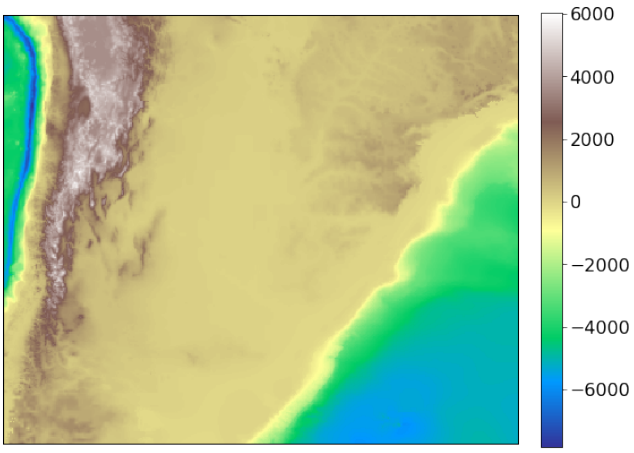


Figure 2.4: Elevation map (in meters) of the region covered by the images on the Dataset.

solar resource [17].

2.1.2 Preprocessing

There are two main issues to assess or inspect in the LES image database. First, some of the images present corrupted regions as a result of problems in their acquisition. Secondly, the images are not independent of the incidence angle of the sunlight.

The first issue is corrected before the second one. The regions, as masked in the database, are filled with valid values. Specifically, we run the Navier-Stokes inpainting method [18], implemented in OpenCV [19]. The percentage of images affected by the inpainting is 1.2%, but only 0.12% have more than 1% of the pixels modified.

The second issue is related to the nature of the images. The images in the LES database represent the reflectance factor, a dimensionless quantity that normalizes the measured radiance observed by the satellite's radiometer by the maximum value that the sensor can detect. In these images the illumination of the Sun over the Earth is observable, i.e. the pixels in the Sun's direction appear brighter than the others. This is a known geometrical effect associated with the reflection of the Sun's radiation on the Earth's surface. The images can be converted to Earth reflectance (or Earth albedo) by conducting a normalization step which makes them roughly invariant to the sunlight incidence angle. This normalization is done by dividing the images by each pixel's cosine of the solar zenith angle, $\cos\theta_z$. This is done as follows: given a planet reflectance image as in Figure 2.5(a), by the time of day and day of the year the image was captured the cosine of the solar zenith angle for each pixel is calculated resulting in an image composed of these values, named cosangs (Figure 2.5(b)). Three regions are defined in this image. The first is where values are above 0.15, the second is for values in between [0, 0.15], and the third is the rest of the values. These regions are shown in Figure 2.5(c). To do the calculation each pixel is mapped to the geographic latitude and longitude it represents. The reflectance factor image is divided pixel-wise by the cosangs, only on the pixels where the mask is greater than 0.15 and the rest of the values are set to zero, to obtain Figure 2.5(d). The positive values correspond to the daylight part of the image with solar altitude greater than $\simeq 8.5^\circ = 90^\circ - \arccos(0.15)$.

The reflectance image is also divided by the cosangs values in the region where the values are between [0, 0.15]. The values in this region are clipped to be between zero

and the mean of the normalized image in daylight pixels to obtain Figure 2.5(e). This region is called the terminator (also known as the "twilight zone") and it separates day and night. It is generated by the bending of the sunlight when interacting with the atmosphere [1]. The final normalized image is obtained as the sum of the normalized image where cosangs pixels are greater than 0.15 and the normalized image where these are in between [0, 0.15]. The result is shown in Figure 2.5(f). Figure 2.6 shows an example of an image before and after normalization.

After normalization, some images might contain pixels with values higher than 100. This is not desired as pixel values originally are in the [0, 100] range. This phenomenon is observed mostly in images at the start and end of the day. To solve the issue, pixel values are clipped to 100. Another method tested but found worse, namely replacing outliers with the mean, is shown in Appendix A.1.

2.2 Data Filtering

With a separation of ten minutes between images, each day should have 144 samples. As mentioned, the number of images per day can be less because of operational issues with the satellite. Given that 2020 was a leap year, $366 \times 144 = 52704$ images are expected, but the available dataset has 52314 images (390 missing images).

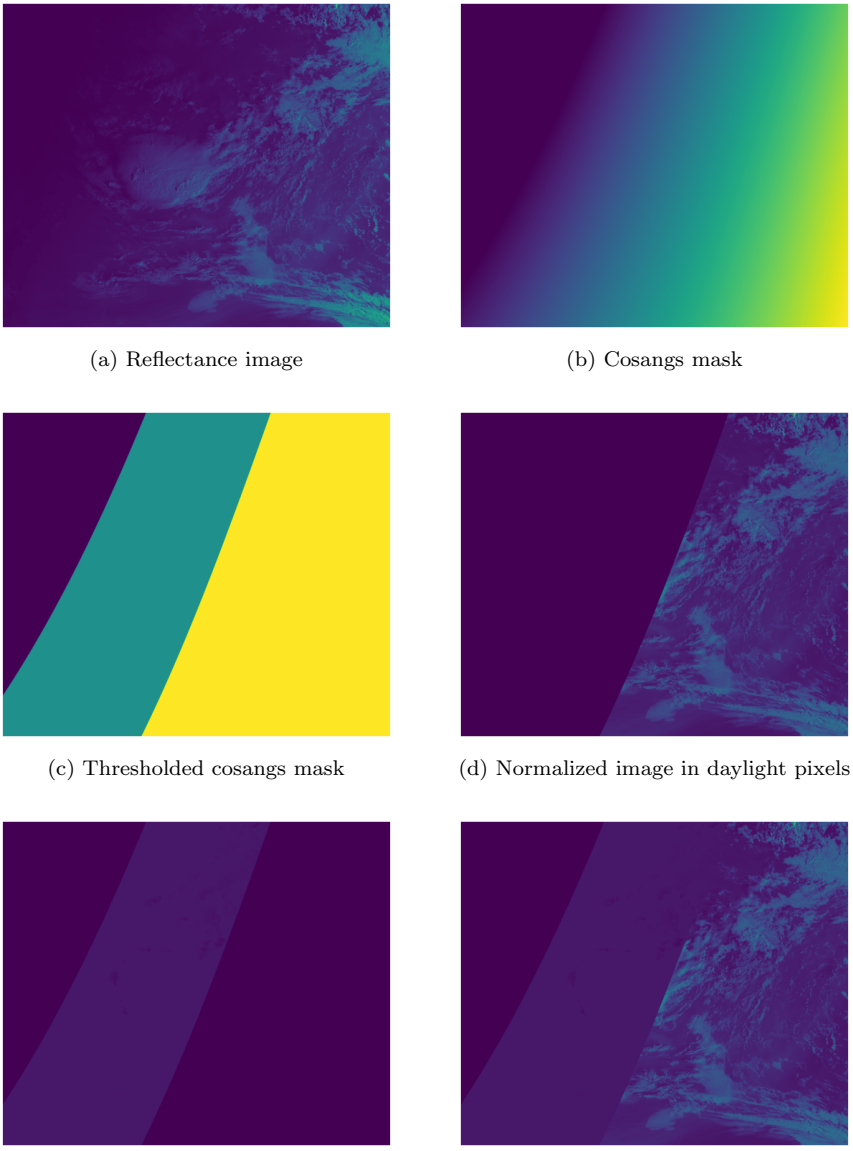
The sunrise and sunset times vary throughout the year. The hours before and after the start and end of the day, respectively, are not important for the task addressed in this work. Taking this into consideration, images with a maximum pixel value of zero are left out from the final dataset. This filter removed 23707 images, distributed in time as shown in Figure 2.7.

After the initial filter, another one was applied to find corrupted data that is not representative of the rest of the database and could jeopardize the training process. Through a first look at the database, an issue of black rows appearing in some cases was noticed. To detect more of these cases, a filter was designed to find multiple consecutive rows of pixels with a value of zero. After applying the filter function to the whole dataset, 47 images presenting this corruption mode were detected. Some examples are shown in Figure 2.8.

Considering the nature of the data that is used, two consecutive images should be very similar, as cloud and sunlight movement is slow in the 10-minute timescale. To find other corrupted images, another filter was run through the dataset in chronological order by taking two consecutive images and measuring the root mean squared difference (RMSD) between them. A large difference is a potential sign of a corrupted image, thus pairs that present extreme differences are manually analyzed. For instance, pairs of images in the tail of the histogram of differences between consecutive images shown in Figure 2.9 potentially contain a corrupted image. After examination of the images on the tail, we can identify corrupted images, like the one shown in Figure 2.10. After filtering out all non-desired data, the dataset consists of 28557 images.

This dataset was initially used for some experiments. Later in the project, it was noticed that the models had a different performance according to the time of the day, particularly in the sunrise and sunset images. Given that predicting these sequences is not relevant to the solar forecasting problem, it was decided to filter out images that contained pixels with $\cos\theta_z \leq 0.15$. Table 2.1 shows the number of images after each filter.

[1] https://sos.noaa.gov/catalog/datasets/daynight-terminator-daily



(a) Reflectance image

(b) Cosangs mask

(c) Thresholded cosangs mask

(d) Normalized image in daylight pixels

(e) Normalized image where cosangs pixels are between 0 and 0.15

(f) Final normalized image

Figure 2.5: (a) Reflectance image. (b) Cosangs mask used for normalization. (c) Thresholded cosangs mask. (d) Normalized image in daylight pixels. (e) Normalized image where cosangs pixels are in between [0, 0.15]. (f) Image resulting from the normalization step.

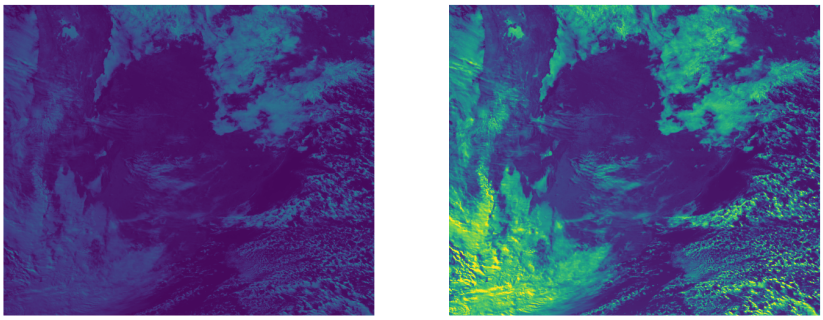


Figure 2.6: Example of an image before normalization (left) and after (right), from June 29, 2020 at 13:30 (UTC-0).

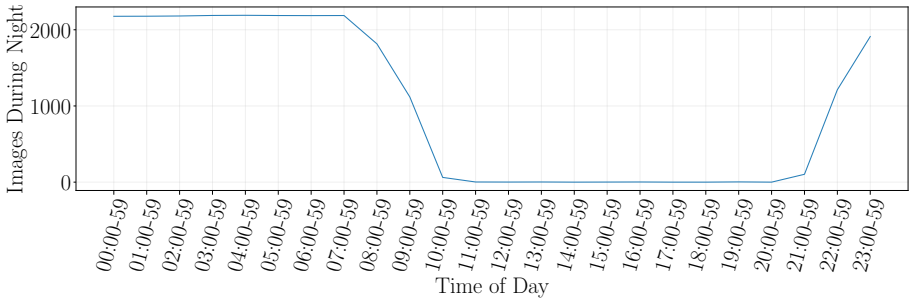


Figure 2.7: Number of images with maximum value equal to zero in function of the hour the image was taken. Timezone UTC-0.

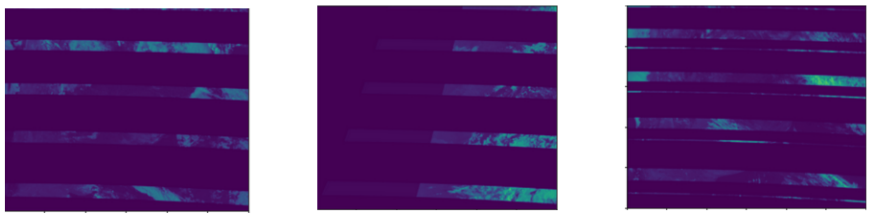


Figure 2.8: Examples of damaged images with black rows.

	# images after filtering		
Available Images	**Night**	**Corrupted**	**Not fully-daylight**
52314	28607	28557	16949

Table 2.1: Number of available images and number of images after each filter.

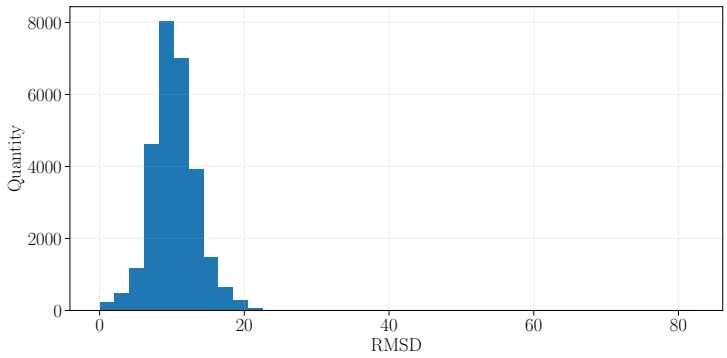


Figure 2.9: Histogram showing the Root Mean Squared Difference between two images which are consecutive in time. There is an outlier around the value 80.

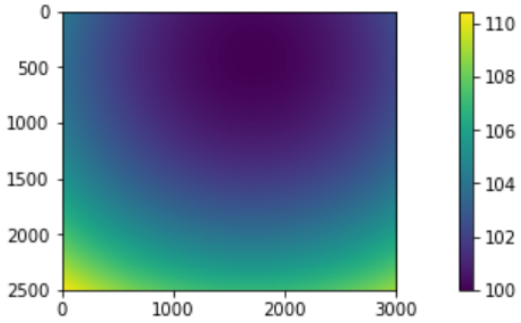


Figure 2.10: Anomaly found with the consecutive images difference filter.

2.3 Training Datasets

2.3.1 Image crops

As image sizes get bigger, the computing power needed increases as well as the time consumed for each model training. To be able to execute more experiments without restrictions, spatial crops of the original 2501 × 3001 images were created, resulting in three sub-datasets. Figure 2.11 shows the bounding boxes of the crops. The smallest crop is centered over Montevideo to obtain images with both sea and land presence. The middle-sized crop contains Uruguay as it is the first country where the model might be applied. The largest crop also contains parts of Argentina and Brazil. Table 2.2 shows the size of the images in each sub-dataset and the geographic region covered.

When saving the images for each sub-dataset it was necessary to filter again for sub-images with all its pixel values equal to zero, as it could happen that the 2501 × 3001 image passed the initial filter because it had a portion of the image with daylight but the cut is over an area where it is night time. The same is done to filter the images that contain pixels with $\cos\theta_z \leq 0.15$. Table 2.3 shows the resulting number of images in each subset after each of the two mentioned filters.

On the other hand, working with smaller regions limits the time in the future at which a confident prediction can be reached. With small images, there is not enough context to predict where clouds will be for the largest lead times, as clouds that aren't

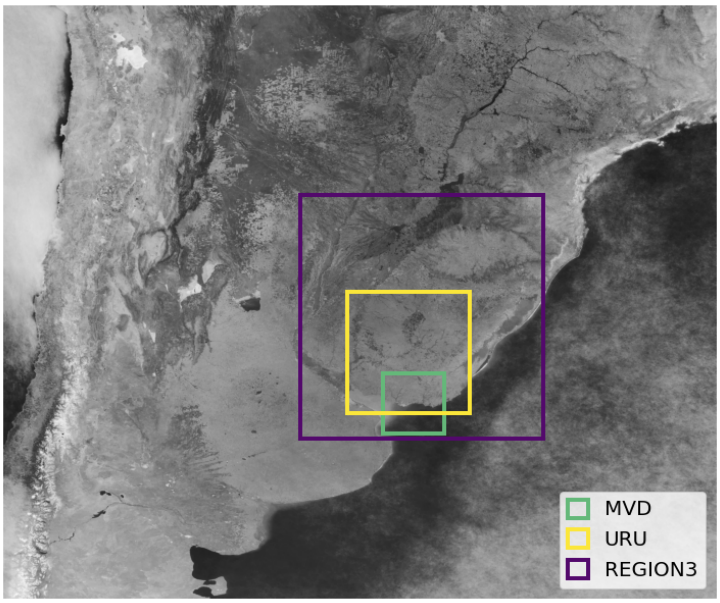


Figure 2.11: Visualization of the areas covered by the datasets generated. The purple, yellow and green squares mark the borders of the REGION3, URU and MVD datasets respectively.

Dataset	Size of images (px)	Latitude range	Longitude range
MVD	256×256	33°30'S - 36°02'S	57°00'W - 54°27'W
URU	512×512	30°02'S - 35°10'S	58°30'W - 53°23'W
REGION3	1024×1024	26°00'S - 36°14'S	60°30'W - 50°16'W

Table 2.2: The three sub-datasets with the size of the images and the geographic coordinates each one covers.

	# images after filtering	
Dataset	Night	Not fully-daylight
MVD	26554	22224
URU	27047	21820
REGION3	27920	20842

Table 2.3: Total number of images after each filtering for the three sub-datasets.

present in the input images may be present in the future. Considering this, initial experiments use small datasets to get preliminary results, and selected models are later run on the larger datasets.

2.3.2 Splitting

As is common practice in projects involving machine learning (ML), the dataset was divided into two subsets, one for training and the other for testing. For this purpose, 75% of the data went into the training set and 25% into the testing set. The split is done day-wise, meaning that 274 randomly selected days of the year went into the training split.

A hold-out approach was chosen for the model validation during training. This means that some of the days kept in the training dataset are left apart as a validation subset. These images aren't used as input to the model for tuning the parameters. Instead, after each time the parameters are updated, the current model is evaluated over this validation dataset to measure the performance. This performance is used for updating the learning rate and for choosing the best model. The validation dataset keeps 40 days away from training (15% of the train split). For more detail on the splitting see Appendix A.2.

The testing dataset is reserved for the final evaluation, to get results on data that the models did not see before. More precisely, it is to measure the intra-year generalization performance, meaning that the model could learn from certain days of 2020 and performs similarly when used to predict other days from the same year. It is expected to see a correlation between the training and test datasets as the images are from the same region over the same year. However, the correlation between one day and the next is small, for this reason is that the split was done day-wise, preventing a leak between the train and test datasets. Formally, if images are vectors $X_{t'}, \ldots, X_{t-h}, \ldots, X_t$, where t' is a time index of the previous day, we assume $\mathrm{P}(X_t|X_{t-h}, X_{t'}) = \mathrm{P}(X_t|X_{t-h})$.

2.3.3 Data Preparation

Preparing the data before using it as input to train a model is a must in any ML project. It has been proven by many researchers that for training it is better to work with small values rather than large ones [20], as it accelerates the convergence.

There are multiple ways to lower the maximum value of the images, in this case, the approach used was to divide all images by 100 which is the maximum value a pixel can take. This leaves the resulting images normalized between 0 and 1.

When working with other sources of information such as the spatial and temporal information as input, other normalization techniques were used. The spatial information consisted of three sources, the latitude, longitude, and elevation of the image. The latitude is represented by an image with the same shape as an image from the selected sub dataset, with its rows increasing in value from north to south covering the [0, 1] interval. The longitude representation is analogous to the latitude, but with the columns increasing east to west. The elevation is a crop of the corresponding region in the original map (Figure 2.4), with its values divided by the largest absolute number inside that crop, leaving the underwater areas with negative values and areas over the sea level positive numbers. When using temporal information, an image with the same shape as the dataset images is created, with all its pixels set to the same value. According to the information being represented, this value can be the minute, hour, day of the year, or month the image was captured, and it is normalized to [0, 1].

File type	Number of files	Size (TB)
FR	52314	1.5
MK	52314	0.73

Table 2.4: Size of raw files from the dataset.

2.4 Data Handling

Each image in the LES database is stored raw in a *.FR* and a *.MK* file. Metadata files are also needed to acquire the latitude and longitude to process the data. The dataset is then stored in three folders: the *.FR* files, the *.MK* files and the metadata. Table 2.4 shows the number of files and the size in Terabytes of the raw dataset. The metadata files sizes are negligible.

The data, stored in the LES server, had to be copied to ClusterUY [21] [2]. One approach considered was preprocessing the images in the LES server and only copying the processed images to the Cluster, but this meant that in the need of re-preprocessing the data (if the models required to), the images would need to be copied again.

The final strategy consisted in first filtering the black images, and copying the rest of the files. The test set, since it would only be necessary at the end of the project, was also not copied for the training stage. Since the ClusterUY administrators require not to saturate the input bandwidth, the transferred images were split into chunks and sent overnight via an automated script.

[2]The experiments presented in this paper were carried out using ClusterUY (site: https://cluster.uy)

Chapter 3

Cloudiness Forecasting

This chapter describes relevant related work. This project lies at the intersection between deep learning and solar forecasting. As a result, both fields were surveyed, with a special focus on the works in the intersection. Some of the existing works refer to other meteorological variables closely associated with cloudiness and solar radiation, such as precipitation. The most important architectures to be evaluated are described in this chapter, and the performance metrics that are used are introduced.

3.1 AI for climate

Deep learning-based methods are playing an increasingly important role in most scientific problems. In particular, deep learning strategies are emerging as a promising alternative to address problems related to weather prevision that were historically tackled with physical modeling and numerical methods only. A clear sign of this is given by the current policy at NOAA, which has increased the budget for these kinds of studies and has approved an Artificial Intelligence Strategy Plan, intending to promote the development of these solutions.[1] The same has happened in the UK Met Office, as exemplified by its partnership with Google's DeepMind.

Relevant deep learning methods were created for precipitation [13] [22], wind [23], and storm forecasting [24]. These problems present a direct and evident social and economic impact. Although relevant studies were made for the previous variables, which are central in the meteorological operational practice, cloud cover and solar irradiance appear to be left aside. With the growing usage of solar energy, both variables are getting more attention. Naturally, this problem presents similarities to the other ones. First, intra-day prediction is a complicated task because of the chaotic behavior of clouds and how they move and change their shape. Second, as large amounts of data are available and the prediction problem can be seen as self-supervised, human annotation is not needed. Thus, the problem is an ideal fit for deep-learning methods.

3.2 Solar forecasting

Solar irradiance forecasting is a relevant area of solar energy research. It tries to reduce the uncertainty of future resource availability estimation. The research area provides

[1] NOAA artificial intelligence strategic plan.

methods based on different principles and targets at different forecast horizons. The forecast horizons span from a few minutes ahead to yearly forecasts. For instance, intra-day forecasts are different from a few days ahead forecasts, which in turn are different from monthly or yearly forecasts. The principles behind the methods are related to the different forecast horizons. For instance, all-sky cameras are useful for minutes ahead forecasts. Satellite imaging is useful for intra-day forecasts, up to 4-5 hours ahead. Statistical methods are quite flexible but very dependent on the input data. Numerical weather prediction models are most useful for intra-day and days ahead forecasts. Of course, combinations are also used in the literature.

More relevant to this work is the observation that NWP methods provide a lower performance than satellite-based predictions to forecasts up to 4-5 hours ahead [25] [26] [27]. One of the most used techniques for satellite-based solar forecasting is the estimation of cloud motion vector fields (CMV), which are used to project and generate new images in the near future [28]. Several variations of these methods were developed and studied in [10], including the popular block-matching algorithm [28] and optical flow (OF) techniques. In particular, the OF Farneback formulation is available in OpenCV and will be used in this work as a strong baseline. This is the method currently in use by the LES and is a significant part of the operational solar forecasting system [11].

3.3 Performance Metrics

The decision of how to evaluate the performance of the models is a fundamental aspect of this work. The results must be valuable for the general and specialized reader, otherwise, the meaning might be lost and wrong conclusions could be made. The metrics will be calculated between the predicted images of one of the algorithms and the ground truth for different prediction horizons independently. The result will describe the quality of the prediction.

Solar forecasting has several guidelines on how to assess the performance of the predictions [29] [30]. These metrics are naturally inspired by the meteorological and solar radiation fields. The metrics used in this context represent a subset of the whole span of indicators that can be used, and do not necessarily overlap to a large extent with the metrics that are used in other fields, such as engineering, signal, or image processing. In particular, the field of computer vision has various metrics that assess different aspects of an image estimation or reconstruction, that can be used in the context of satellite-based forecasting. Moreover, the above-mentioned guidelines only cover solar radiation forecasting, not cloudiness forecasting, hence there exists some margin to use and propose different metrics. The lack of a unified framework has led authors to use different performance metrics, mostly guided by the knowledge field they are more familiar with. To account for this variety, many metrics are used in this work to evaluate the accuracy of the models. Metrics might have similar behavior but each one adds new relevant information on the performance of the predictions to cover all the important aspects. These metrics are listed and explained in the following.

3.3.1 MBD, MAE, MSE, RMSE

The Mean Bias Deviation (MBD), Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) are commonly used metrics. Taking two sample images X and $\hat{X}$, each with size $H \times W$, the MBD, MAE, MSE, and RMSE between these can be calculated as:

$$\text{MBD}(\hat{X}, X) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} (\hat{x}_{ij} - x_{ij}), \tag{3.1}$$

$$\text{MAE}(\hat{X}, X) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} |\hat{x}_{ij} - x_{ij}|, \tag{3.2}$$

$$\text{MSE}(\hat{X}, X) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} (\hat{x}_{ij} - x_{ij})^2, \tag{3.3}$$

$$\text{RMSE}(\hat{X}, X) = \sqrt{\frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} (\hat{x}_{ij} - x_{ij})^2}. \tag{3.4}$$

Each metric will give a different kind of information about the difference between a predicted image and the ground truth. The MBD measures, on average, if the predicted image has higher or lower values compared to the ground truth, i.e. the bias of the prediction. A positive MBD value means the prediction has, on average, higher values than the ground truth, and a negative one means lower values. This is because the MBD considers the sign of the difference between the images. The MAE, MSE, and RMSE measure the average magnitude of the deviation of the predicted image. As their name describes, MAE is an absolute error, and MSE and RMSE are squared errors, so they don't consider the direction or sign of the error. Since MAE is an absolute error it penalizes all errors with the same weight, while MSE and RMSE give a higher weight to large errors.

The relative metrics are used to allow a more representative comparison between techniques evaluated on different climates or geographic regions. These are calculated from the standard metrics: MAE, MSE, RMSE, and MBD. The notation used is MAE%, MSE%, RMSE%, and MBD%, although in other works it can be referenced as rRMSE for example. To calculate the relative metric at an image level, the standard metric is calculated as usual and then it is divided by the ground truths mean value.

$$\text{METRIC}\% = \frac{\text{METRIC}(\hat{X}, X)}{\bar{X}} \times 100. \tag{3.5}$$

3.3.2 Forecasting Skill

The Forecasting Skill (FS) aims to measure the accuracy of a forecast compared to a baseline. This is similar to the relative metrics, where the results are less reliant on the climatic and geographic characteristics of the region.

In the case of RMSE, using the Persistence as a baseline (well-known baseline in time series forecasting, introduced in Section 3.5.1) the FS is given by.

$$\text{FS} = 1 - \frac{\text{RMSE}(\hat{X}, X)}{\text{RMSE}(\text{Persistence})}. \tag{3.6}$$

From the equation, it is clear that the second term is minimized when the numerator goes to zero, hence a better forecast skill is closer to 1. A positive forecast skill means the prediction is better than the persistence in RMSE, while a negative value means it is worse.

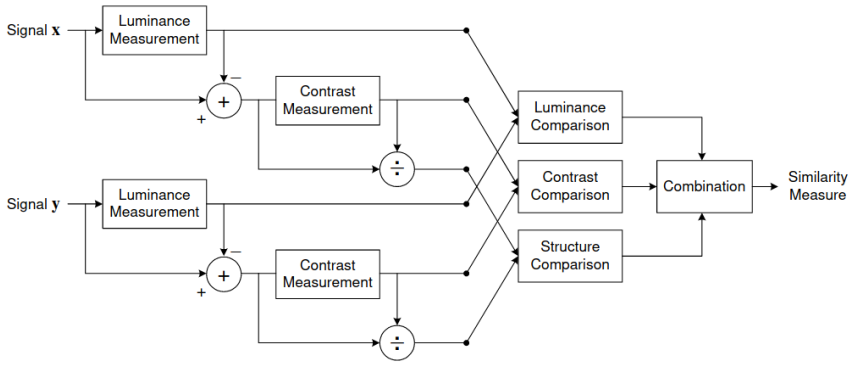


Figure 3.1: SSIM measurement system diagram. Figure extracted from [31].

3.3.3 SSIM, PSNR

The Structural Similarity Index, presented in [31], measures the structural degradation between a distorted image and a reference. This measurement is done by comparing three components: luminance, contrast, and structure. Under the assumption that the human visual system is highly adapted to extract structural information, SSIM aims to measure the quality of images based on the degradation of the structural information. Figure 3.1 shows the SSIM measuring system. The signals "x" and "y" are compared as explained, resulting in a value in the interval $[-1, 1]$, $+1$ meaning the signals are structurally similar, and -1 the opposite.

Peak Signal to Noise Ratio is another standard metric used in the literature. This expresses the quotient between the maximum possible value of the signal (which is related to the power) and the power of the noise that causes the distortion.

Given that many signals have a wide dynamic range, usually, the measurement is expressed in decibels. The expression is shown in Equation (3.7) and it reaches its maximum when the mean squared error is minimum.

$$\begin{aligned} \mathrm{PSNR}(\hat{X}, X) &= 10 \log_{10} \left(\frac{\mathrm{MAXVAL}(X)^2}{\mathrm{MSE}(\hat{X}, X)} \right), \\ &= 20 \log_{10}(\mathrm{MAXVAL}(X)) - 10 \log_{10}(\mathrm{MSE}(\hat{X}, X)). \end{aligned} \tag{3.7}$$

The acceptance of these metrics as measurements of how similar two images are with a human-like focus have been discussed in different works. An article about image restoration [32] compared the performance of identical architectures trained with different loss functions, and concluded that using MAE as training loss yielded better results than using SSIM or MSE. Another article critiquing SSIM and PSNR, went to the extent of calling these metrics simple and shallow, as they fail to account for many nuances of human perception [33].

3.4 Related work

The intra-day forecast is usually referred to as "nowcasting". Only a handful of deep-learning methods have been explored for this purpose. This section aims to describe

these related works. For a brief introduction to basic deep learning concepts see Appendix B.1.

Meteo France AI Lab researchers [12] evaluated several deep-learning architectures over a dataset consisting of binarized images (1 = cloud, 0 = no cloud) shot by the Meteosat Second Generation satellite of size 3712×3712 pixels and cropped to 256×256 pixels centered in France, with a cadence of 15 minutes. The authors carried out experiments to train over 20 models obtained by combining convolutional neural networks (CNNs), recurrent neural networks (RNNs), long short-term memory networks (LSTMs) in particular, and the U-net segmentation model [34]. According to their results and our informal meeting with the authors, the U-Net model was the best performing architecture. This promising model is described in detail in Section 3.5.2. All the models outperformed the persistence model (i.e. using the last input as prediction, explained more in detail in Section 3.5.1) over MSE, predicting every 15 minutes up to 1h30min into the future. Another interesting finding is that, unlike extrapolation techniques based on the cloud motion estimation, which tend to preserve the initial dimension of the clouds, the deep learning models appear to have the capacity of incrementing and decrementing the size of the clouds.

An article published in 2020 [35] used a dataset consisting of infrared satellite images taken every hour from the FengYun-2G satellite. This paper introduces a particular architecture based on the Convolutional LSTM [36], which they named Multi-GRU-RCN. The architecture is shown in Figure 3.2. It consists of an encoding network and a forecast network. They used batch normalization [37] to speed up learning and avoid overfitting. The appearance of clouds in the peripheral of the input images is a problem to take into account, especially due to the time lag between consecutive images. To solve this, the authors introduce the concept of spatial context. The Multi-GRU-RCN takes two images as inputs, one covering a small region (64×64 pixels) and the other covering a larger region (128×128 pixels) centered in the first (spatial context). The network predicts only over the small image, allowing it to "see" incoming clouds. The authors compared this architecture against other models commonly used in image prediction such as ConvLSTM, LSTM, and GRU, and against the CMV, which calculates the variational optical flow [38] (VOF) for every pixel. The dataset used to train these networks consists of normalized images of size 512×512 pixels. For each image, 16 patches of size 128×128 were taken as inputs for the network's training. The networks minimize MSE and compare the performance of the algorithms using well-known metrics in the deep learning literature such as PSNR and SSIM. The authors state that although the models achieve promising results, they have limitations due to the blurred outputs when comparing them to the VOF. This blur is a natural result of using MSE as the loss function since it averages over all the possible outcomes. They mention the use of generative adversarial networks (GANs) as a potential alternative to enhance prediction sharpness and image definition.

A curated cloud-classified dataset [39] was presented to promote further research in the area, named the "CloudCast" dataset. It has ten different could types annotated on a pixel level centered over Europe. It consists of 70.080 images of size 928×1530 pixels, with a spatial resolution of 3×3 km and a cadence of 15 minutes between consecutive images. This granularity improves over previous public databases, which previously had a spatial resolution of 9×9 km per pixel and one-to-multiple hours between images. On top of the dataset, the authors present experiments using the ConvLSTM model [40], a Multi-Stage Dynamic GAN, and TVL1 (optical flow), up to 4 hours into the future. For the ConvLSTM model, the authors use an architecture similar to the U-Net. Both models add a common module used in deep learning architectures that feeds the outputs of a layer to other layers that are not directly connected. This module is known as a skip connection since it transfers data by

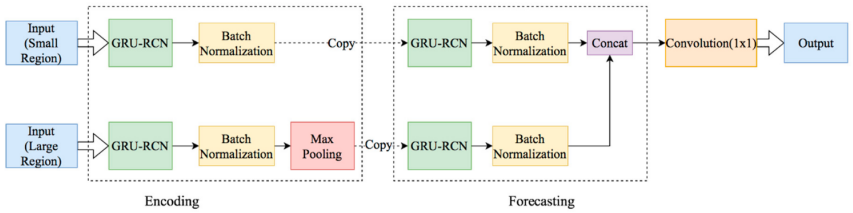


Figure 3.2: Multi-GRU-RCN architecture. Figure taken from "Prediction of Short-Time Cloud Motion Using a Deep-Learning Model" [35].

skipping layers. It is used to solve a common problem known as vanishing gradient and allegedly simplifies the model task. The loss function is MAE, presumably making predicted images sharper. The evaluation involves common video prediction metrics such as PSNR and SSIM.

During the development of this work a relevant paper was published under the title "IrradianceNet: Spatiotemporal deep learning model for satellite-derived solar irradiance short-term forecasting" [41]. With the same objective as this project, the authors develop a deep learning model to predict solar irradiance up to 4 hours ahead. The IrradianceNet model is based on ConvLSTM cells and consists of an encoder-decoder network. As this model was implemented, it will be described in detail in Section 3.5.4. The experiments were ran on the European SARAH-2.1 dataset [42]. The dataset contains images of the surface solar irradiance that have a cadence of 30 minutes, in contrast to our dataset that contains images in the visible channel and a cadence of 10 minutes. Their images have a size of 512×512 pixels (as the URU dataset) and each pixel has a spatial resolution of $0.05° \times 0.05°$ degrees (a lower resolution compared to the $0.01° \times 0.01°$ degrees resolution provided by the GOES-16 satellite). The values of the pixels are in the range $[0, 120]$. The dataset has information stored from 1983 to 2017, although the experiments only use data from 2010 onward. For training the models, the authors used the data from 2010 to 2014. The validation set consists of the images from 2015. For the final tests, the images from 2016 to 2017 were reserved. This point marks a difference from our work, where the training, validation, and test sets all come from the same year (2020).

To measure the IrradianceNet performance, its results were compared against persistence, ERA5 reanalysis, and TV-L^1 [43] Optical Flow. The ERA5 reanalysis is a Numerical Weather Prediction (NWP) model provided by the European Centre for Medium-Range Weather Forecasts [2]. As explained by the authors, it combines large amounts of historical observations into global estimates using advanced modeling and data assimilation systems. The TV-L^1 algorithm falls into the same category as the Farneback one (presented on Section 3.5.1), as both are Cloud Motion Vector techniques. Their performance has already been compared against each other on a dataset similar to the LES dataset (it has a cadence of 30 minutes), and the research [10] concludes with a better performance using the TV-L^1 algorithm.

A group of researchers from DeepMind develop a deep generative model (DGM) for precipitation nowcasting [22]. This model intends to address the problems that some state-of-the-art models have, such as producing blurry predictions for longer lead times. It uses radar information for precipitation forecasting. For this model, they use a generative adversarial network, shown in Figure 3.3. They use 4 frames separated by 5 minutes intervals (Context in the figure), corresponding to the previous 20 minutes

[2] https://www.ecmwf.int/

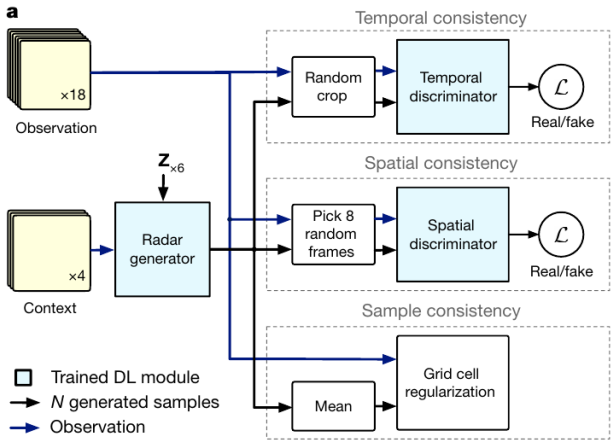


Figure 3.3: DGMR architecture. Figure taken from "Skilful precipitation nowcasting using deep generative models of radar" [22].

to predict 18 frames (Observation in the figure), meaning 90 minutes ahead. The model is penalized using a loss function consisting of 3 terms as shown in the figure. The first term is defined by a spatial discriminator that ensures spatial consistency. The second, a temporal discriminator focus on the temporal consistency, and the third term is added for regularization which improves performance, the authors state. They compare the performance against PySTEPS (a system that follows the reasoning of Numerical Weather Predictions systems but estimates motion fields using optical flow). The other baselines include the deep learning model U-Net and a radar-only implementation of the MetNet architecture, a complex model used for precipitation forecasting. Their experiments show that the U-Net and the MetNet model fail to capture any small-scale structure and produce excessive blurring, while the PySTEPS overestimates the rainfall. Their model preserves small-scale structures, though less accurate at the 90 minutes lead time. Additionally, they conducted an experimental study to assess expert judgments of value. In this study, the identity of the methods presented to each group was anonymized, in random order. The results show that 89% of the experts preferred the generative model over the others.

Google Research presented a neural network for precipitation forecasting up to 8 hours into the future over the US [13]. The new architecture, shown in Figure 3.4, takes into account multiple sources of information, including satellite images, radar images, latitude, longitude, and time of day, among others. This information is concatenated in the channel dimension. The architecture comprises a spatial downsampler, a temporal encoder, and axial self-attention.

An ablation study of the impact of the number of input images over the performance was conducted in some of the works [12] [13] [35]. Using more than 2 images does not offer relevant advantages for the U-Net model [12]. This finding is in line with other experiments [13], where the number of input images was reduced from 6 to 2, reducing the temporal context from 90 to 30 minutes, and there was no significant negative impact. These findings were different from those in [35], where MSE decreases (improves) when using 6 instead of 2 images, without significant computational overhead.

As depicted in this section all datasets vary in size, temporal resolution, and the meteorological quantity that is being modeled. Different authors found different forms

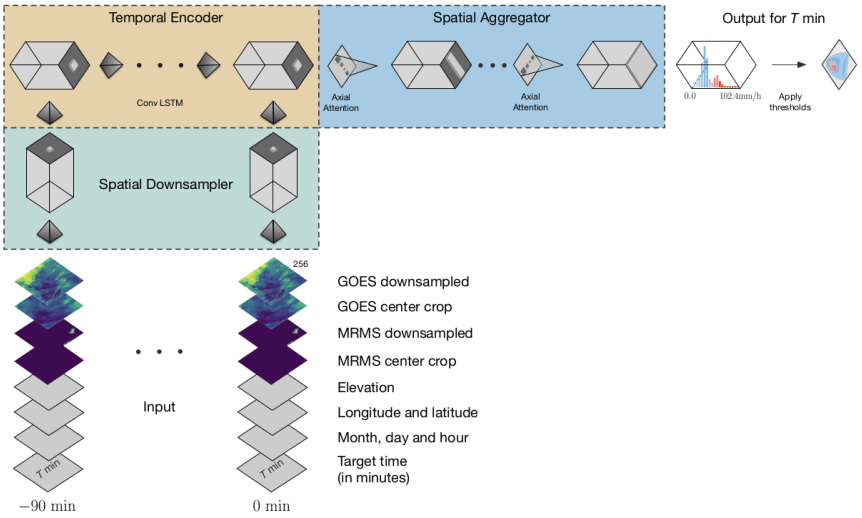


Figure 3.4: MetNet architecture. Figure taken from "MetNet: A Neural Weather Model for Precipitation Forecasting" [13].

to simplify the problem, from binarization of the images or making it a multi-class classification problem by setting thresholds, to background subtraction in the preprocessing step. Every dataset from the literature presented here differs from ours. Apart from this, every author had to reduce the size of the input images due to the computational restraints that high-resolution images represent when training deep learning models. Our datasets, described in Section 2.3, reach up to 4 times larger than the datasets presented in this section, and it is the first time these ideas are applied using GOES satellite images.

3.5 Architectures

The most relevant architectures were extracted from the literature review. This section explains in more detail the selected strategies.

Two benchmark algorithms were used to establish a baseline to compare with the deep learning algorithms. These are Persistence and Cloud Motion Vectors (CMV). A version of the CMV algorithm with a spatial Gaussian blur applied to its prediction was also used.

3.5.1 Baseline models

The persistence method is simple, yet it has been proven very useful as a baseline reference in meteorological forecasts. Having a reasonable baseline is important in all machine learning problems, as it marks the minimum performance a model should reach to be considered functional. In general, the persistence method implies assuming constant the last entrance of a temporally-ordered data series, intending to represent a naive way of forecasting. In the context of this work, the persistence is to maintain constant the last image for all the following forecast horizons. For fast-changing time series, it might not make sense to keep a stationary prediction, but for cloudiness

forecasting, it is competitive with other forms of establishing a prediction. This is especially true for the shorter lead times, as consecutive images do not present large differences between each other. Of course, this method becomes worse with increasing forecast horizons and the performance downgrades, but it still provides a reference for comparison. Taking into account its simplicity and reasonable usability, persistence is strongly present as a baseline in most related work.

As mentioned in the objectives in Section 1.6, the models based on Deep Learning should be an improvement to the system already in use by the LES. The core of the system is a Cloud Motion Vector (CMV) satellite technique [11] [9]. The CMV algorithm estimates the direction of the motion of the clouds at a given pixel on the (x, y) position, generating two scalar fields $u(x, y)$ and $v(x, y)$ corresponding to the components of the motion vector. These scalar fields are used pixel-wise on the image on t_0 to obtain the estimation of the image on t_1:

$$\hat{X}_{t_1}(x, y) = X_{t_0}(x - u,\ y - v). \tag{3.8}$$

The Optical Flow algorithm from Farneback [44], implemented in the open-source library OpenCV, was used to calculate the CMV. The algorithm takes as input two images, named $X_{t_{-1}}$ and X_{t_0}, and the time elapsed between them (ten minutes in this work). The output of the algorithm is the Cloud Motion Vectors; Figure 3.5 presents an example of a CMV map. To generate a prediction, the CMV map is applied to the last image of the input (X_{t_0}) to extrapolate and obtain the first prediction ($\hat{X}_{t_1}$), ten minutes into the future from the last input.

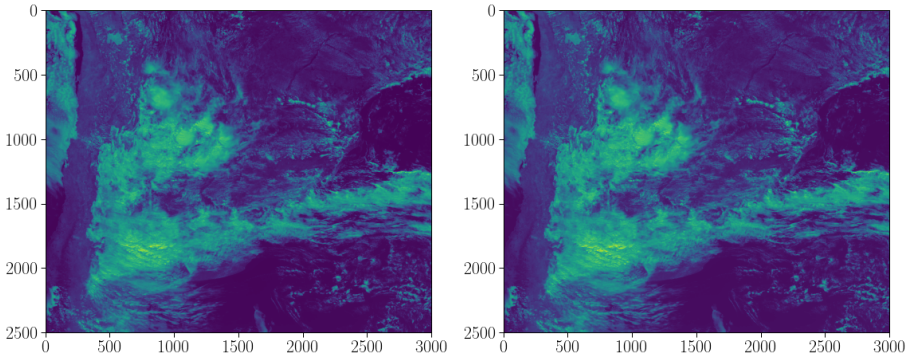
3.5.2 U-Net

In 2015, the paper "U-net: Convolutional networks for biomedical image segmentation" [34] was published. In it, the authors use an autoencoder base for their architecture and add skip connections between the Encoder and Decoder. Until today, U-Net continues to be a reference in the state of the art in segmentation and other computer vision problems, and this pushed forward the creation of new architectures based on the original [45] [46] [47].

The decision to include the U-Net architecture in this work comes from the Meteo France AI lab article [12]. In a conversation with Léa Berthomier, one of the authors of the paper, she commented to us about the alternatives they tried and how they were not able to improve the original U-Net results, both in the real data and in the artificially made data.

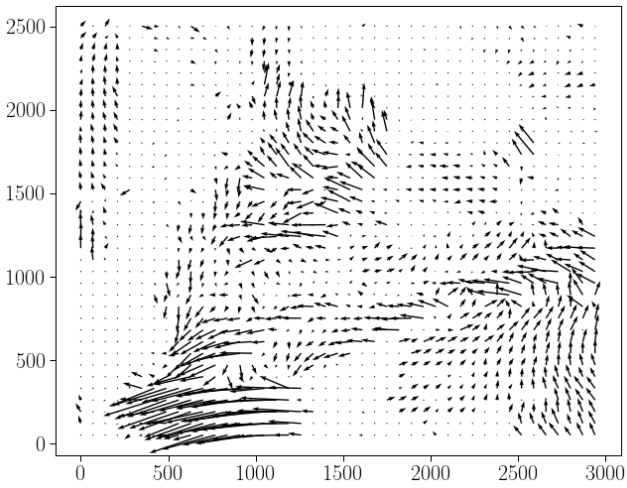
This network (Figure 3.6) can input and output 3-dimensional data of any size, so without any changes to the architecture, it can be used on the MVD, URU, or REGION3 datasets. Through convolutional layers in the Encoder, the network learns features and patterns in the image sequence, for instance, it may easily learn distinguishable issues, like speed and moving directions of the clouds and others. In the Decoder, the prediction is built using the output of the convolutional layers at different levels of the Encoder and the features extracted and upsampled at each level of the Decoder using bilinear interpolation (represented by the green arrows in Figure 3.6). The skip connections benefit the Decoder in reconstructing the output with the additional information, and make the learning more stable by reducing the risk of vanishing gradients.

A difference between the original implementation of the U-Net in Figure 3.6 and the initial version used here is the addition of padding in the convolutional layers, so there is no reduction in the size of the output image with respect to the input. It is standard to start the network with two convolutional layers with 64 filters and a kernel of size 3×3 each, so if the input to the first convolutional layer has a height



(a) Image from March 31, 2020 at 15:40 (UTC-0)

(b) Image from March 31, 2020 at 15:50 (UTC-0)



(c) CMV from (a) and (b). The CMV is downsampled to facilitate its visualization.

Figure 3.5: Example of the CMV map from two consecutive images.

H, width W, and C channels, the output of the second convolutional layer (in this case) has shape H $\times$ W $\times$ 64. To take the output of a filter layer in the Encoder and input it to the next level, it is necessary to reduce its height and width by half. The chosen method to do this reduction is by applying a max-pooling layer (represented by the red arrows in Figure 3.6), which consists of passing a 2 $\times$ 2 window through each channel that only keeps the maximum value in it. After the max-pooling layer, the number of filters in the next level is duplicated.

The quantity of trainable parameters in the U-Net varies with the number of filters in the first layer, the length of the input sequence, and the channels in each image. The number of parameters does not depend on the height and width of the images. The equation to calculate the number of parameters is:

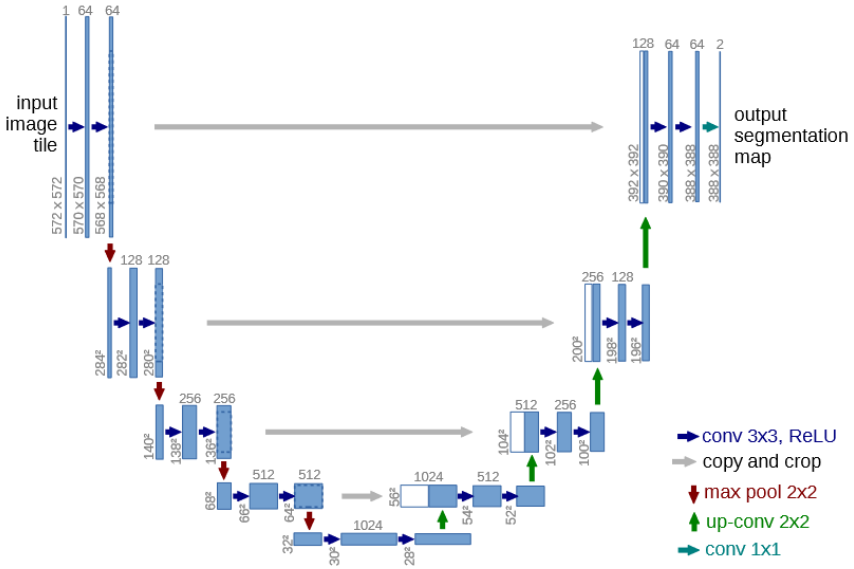


Figure 3.6: U-Net Original Architecture. Figure taken from U-net: Convolutional networks for biomedical image segmentation [34].

$$\text{Encoder: } 9\,S\,C\,f + 2295\,f^2, \tag{3.9}$$

$$\text{Decoder: } 1917\,f^2 + f + 1, \tag{3.10}$$

where C is the number of channels in an image, S is the length of the input sequence, and f is the number of initial filters. Making $C = 1$, $S = 3$ and $f = 64$ the network has 17,254,145 trainable parameters. Choosing the number of initial filters is relevant because it gives the size that the model will occupy in memory, and if it is too big it can cause an impossibility to work with large images such as the ones in the REGION3 dataset, since there is not enough space left in memory for the images. Figure 3.7 shows the number of parameters and the size of the network for different values of f with $C = 1$ and $S = 3$.

It is relevant to note from Figure 3.7 that the parameters alone do not take up much space in memory, but the issue emerges when training the model. To adjust the values of the parameters (also called weights), a batch of images is passed through the network to generate the predictions (a forward pass), but the intermediate outputs of the network are also saved momentarily, as these values are needed further, to compare the predictions against the ground truth. The process of updating the weights is done using the backpropagation algorithm, which takes the result of the loss function and the intermediate outputs to update the weights to minimize the error made on that particular input. More parameters and larger images translate into more intermediate results to save during training, so when performing backpropagation the use of memory increases considerably with respect to the size of each image and the number of initial filters, as shown in Figure 3.8.

Apart from the memory issue, needing a really long training time is not desirable. The time it takes to run an epoch is proportional to the number of operations per-

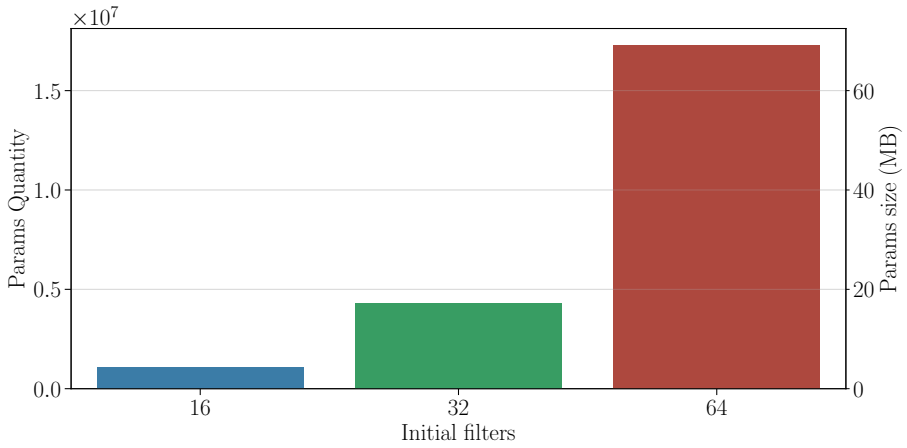


Figure 3.7: On the left y-axis it is marked the quantity of trainable parameters for different number of initial filters. The right y-axis shows the size it takes to allocate those parameters.

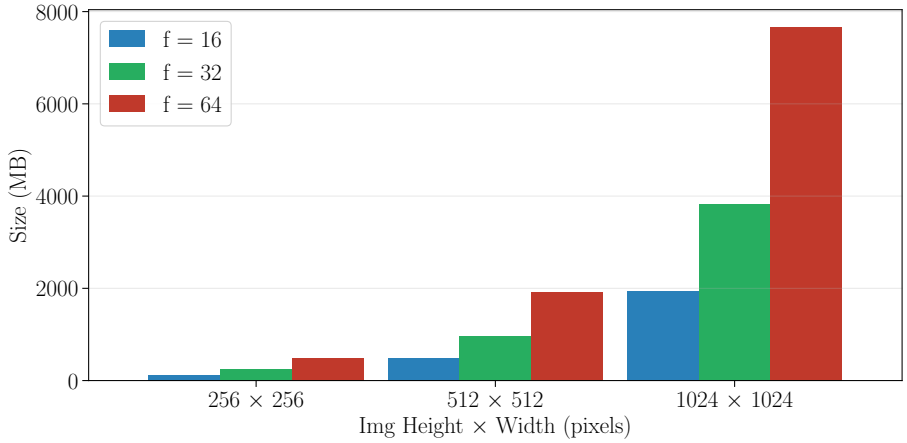


Figure 3.8: Memory used during training for different images sizes and number of initial filters.

formed during training. As is shown in Figure 3.9, this value increases with the size of each image and the number of initial filters.

3.5.3 U-Net Variations

Given the positive preliminary results obtained with the U-Net, further investigation about this architecture was conducted to find possible improvements. As mentioned before, the U-Net architecture inspired multiple versions of itself. In this case, three architectures [3] presented in different papers about segmentation in medical images are tested. These three architectures are the Attention U-Net [46], Nested U-Net [45], and Recurrent Residual U-Net (R2U-Net) [47]. Each one is based on the U-Net and

[3]The code for the networks mentioned in the next subsections was taken from this repository

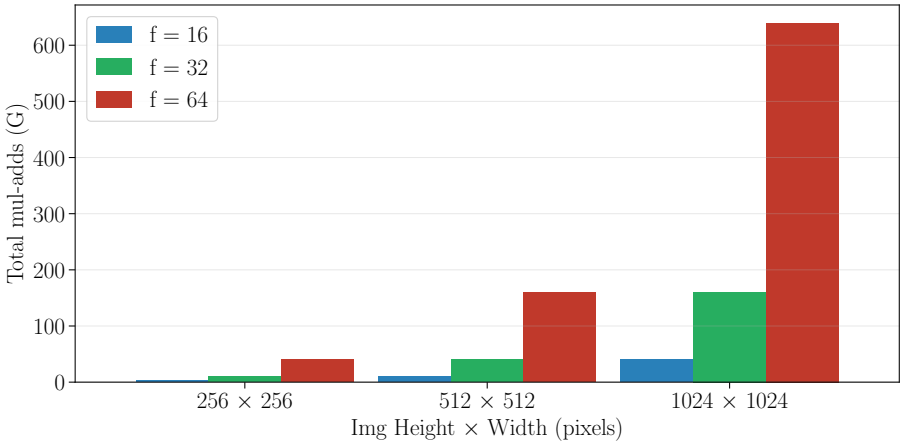


Figure 3.9: Number of operations done in one forward pass according to the initial filters (f) and image size.

adds a unique idea to it, which are explained further in Appendix B.2. In the Attention U-Net the main modification was adding soft-attention layers, which allow to detect the most relevant regions in the input images and assign more importance to those while processing. The Nested U-Net modifies the skip connections by adding more convolutional layers in between the encoder and decoder, referred to as dense skip connections, which are expected to reduce the semantic gap between the feature maps and improve the gradient flow. Finally, the R2U-Net, instead of the original convolutional layers this network has Recurrent Residual Convolutional Units (RRCU), where the input is passed through recurrent convolutional layers, and the output of the second convolutional layer is added to the initial input.

The implementations of these models in this work have around 36 million trainable parameters, approximately double the largest U-Net considered in Section 3.5.2. Because of the different techniques added to the variations, the number of operations needed to generate a prediction differs significantly, with Attention U-Net needing the fewest, while the Nested U-Net and R2U-Net need approximately double and triple than the Attention U-Net, respectively.

3.5.4 IrradianceNet

The IrradianceNet model [41], introduced in Section 3.4, is an encoder-decoder network based on ConvLSTM cells. The architecture is shown in Figure 3.10.

Specifically, this model is composed of a spatiotemporal autoencoder architecture with three ConvLSTM layers in both the encoder and the decoder, capable of taking multi-channel inputs of dimension $[C,\ t,\ H,\ W]$. The output of the encoder after the three ConvLSTM layers is a tensor of shape $[128,\ t,\ \frac{H}{4},\ \frac{W}{4}]$. The decoder part of the network upsamples the tensor to obtain an output of shape $[1,\ t,\ H,\ W]$. The final step of the network is to apply a 3D CNN to obtain the output sequence.

Two different approaches were tested to adapt the network to the European SARAH-2.1 dataset [42], as the images of size 512×512 pixels were too large to be trained on without a modification. The first one involved dividing the 512×512 pixels images into 16 non-overlapping patches of 128×128 pixels. This method allows to select them in a random manner and input to the network. The second approach involved

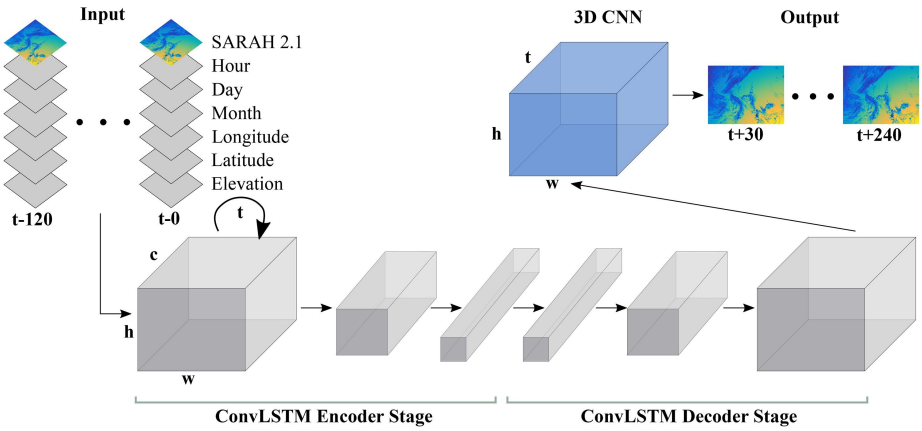


Figure 3.10: IrradianceNet Architecture. Figure taken from IrradianceNet: Spatiotemporal deep learning model for satellite-derived solar irradiance short-term forecasting [41].

downsampling the images to size 128×128 pixels. For comparing performance with the other models trained in this project, only the IrradianceNet based on patches was used, because it was the only option that could be implemented with the REGION3 dataset without altering the images. It may be of interest to notice that in the article, when the patch-based solution is introduced, the reference mentioned for this approach is the MetNet paper [13]. Also the MetNet and the IrradianceNet architectures are fairly similar.

The models presented use four previous time steps as input to generate a prediction. Apart from the satellite images, the authors experiment with other sources of information as input. The other data used from each image is the hour, day, and month it was taken, and the longitude, latitude, and elevation of the patch selected. The improvement of the model trained with this extra data over the one with only images is not too important but still relevant, according to the authors. In a communication with Andreas Holm Nielsen, one of the authors, we asked whether they had done any study on which extra source of information gave the best improvement but he stated that they had not. For the implementation of IrradianceNet in this work, two models are used, one only with images and the other with images and geographic information, i.e. the longitude, latitude, and elevation.

The recurrent aspect of LSTM architectures allows to input a sequence of ordered data (images in this case) and output another sequence. So when the model is given a fixed input, it could output predictions for all the intermediate images till reaching a certain time horizon (Figure 3.11). The authors indicates that they trained a model for each of the four lead times independently. This means they didn't train a single model capable of predicting the whole sequence of images at the 30, 60, 90 ..., and 240 minutes lead time together. In their method, the intermediate images aren't compared against the ground truth during training, only the last output, so the intermediate images are not predictions. The change concerning Figure 3.11 is that the first outputs in the decoder aren't used to calculate the loss, so during backpropagation, the weights are updated to fix the last output only. This implementation is represented in Figure 3.12.

3.5.5 Generative Adversarial Networks

Generative models, as opposed to discriminative models, estimate the probability distribution $P(X)$ of the input data X. Discriminative models, on the other hand, try to

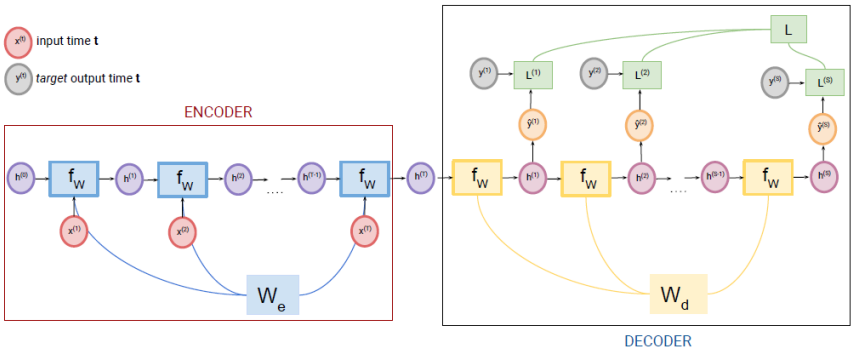


Figure 3.11: Recurrent Neural Network for sequence prediction. Figure taken from UdelaR Engineering Faculty course Deep Learning for computer vision, class 14.

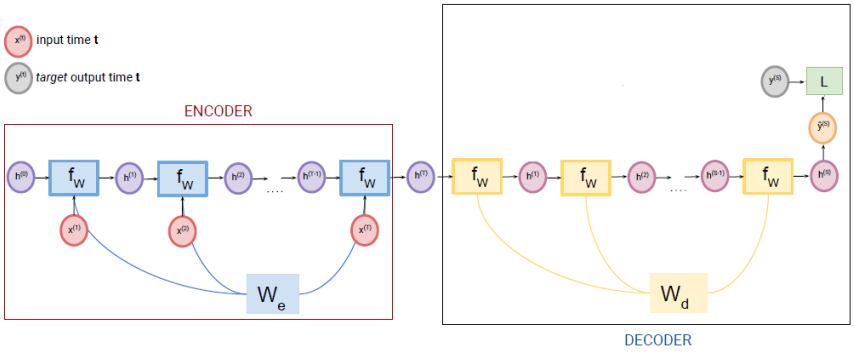


Figure 3.12: Recurrent Neural Network for sequence prediction, using only the last output to calculate the Loss and update the weights. Figure edited, original taken from UdelaR Engineering Faculty course Deep Learning for computer vision, class 14.

estimate $P(Y|X)$, where Y is a target. Generative models allow to sample from the estimated $\hat{P}(X)$.

Generative adversarial networks (GAN) [48], are a special kind of neural network that estimate the data distribution through an adversarial process in which two models are trained simultaneously. These two models are the generator G, which captures the distribution of the data, and the discriminator D which estimates the likelihood that a sample data point comes from the data versus being generated by the generator. The idea for this networks is shown in Figure 3.13. As seen in the figure, the generator takes a random noise vector from a latent space and outputs an image. Then, either this image (the fake one) or a real image from the training set is passed through the discriminator, where it has to decide whether the image is real or not. The theory behind how GANs learn can be found in Appendix B.3.

Despite the extraordinary results that these architectures are capable of achieving, initially, they showed common "failure modes". The most common are:

- Vanishing gradient: ML models that are based on minimizing the gradient of a loss function and backpropagation, can suffer from the vanishing gradient problem. When the error is backpropagated through the network, the weights

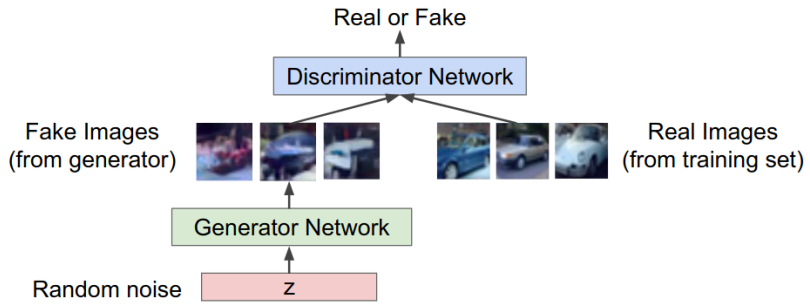


Figure 3.13: GANs basic architecture idea. Figure taken from lecture 11 from the cs231n course from Stanford.

change proportionally to the gradient and if this is too small, the weights do not update and the network stops learning. It has been found that the generator training can end up suffering from this problem if the discriminator is too good [49].

- Mode collapse: Mode collapse happens when the generator maps several inputs to only one type of output. This can happen if a given output is good at tricking the discriminator.
- Convergence problems.

To solve these problems, as well as to make it possible to use these models for a wide variety of tasks, different types of GANs have been developed [50] [51] [52]. In particular, the wGAN [53] makes use of the Wasserstein distance as the training loss function to measure the distance between the probability distribution of the data versus the one from the generator. The resulting wGAN has more stability than the vanilla GAN.

Apart from the mentioned ones, other variations have achieved remarkable results in different domains. For predicting future frames in video sequences, two studies arose in the literature research step, that used GANs as architectures: [54], [55].

Wasserstein GAN

The wGAN [53] uses the Wasserstein distance as the loss function instead of Jensen-Shannon divergence (the one used in the vanilla GAN, and explained in Appendix B.3). This distance, as stated before, offers a more stable training process which helps to prevent mode collapse. This metric is also referred to as the earth mover's distance. Informally, this can be interpreted as the minimum amount of energy required to move a certain amount of dirt (earth) with the shape of one probability distribution into the shape of a target probability distribution [4].

An additional upside to this version of a generative adversarial network is that the loss function provides a termination criterion. If the generator manages to minimize the difference in the expected values of the generated images and the ground truth, then the probability distributions from these sources are equivalent and the discriminator can no longer distinguish between one another. The adaptation of the wGAN for this work is explained in Section 4.4.

[4] Earth mover's distance.

For a detailed mathematical explanation of the differences between the wGAN and the vanilla GAN, see Appendix B.3.

Chapter 4

Models' optimization and selection

This Chapter presents the training and evaluation of the architectures and algorithms presented in Section 3.5 on the training and validation datasets, respectively. To look for the best performance, decisions are made over the architectures and how they are trained. The optimization includes the Deep Learning algorithms and also the Cloud Motion Vectors state-of-the-art baseline method. The models with the best results are selected for their final performance evaluation over the testing dataset (unseen for any kind of training), which will be covered in the next chapter.

All Deep Learning models and training pipelines use the Pytorch machine learning framework [56]. Most experiments were run on the national supercomputing platform ClusterUy [21].

4.1 CMV: Cloud Motion Vectors

There are two ways of applying the scalar fields $u(x,\ y)$ and $v(x,\ y)$ calculated by the CMV algorithm to reach horizons further apart from the first ten minutes mark. The first approach would be calculating the CMV and multiplying its values by the number of lead times it should be applied to reach the desired horizon (k). This logic is exemplified by Equation (4.1), where X_{t_o} is the current last available image, and $\hat{X}_{t_k}$ is the predicted image at a lead time k.

$$\hat{X}_{t_k}(x, y) = X_{t_0}(x - k \times u,\ y - k \times v). \tag{4.1}$$

The other way to use the CMV is by generating the intermediate steps iteratively and incrementally. A prediction ten minutes into the future is made using the current time image and the CMV, and after the same CMV is applied to the previously predicted image to generate the next one, and so on. This procedure is shown in Equation (4.2) and does not require an incrementation of the CMV by any factor.

$$\hat{X}_{t_k}(x,\ y) = \begin{cases} X_{t_0}(x - u,\ y - v),\ k = 1, \\ \hat{X}_{t_{k-1}}(x - u,\ y - v),\ k > 1. \end{cases} \tag{4.2}$$

After evaluating on the training dataset, it was concluded that the second approach had a better performance. This was previously observed in the Master Thesis of G. Giacosa "Solar energy forecasting from satellite images" [9], using the GOES-East images of the previous satellite generation, with a 30 minutes image time rate.

In this algorithm, the movement of the clouds causes undefined edges in the predicted images since the values of the pixels outside the image are not known. For

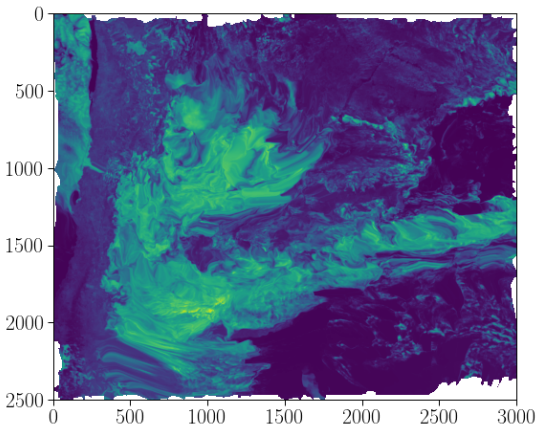


Figure 4.1: Prediction four hours ahead from the images captured on March 31, 2020 at 15:40 and 15:50 (UTC-0), using CMV. The white pixels in the image borders are NaN values.

30 min	**60 min**	**90 min**
6.93%	13.69%	20.01%

Table 4.1: Mean percentage of NaN values for the CMV predictions on the MVD validation dataset, for 30, 60, and 90 minutes prediction horizons.

example, if there is a movement of clouds to the right side of the image, then on the left side of the predicted images there will be undefined values, as this information is not available. These values are filled with NANs (Not a Number) and have to be handled in the evaluation. An example of this effect is shown in Figure 4.1, where a prediction four hours ahead is done using the CMV methodology, and the pixels that should enter the image are considered NANs (white pixels in the image borders). Table 4.1 and Table 4.2 show the percentage of NaN pixels produced by the CMV as a function of the prediction horizon for the MVD and REGION3 dataset, respectively. These values are of interest given that the evaluation of the CMV over these datasets can only be done on not-NaN pixels and this affects the performance. As can be seen, for the one hour lead time, the percentage of NaN pixels for the REGION3 dataset is 3.12%, while for the MVD dataset is 13.69%, more than 4 times bigger.

Another issue that is relevant to analyze is the dependence of the CMV performance with the spatial smoothing of the predicted images. Applying spatial smoothing to the predictions of the CMV algorithm is proven to generate better results [10]. This can be attributed to the predictions of clouds being inaccurate at long predic-

1hr	**2hr**	**3hr**	**4hr**	**5hr**
3.12%	6.10%	8.91%	11.60%	14.11%

Table 4.2: Mean percentage of NaN values for the CMV predictions on the REGION3 validation dataset, for 1, 2, 3, 4, and 5 hours prediction horizons.

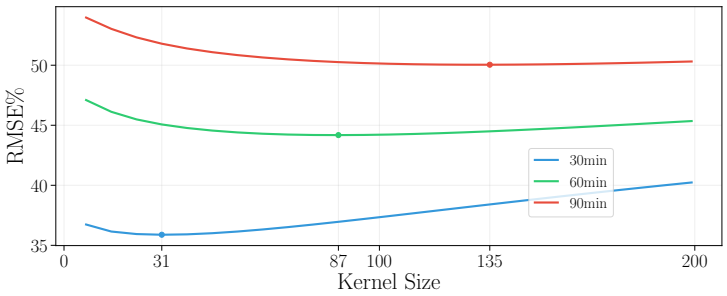


Figure 4.2: Kernel size optimization of blur applied to CMV for MVD. Calculated as the average in the MVD validation dataset, for 30, 60, and 90 minutes prediction horizons.

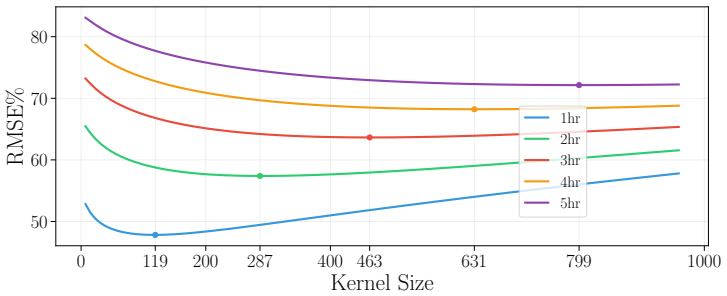


Figure 4.3: Kernel size optimization of blur applied to CMV for REGION3. Calculated as the average in the R3 validation dataset, for 1, 2, 3, 4, and 5 hours prediction horizons.

tion horizons, so a smoothing effect can average these inaccuracies. A Gaussian blur for the spatial smoothing was inspected in this work, as a way to optimize the CMV performance by minimizing its error in RMSE%. It shall be noticed that the spatial smoothing applied in [10] is the simple average in a square cell of variable size, depending on the time horizon. The smoothing can be applied with a fixed size to all forecast horizons or optimized for each one individually. The second option is known to be better so an optimization for all relevant horizons was done. For each forecast horizon, the optimal kernel size of the Gaussian blur was determined, for the MVD validation dataset (Figure 4.2) and the REGION3 validation dataset (Figure 4.3) independently.

4.2 U-Net

In order to achieve the optimal performance of this architecture, variations in several aspects are tested. These variations include changes in the training methods, the architecture, the input, and the output of the model. The U-Net architectures use three previous time steps, separated by ten minutes, as input to generate a prediction.

4.2.1 Recursive Prediction vs Direct Prediction

For training the autoencoders described in Section 3.5.2 and Section 3.5.3, two modalities were discussed. The first option was to train a network to predict the next image in the sequence, meaning that a unique prediction system for one forecast horizon of ten minutes should be trained and used recursively. After the first prediction, the

	Training loss function			
Validation metric	MAE	MSE	SSIM	MAE-SSIM
MAE	**0.06595**	0.06971	0.06754	0.06656
MSE	0.01326	**0.01266**	0.01465	0.01404
SSIM	0.6034	0.5922	**0.6105**	0.6093

Table 4.3: Validation errors predicting at 60 minutes, in the MVD dataset, for U-Net models trained using four different loss functions: MAE, MSE, SSIM, and MAE-SSIM. Evaluated with MAE, MSE, SSIM.

output would be re-entered to the network as the latest input, and with it predict another image, getting a prediction horizon of twenty minutes. Applying the ten-minute model recursively the system would reach all the required forecast horizons. The second option was to train different models separately, so each is specialized in generating predictions for a specific prediction horizon.

Although it takes more time and computational resources to train a model for each prediction horizon, it gets better results than the recursive network, mainly at longer lead times. For the subsequent experiments in this section, the direct model approach will be the used modality.

4.2.2 Training Metric

When training a model, an error metric must be chosen as the loss function to be minimized. This error is used to guide in which direction the model parameters must be updated during training, and it is of considerable importance. In Section 3.3, many metrics were presented, each covering an important aspect to be considered in the predictions. It is a known effect that a model trained to minimize a certain metric can perform differently when evaluated on another metric [32], so in this section, the experiments aim to choose the training loss with the best overall performance.

Four identical U-Net architectures are trained for 80 epochs with the same training setup, with the only difference being the training loss function, which is also the corresponding metric to optimize in the validation dataset (validation curves shown in Appendix C.2). The four metrics that were used are: MAE, MSE, SSIM, and MAE combined with SSIM (which is optimized to reduce MAE in validation).

Table 4.3 shows the errors in the validation dataset for each model. The results show that a model does better when evaluated with the same metric used for training as expected. To get the model with the best overall performance, the percentage increment each model had for a certain metric compared to the best performing model in that metric was calculated. Table 4.4 shows the relative comparison against the model with the best validation metric value. In the average relative values (Average row in Table 4.4), the model trained with MAE has the lowest average validation error. This means that training the U-Net model with MAE would be the best option.

4.2.3 Data Augmentation

With image classification problems, it is common practice to use data augmentation to simulate new data in limited datasets. This approach consists in taking images and modifying their look to simulate new data during training. Networks tend to benefit from it, as having more data during training can prevent overfitting. The modification can be a simple rotation, changing pixel values, or cropping the image. When trying to

	Training loss function			
Validation metric	MAE	MSE	SSIM	MAE-SSIM
MAE	**0.0%**	5.69%	2.40%	0.91%
MSE	4.72%	**0.0%**	15.67%	10.86%
SSIM	1.17%	3.08%	**0.0%**	0.19%
Average	**1.97%**	2.93%	6.02%	3.99%

Table 4.4: Relative comparison against the best metric value based on Table 4.3.

predict cloud movement over a fixed region, data augmentation could be an obstacle, as the network might lose reference from where the clouds come in average.

To test the effects of data augmentation in a U-Net model, 90 epochs of training with and without this approach were applied for the 10-minute forecast horizon over the MVD dataset. The modification applied to the images was a random rotation of 90°, 180°, or 270°. The evaluation was over the validation dataset without data augmentation.

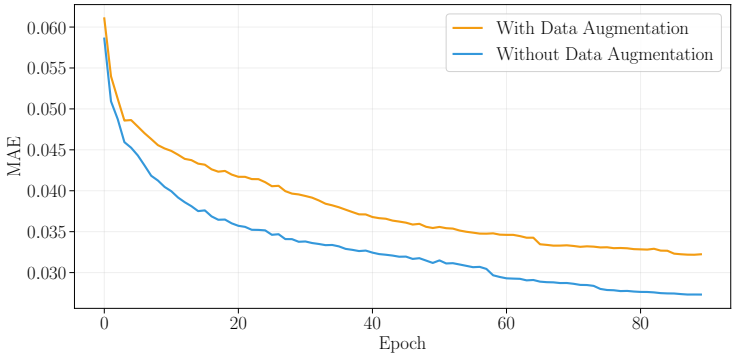
Figure 4.4 shows the learning curves for the model, with and without data augmentation. Using data augmentation shows higher errors in the training learning curve as expected. This is because the model constantly receives modified data making it harder for it to find the underlying patterns and structures in the images. However, the performance over the validation dataset is not benefited. In Figure 4.4(b) both curves tend to the same value. This kind of data augmentation may not be useful for this problem due to the geographic background, which is easier to learn without the rotations.

4.2.4 U-Net Variations Comparison

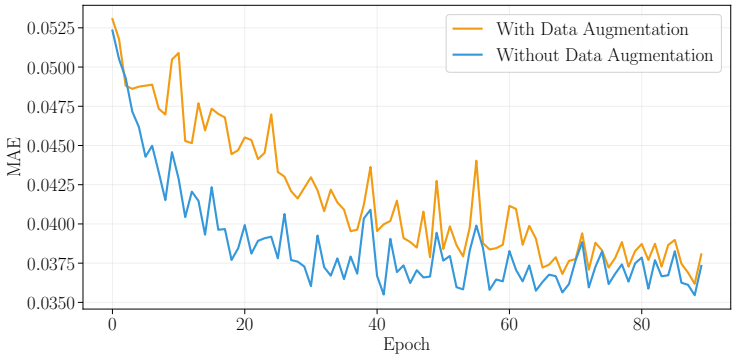
The autoencoder architectures presented on Section 3.5.3 were compared against each other to select the one with the best performance. For this, two forecast horizons were considered (10 and 60 minutes), and each U-Net variation was trained for them. The networks were trained and evaluated on the URU dataset. This was conducted to get an idea of how well they would perform on REGION3, and how much computational power is required for training.

All the networks presented here were trained using the same configuration: 80 epochs maximum, MAE for loss function, Xavier initialization [57] for weights, and Adam optimizer [58] with an initial learning rate (LR) of 1×10^{-3} scheduled to be reduced in half if validation performance doesn't improve for 15 epochs. The training only differs in the batch size, as the maximum value possible for each network, shown in Table 4.5, was selected.

Although the results did not come from intense experimentation, it was useful to reduce the number of U-Net variations to use in the following stages of training and testing. Table 4.6 shows the performance of the U-Net variations over the URU validation dataset for the aforementioned prediction horizons. As can be seen, the U-Net achieves the best MAE and RMSE for both prediction horizons. Another important factor in the selection of one of these models was the maximum batch size possible, which is already low for the URU dataset for all the U-Nets variations. Since the idea was to train using the REGION3 dataset and the images from these are four times bigger than the ones from the URU dataset, the U-Net variations could not be trained with the available resources. This is because the larger the images, the



(a) Training learning curve.

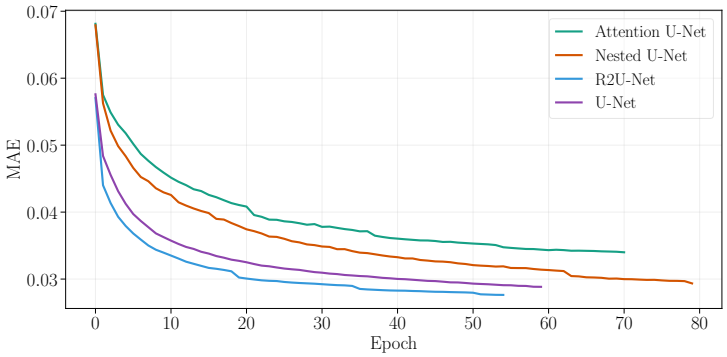


(b) Validation learning curve.

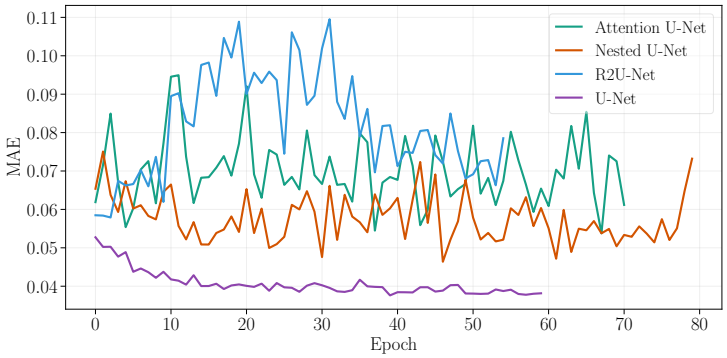
Figure 4.4: Effect of data augmentation in training and validation learning curves, using MAE as loss.

Architecture	**Max Batch Size**
U-Net	7
Attention U-Net	4
Nested U-Net	2
R2U-Net	2

Table 4.5: Maximum batch size allowed for each of the U-Net variations using the URU dataset. This results come from using a single Tesla P100-PCIE-12GB GPU.



(a) Training learning curve.



(b) Validation learning curve.

Figure 4.5: Training and validation learning curves for a ten minutes prediction horizon using URU dataset.

more GPU resources are required, and if these resources are fixed, the batch size needs to be lowered to accommodate the larger images.

Figure 4.5 shows the training and validation learning curves for the models. The takeaway message from these curves is that all three variations showed issues with their capabilities to generalize to the validation dataset, as they have regular decreasing learning curves on the training dataset but the validation ones are noisy. Refer to Appendix C.3 for the learning and validation curves for the 60 minutes lead time.

Taking into consideration the points mentioned before, it is clear that the original implementation of the U-Net outperforms the other variations. In the following, the U-Net is the autoencoder that was chosen to be tested and improved.

4.2.5 Number of filters and batch size trade-off

Because of how the U-Net architecture is set, the number of filters along the network is directly proportional to the number of filters in the first layer. The analysis presented in Section 3.5.2 explains the memory requirements and computational cost used when training a U-Net network. The default number of filters in the first layer is 64, but it was of interest to try other options as changing the complexity of a model can impact the performance.

	10 min		60 min	
Model	MAE	RMSE	MAE	RMSE
U-Net	**0.0384**	**0.0755**	**0.0858**	**0.1389**
Attention U-Net	0.0538	0.0933	0.0919	0.1487
Nested U-Net	0.0464	0.0837	0.0923	0.1507
R2U-Net	0.0579	0.1091	0.1093	0.1718

Table 4.6: Performance of the U-Net variations over the URU validation dataset for the 10 and 60 minutes prediction horizons.

Initial Filters	**# Parameters**	**Max Batch Size**
16	1,080,929	7
32	4,318,401	3
64	17,262,977	1

Table 4.7: Number of total trainable parameters and maximum batch size allowed according to the number of initial filters. This results come from using a single Tesla P100-PCIE-12GB GPU.

To get meaningful results, four networks with different amounts of filters were trained on the REGION3 dataset for a prediction horizon of 60 minutes. From those networks, one has 64 initial filters, another 32, and two have 16 with different batch sizes. The reason to add an extra 16 initial filters network is to test the impact of the batch size used for training.

Table 4.7 shows the number of parameters the U-Net has, as a function of the number of initial filters. Additionally, the third column shows the maximum batch size possible on a Tesla P100-PCIE-12GB GPU. As can be seen, the greater the number of filters, the larger the number of trainable parameters, and the lower the maximum possible batch size.

The learning curves in Figure 4.6 and Figure 4.7 show the huge effect the batch size has in models training. The architecture with 64 initial filters and batch size one overfits, as it has the best performance on the training dataset with a learning curve consistently lower than all the other models, however on validation the learning curve is noisy. This is due to the batch size, as Batch Normalization is useless in this case, and the regularizing effect of batch normalization is gone. The comparison between the two models with 16 initial filters makes this effect clear, with batch size equal to one, the model is capable to overfit, but on validation, this model shows an important downgrade from the model with batch size seven.

The two models with acceptable performance on the validation dataset are the ones with 32 and 16 initial filters, and batch size greater than one. There is a trade-off between complexity and batch size, with higher complexity requiring a lower batch size. The more memory required from the GPU for the model training, the less memory is left for the batch of data, causing the batch size to be reduced. Choosing the model to use for the final results, came to see which had better performance on the validation dataset. Table 4.8 shows the best performance for each network. The architecture with the smallest number of parameters and largest batch size has the best performance in validation, so the U-Net network with 16 initial filters and a batch size of 7 is selected as the best performing model in this stage, and will be the one used in the following.

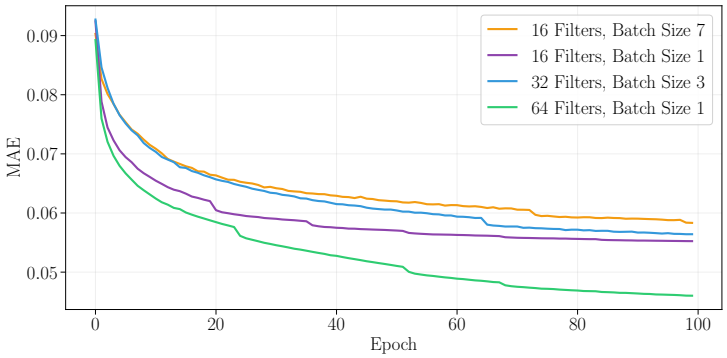


Figure 4.6: Training learning curves for the U-Net with different initial filters and batch size, over the REGION3 train dataset.

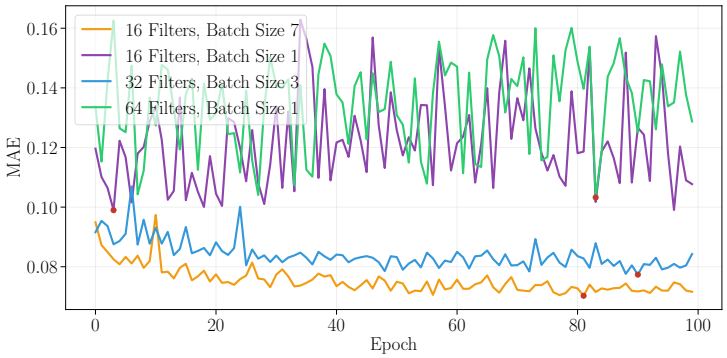


Figure 4.7: Validation learning curves for the U-Net with different initial filters and batch size, over the REGION3 validation dataset. The red dots mark in which epoch the model reached the minimum error.

		60 min		
Initial Filters	**Batch Size**	MAE	MSE	SSIM
16	7	**0.0703**	**0.0166**	**0.5877**
16	1	0.0990	0.0253	0.5162
32	3	0.0774	0.0178	0.5567
64	1	0.1032	0.0277	0.5367

Table 4.8: Best performance on validation for the 60 minutes prediction horizon for the U-Net with different initial filters and batch size.

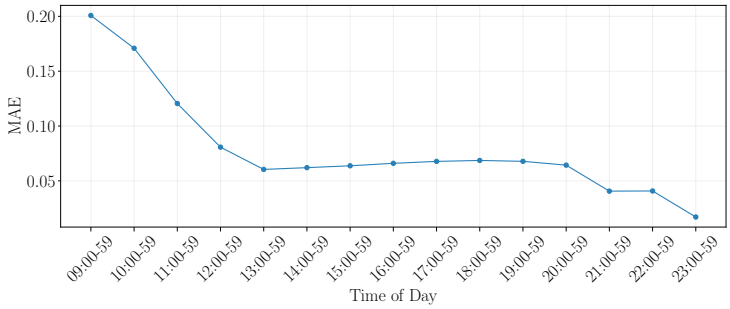


Figure 4.8: MAE at different timestamps for predictions an hour ahead, using U-Net (16 filters) on the REGION3 validation dataset. Each point represents the mean MAE for images between the marked time and the next. Timezone UTC-0.

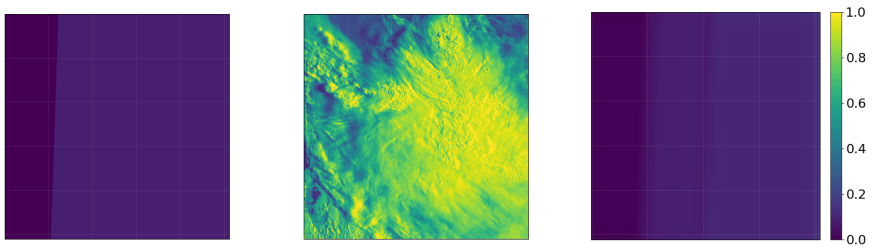
4.2.6 Sunrise and Sunset Effect

Further studying the performance of the chosen U-Net from Section 4.2.5, it was noticed that the model had a different performance according to the time of the day. Although in Section 2.2 night images were filtered, it was noticed that sunrise and sunset (S&S) hours also add a particular behavior, as these contain images with pixels for which $\cos(\theta_z)$ is under 0.15, meaning these are not in full daylight. Taking the U-Net with 16 initial filters and a one-hour prediction horizon, the mean MAE error in non-overlapping 60 minutes intervals throughout the day was calculated on the REGION3 validation dataset. The results shown in Figure 4.8 demonstrate that during the start of the day the error is higher than in other hours, while at the end it tends to be lower.

To understand the reason for this difference, an example of a prediction during sunrise and sunset is presented. The example of the sunrise problem is shown in Figure 4.9, where the model has as input dark images, but the output should be an image with full daylight, which as seen in the example could contain clouds. However, it is impossible for the network to accurately predict the clouds at the start of the day, so the output is an image similar to the last input which is far from the ground truth, generating a large error. During sunset, the inverse issue happens. As shown in Figure 4.10, the model has as input dark images and needs to predict also dark images, so it is easier to generate correct predictions.

Given that solar power generation depends on the time of the day, not all predictions of solar irradiance carry the same weight. To evaluate the impact of using these S&S images on the overall U-Net performance, an experiment was run. Two U-Net architectures were trained for ninety epochs over the URU dataset. The first one uses the S&S images in the training set and the second one only full daylight images. This second dataset consists of 70% of the non-filtered one. In both cases, the models were evaluated on the validation set in the same manner, using S&S images, and without.

Figure 4.11 shows the training learning curves for the model trained with and without sunrise and sunset images. As can be noticed, training without S&S better minimizes the training loss. This is coherent with the previous explanation, since the network inevitably fails at predicting the sunrise and sunset images when training with sunrise and sunset images. The results from validation are shown in Figure 4.12. For both models, validation improves when evaluating on the dataset without sunrise and sunset. This result is expected since, as stated before, for these images, the models have no information to predict the change in the images. Furthermore, the model trained without sunrise and sunset has a more stable training than the one trained

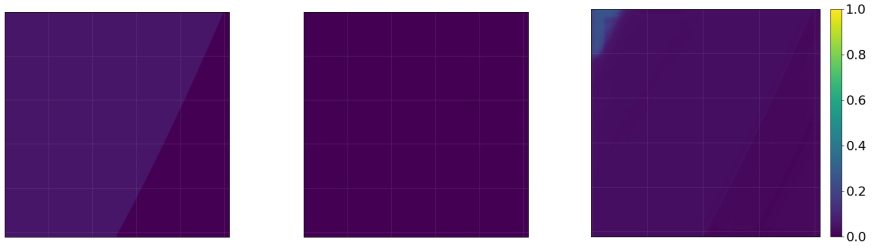


(a) Image from September 27, 2020 at 9:40 (UTC-0).

(b) Image from September 27, 2020 at 10:40 (UTC-0).

(c) Prediction of (b).

Figure 4.9: Prediction made with the U-Net (16 filters) over REGION3 during sunrise. Image (a) is the last input, and image (b) is the ground truth one hour after (a). Image (c) is the prediction of (b).



(a) Image from July 29, 2020 at 20:50 (UTC-0).

(b) Image from July 29, 2020 at 21:50 (UTC-0).

(c) Prediction of (b).

Figure 4.10: Prediction made with the U-Net (16 filters) over REGION3 during sunset. Image (a) is the last input, and image (b) is the ground truth one hour after (a). Image (c) is the prediction of (b).

with, and since predicting sunrise and sunset is a separate problem and the objective is to predict cloud movement, it was decided to train only using full daylight images and evaluate in the same manner for the rest of the experiments.

4.2.7 Network Output

The first reasonable approach is to predict the ground truth directly, this means, if the network has as inputs $X_{t_{-2}}, X_{t_{-1}}, X_{t_0}$, a network generating a prediction for one hour into the future would output $\hat{X}_{t_6}$ as a prediction of X_{t_6}. The modification tested in this section consists of changing the target prediction to the difference between the last image in the input sequence and the desired objective for the given time horizon. The pipeline to train and evaluate a model with this approach is shown in Figure 4.13. In the training phase, the U-Net prediction is compared against the ground truth difference (the difference between the ground truth and the last image in the input sequence). This loss is used to update the U-Net.

While the training process differs from the previous approach, the architecture only differs from the original U-Net (that predicts the image itself, not the difference with the last one) in the output activation function. In this case, the output activation is the hyperbolic tangent (tanh). This change is done to have the values of the output prediction in the $[-1, 1]$ range so when it is added to the last input (which has values

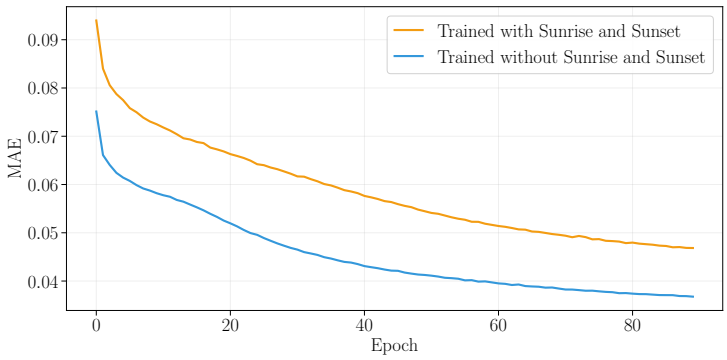


Figure 4.11: Training learning curves for the U-Net trained with and without S&S for a 1 hour prediction horizon over the URU dataset.

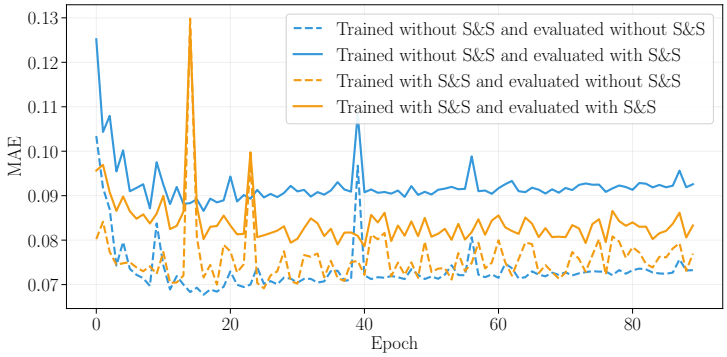


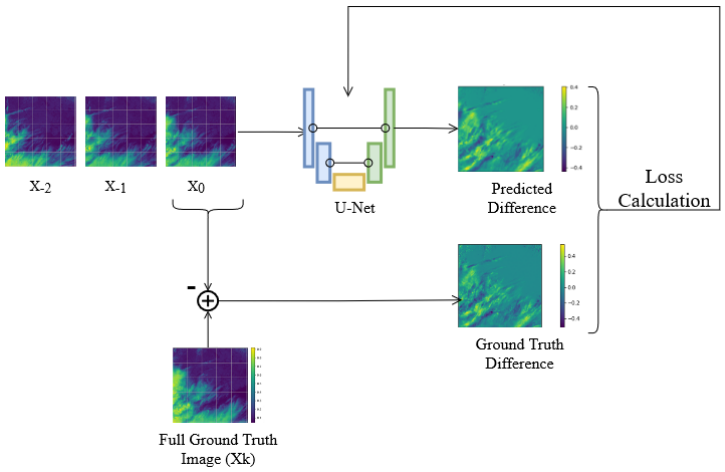
Figure 4.12: Validation curves for the U-Net trained over the URU dataset. The orange curves are trained using the dataset with S&S, the blue ones without. The solid lines are evaluated using the dataset with S&S, the dashed ones are evaluated in the dataset without.

in the [0, 1] range), the full image prediction can change any pixel from 0 to 1 or vice versa. In the following, this model is referred to as U-Net Diff.

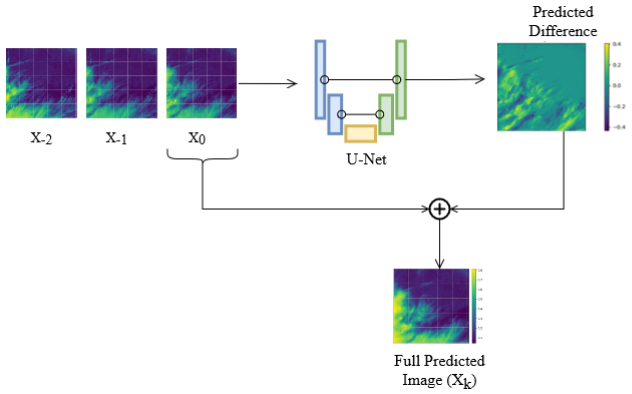
To test this modification, the two types of U-Nets were trained for prediction horizons from one to five hours ahead. The models trained use the results from Section 4.2.5 and Section 4.2.6, so the number of initial filters is 16, the batch size is seven, and only full daylight images are used. The training for both variations is the same as in the previous sections (loss function, initialization, optimizer, etc.) and with 100 epochs. For the learning curves we refer to Appendix C.4.

The final results on validation show similar performance for the two networks, as seen in Figure 4.14. The U-Net Diff does achieve better results for four of the five forecast horizons. Given that both models seem promising, the two are evaluated in Chapter 5 with the test dataset.

To have a better understanding of what the U-Net Diff is doing, Figure 4.15 shows an example of a prediction two hours ahead. Figure 4.15(a) is X_{t_0}, and Figure 4.15(b) is $X_{t_{12}}$. As it can be appreciated, the main cloud located in the bottom left corner moved considerably towards the northeast. The U-Net Diff model detected the mentioned movement from the input and generated Figure 4.15(c). To change the position of the large cloud, it added negative values (shown in blue) in the area closest



(a) Training pipeline.



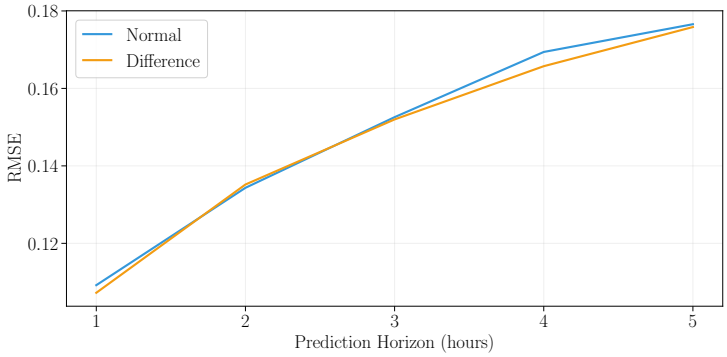
(b) Inference pipeline.

Figure 4.13: Diagrams representing how the images are adapted in order to use the U-Net to predict the difference between the last input and the objective image.

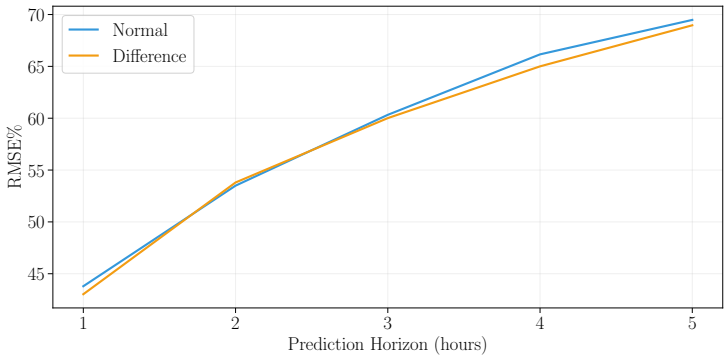
to the southwest of the main cloud and added positive values (shown in red) above the northwest area of the cloud.

4.2.8 Data Inputs

Different types of data were tested as input to improve the performance of the U-Net model. The idea was that extra data apart from the satellite images could help the model generate better predictions. Two types of data were tested in multiple ways. These are the CMV prediction of the corresponding prediction horizon, and the temporal information of the images. Adding extra channels to the input slightly modifies the U-Net architecture by increasing the depth of the filters only in the first convolutional layer. These tests were run on the MVD dataset and with a prediction horizon of 10 minutes. For the U-Net model that used CMV as extra input, the CMV predictions were pre-generated and saved before training to speed up the training

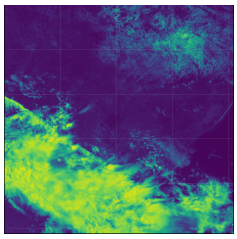


(a) RMSE on validation.

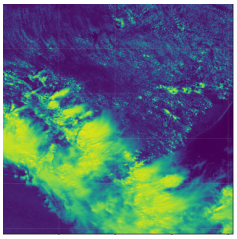


(b) RMSE% on validation.

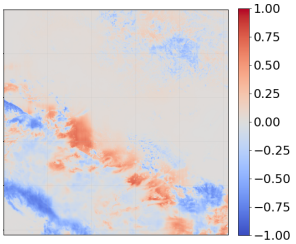
Figure 4.14: Comparison between the U-Net and U-Net Diff on validation for the REGION3 dataset.



(a) Image from December 30, 2020 at 13:40 (UTC-0).



(b) Image from December 30, 2020 at 15:40 (UTC-0).



(c) Predicted difference between (a) and (b).

Figure 4.15: Example of the output of the U-Net Diff when predicting at a 2 hours prediction horizon.

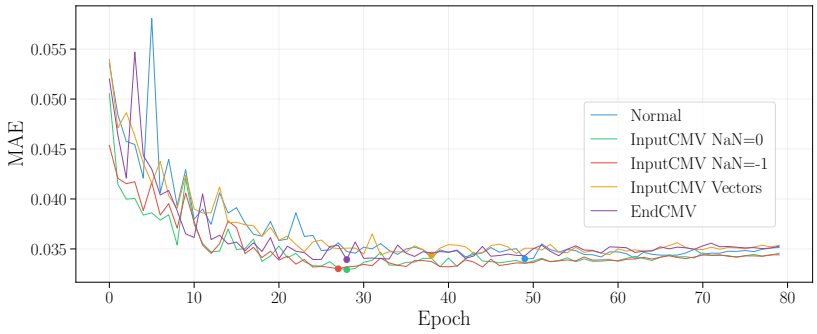


Figure 4.16: Validation learning curves using MAE in the MVD dataset adding the CMV prediction as an input, with a prediction horizon of 10 minutes.

phase.

Four ways to combine the U-Net input with information generated from the CMV algorithm were tested. The predictions of CMV, as mentioned in Section 4.1, can generate undefined values in their borders. The U-Net model can't have undefined values in its inputs, so two modifications of those values were tested. One consisted in replacing them by 0 (InputCMV NaN=0) and the other with -1 (InputCMV NaN=-1). The idea behind these two options is that using 0 as replacement would be considered a normal pixel in the input by the U-Net model, while a -1 value could be considered an exceptional case. In these models, there will be four inputs corresponding to three previous real images and the CMV prediction for the corresponding forecast horizon.

The third option also uses the CMV prediction, however in this case this is concatenated to the input of the last convolutional layer of the U-Net architecture. This option is referred to as InputCMV END, and it replaces the invalid pixels with zero. Adding data at different steps of the model can have different effects, so it is worth testing if adding this information at the end of the model could outperform the previous option of adding it at the start.

The final modification uses the Cloud Motion Vectors directly instead of the predicted image, this method is called "InputCMV Vectors". The two scalar dense fields returned by the CMV are added as inputs to the model. These indicate the direction of movement of the clouds in each direction x and y. This test is intended to understand if adding the vector field as input information could help the U-Net model predict the movement of clouds.

Figure 4.16 shows the validation learning curves in the MVD dataset for the different U-Nets combined with CMV. The InputCMV-Vectors and EndCMV show no improvement compared to the U-Net trained only with satellite images. Meanwhile, the InputCMV-NaN=0 and InputCMV-NaN=-1 show a small appreciable improvement from the rest, reaching almost the same performance between them. Since adding the prediction of CMV to the inputs has a potential improvement for U-Net, the InputCMV-NaN=0 will be considered in the final evaluations in Chapter 5, where it will be possible to see if the improvement is maintained in larger forecast horizons.

The second experiment consists of three different ways to treat temporal information. This follows the proposal of IrradianceNet [41] and MetNet [13], where the timestamp of the images is used as input. To add inputs to the U-Net model they need to have the same dimension as the other inputs. For example, to represent the hour an image was captured, a matrix of H $\times$ W is used, where all elements are set to the hour value. Also, all values are normalized to the same range of the images, [0, 1].

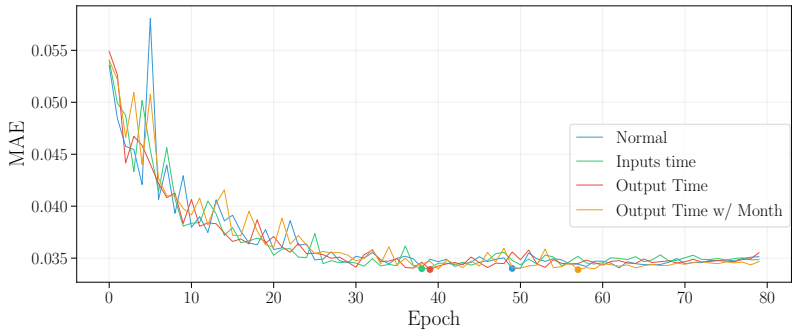


Figure 4.17: Validation learning curves using MAE in the MVD dataset adding the time of images as an input, with a prediction horizon of 10 minutes

That means minutes are divided by 60, hours by 24, days by 366, and months by 12.

The first method adds the minute, hour, and day for each image in the input sequence. This means each sample in the input sequence will have four channels. The sequence can be represented in the following order: Img1, MM-Img1, HH-Img1, Day-Img1, Img2, MM-Img2, HH-Img2, Day-Img2, Img3, MM-Img3, HH-Img3, Day-Img3. The next method adds the minute, hour, and day of the objective prediction horizon. The input consists of the three image sequence concatenated with three more layers: MM-output, HH-output, and Day-output. In the third method, the minute is replaced with the month of the prediction horizon.

Figure 4.17 shows the MAE error during training in the validation MVD dataset of the U-Nets using temporal information. The minimum MAE values are almost identical and there is no real benefit to adding a time reference as model input. The results conclude that using the CMV prediction has the potential to help the U-Net generate better predictions while adding temporal information does not help our data. The reason why the temporal information does not improve performance could be an effect of the filter used in Section 4.2.6, as all remaining images have very similar characteristics and the underlying patterns are hard to recognize. Also, to learn a pattern from the season of the year, more data might be needed. The CMV prediction as input will be further tested in Chapter 5.

4.3 IrradianceNet

For this project, two types of IrradianceNet models were trained. One only uses images as input, and the other the images with spatial (latitude and longitude) and topographic data (elevation map). The latter is named IrradianceNet GEO. As it is trained with patches due to its computational cost, the spatial information provides the network with the knowledge from where the patch is taken. In Figure 4.18 the additional data used for training on the REGION3 dataset is shown. The elevation data is normalized for training by dividing all elements by the maximum absolute value (4196 meters). As mentioned before and explained using Figure 3.12, the model used can predict the intermediate steps. In the original article, there is a step every thirty minutes for every horizon, meaning that for a two-hour prediction there should be three intermediate steps and one more final prediction. In this project, the images have a ten-minute cadence, so we intended to maintain a ten-minute step. This was discarded as it was too computationally expensive, so a thirty-minute time step was

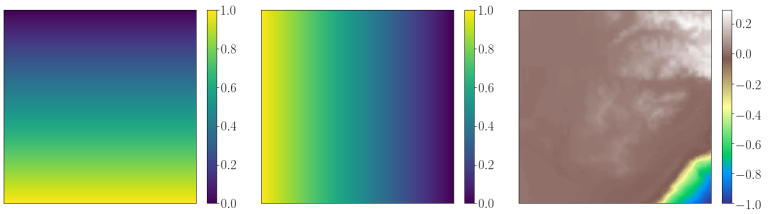


Figure 4.18: Spatial data used for training IrradianceNet GEO on the REGION3 dataset. The first two images inform from where the patch was taken (simulating the geographic coordinates), the third images is the elevation of the region with normalization.

Input Channels	Output Frames	Total Params	Inference Ops (G)
1	2	12,024,321	143.65
1	10	12,024,321	405.58
4	2	12,025,185	143.71
4	10	12,025,185	405.64

Table 4.9: The input channels refers to the information used per patch. For the basic IrradianceNet this value is one, when using geographic data it is four. Input length is four. For a five hour prediction horizon the future frames is equal to ten.

used. To be consistent with the original article, both models use four previous time steps as input to generate a prediction, although in this case, they are separated by ten-minute steps.

As mentioned in Section 3.5.4, this architecture could be trained to output the whole sequence of predictions for a desired horizon (Figure 3.11) or it could be trained to just output the last image in the sequence (Figure 3.12). To test the first approach, a single model was trained to make predictions for all intermediate steps for a time lapse of three hours. However, this model training was too slow, and was unable to reach an acceptable performance for time steps beyond the first one, so this approach was discarded, which is consistent with the original article.

On Table 4.9 the size of the IrradianceNet model with different inputs and outputs is described. The input channels refer to the amount of input information for each image in the sequence. For instance, the image for the basic IrradianceNet consists of a unique channel, and the image with the three spatial sources has four channels. The output frames are the number of outputs required to reach the desired time horizon. As mentioned before for an hour horizon only one intermediate step is done because of the decision to take half an hour steps. Observing the table is clear that there is no significant increase in complexity by using the geographic data, and the biggest changes happen with the output frames as the number of operations needed is proportional to it. It is relevant to point out that the inference operations shown in Table 4.9 are for a single patch (128×128 pixels), so when used on an image of REGION3 this inference has to be done once for each of the 64 patches needed to cover the whole image. Compared to the number of ops needed during inference by the U-Net with 16 initial filters (40.33 G), the IrradianceNet number is huge, and this translates to the time needed to generate a prediction.

The training configuration for IrradianceNet and IrradianceNet GEO is similar to the ones used for the U-Net: 100 epochs, Xavier initialization for weights, and Adam optimizer with a variable learning rate scheduled to be reduced in half if validation

performance doesn't improve for 15 epochs. However, two changes were introduced to be consistent with the training configuration used in the original paper, these are the initial learning rate equal to 2×10^{-3} and the selection of MSE as the training loss. The location of the patches taken during training is random for each batch, on validation the patches are fixed to cover the whole image without overlapping.

The results from running the trained models on the validation dataset are shown in Figure 4.19. It seems that the IrradianceNet GEO is not able to exploit the additional input data as it performs very similarly to the basic IrradianceNet. These results could be attributed to the lower value the geographic information adds to the REGION3 dataset in comparison with the dataset of the original article, which contained a great percentage of the European continent, being a bigger region and with a more variable surface.

4.4 Generative Adversarial Networks

As mentioned in Section 3.5.5, generative adversarial networks and their variants have shown impressive results in image synthesis tasks and many other areas, in particular in the computer vision community. As shown in the literature review of solar and cloud forecast, only one work implements a GAN: the MD-GAN [39]. The authors a priori expected that by improving the stability and the initial accuracy, the MD-GAN would be the best and most consistent model. Based on other references [35], it was found worth trying GANs to predict images more structurally similar to the ground truth. Unlike other architectures previously exposed, generative adversarial networks are two architectures. This presents computational constraints and it was taken into account in the decision-making process.

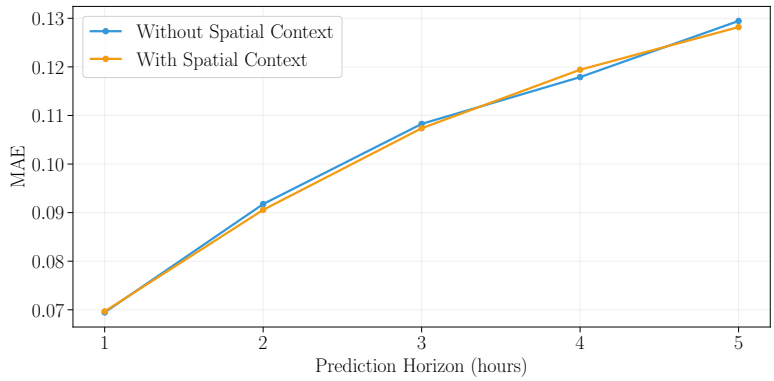
The generator G of a GAN, in its most basic form, takes noise from a latent space with a given distribution as inputs and generates images that are later used as inputs to a discriminator D which determines if the data is real or not. Since the task is to generate more realistic predictions (predictions more similar to the target and less blurred), and taking into account the previous success with the U-Net architecture, the idea of using a U-Net as the generator arose. This modification is also used in a published paper as a solution to image-to-image translation problems [52]. This way, the generator takes the sequences as inputs and not noise from a latent space. Since vanilla GANs are unstable and hard to train, another alternative was considered: the Wasserstein GAN (wGAN). After initial testing, the wGAN architecture was found capable of producing realistic images, and more stable than the vanilla GAN.

The discriminator takes as input one image and returns a vector, later used to determine if the image is real or not. The number of channels in the convolution layers determines the number of trainable parameters. This hyper-parameter is called *Features-D*. Table 4.10 shows the number of trainable parameters for different values for *Features-D* and the memory required for a batch of data given the number of parameters. The latter is important since the computational capacity is finite and the larger the model is, the greater the computational requirements are as seen in the table.

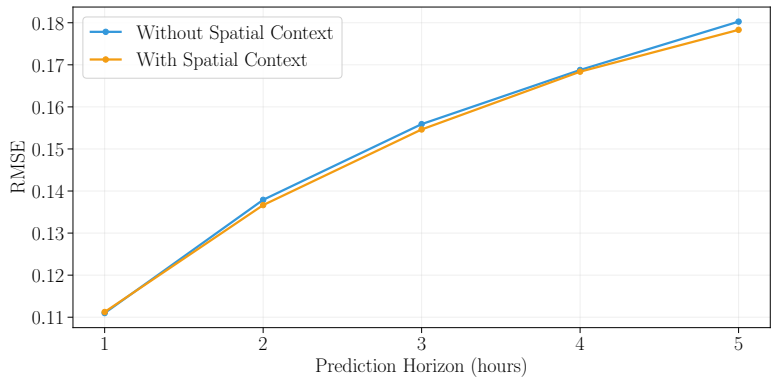
Experiments

This section describes the experiments run for this architecture. These were trained and evaluated over the MVD dataset due to computational constraints.

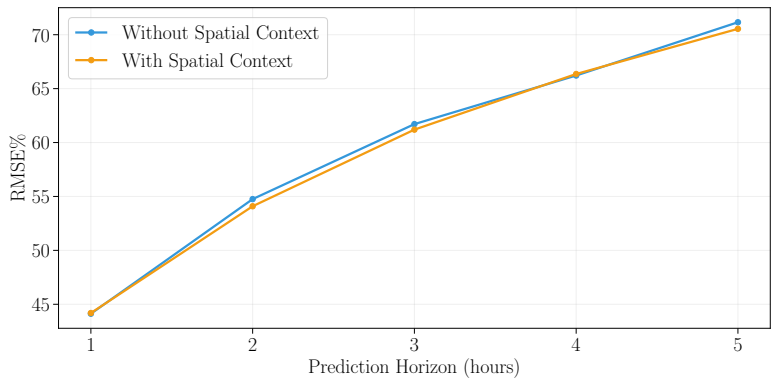
For this model, as in all models in deep learning, there are hyper-parameters to set which determine the performance and learning ability of the model given the task at hand. For the wGAN these hyper-parameters set the speed and ability of



(a) MAE for all horizons.



(b) RMSE for all horizons.



(c) %RMSE for all horizons

Figure 4.19: IrradianceNet and IrradianceNet GEO in the REGION3 validation dataset.

Features-D	# Parameters	Memory (MB)
16	174,801	10.17
32	693,665	21.40
64	2,763,585	47.79

Table 4.10: Number of parameters for the discriminator given the numbers of input features and the memory required for a batch of data given the number of parameters.

the learning process, the size of the discriminator, the number of extra iterations the discriminator is trained [53], and a hyper-parameter that imposes constraints needed to train a good discriminator. To find the best performing model a non-extensive grid-search was performed. In total, 96 models were trained and validated in parallel, consisting of permutations of these hyper-parameters. These models were trained for a lead time of 10 minutes for 15 epochs, for validation the metric used was RMSE. The generators' parameters were initialized with the weights of a pre-trained U-Net on the MVD dataset. These experiments showed that most models could manage to minimize the wGAN loss within the fifteen epochs that they were trained. On the other hand, validating over RMSE yielded poor performance. In order not to cloud the narrative with the details of these experiments, the technical information and the learning curves can be found in the Appendix C.6.1.

The 10 best models according to RMSE on validation were evaluated recursively for up to one hour and compared to the best U-Net and the CMV. Figure 4.20 shows the performance of these models at peak validation performance and in which epoch it was achieved. As can be seen, all models achieved it within a few epochs of training. For the first prediction horizon, all of them performed better than CMV and some models performed slightly better than the U-Net. This performance worsens with every prediction horizon and all models lose at 60 minutes. Figure 4.21 shows recursive validation for these models in the 15th epoch (the last one). Comparing these results to Figure 4.20 makes clear that more satellite-like images, at least in their qualitative appearance, do not necessarily improve this metric, making all models lose against the CMV and the U-Net. The worsening of the results could be since the more training steps, the more the generator loses focus on the forecasting objective, and the closer it gets to only generating satellite-like images.

On the other hand, image quality improves visually with the epochs. To show an example, Figure 4.22 shows the outputs of one of the experiments in the grid-search which managed to minimize the wGAN loss. Figure 4.22(c) sequence corresponds to the wGAN outputs in epochs 1, 3, 5, 10 and 15. As can be observed, these outputs get less blurry and sharper with the epochs, with the outputs from the 10th and 15th epochs being rich in detail, resembling more to the ground truth image than the output at the beginning of the training phase.

The ten models that best minimized the wGAN loss in the grid-search were chosen for further inspection of the results over growing lead times. Using the wGAN loss to choose the best models comes from the fact that this loss is associated with obtaining more realistic images. Furthermore, since minimizing the wGAN loss does not improve the RMSE validation loss, and the idea for the GANs was to predict more realistic images, validation was changed to the SSIM metric for the rest of the GAN's experiments based on the hypothesis that SSIM quantifies the quality degradation of images.

These ten models were trained for a prediction horizon of 60 minutes for 15 epochs. Table 4.11 shows the mean wGAN loss and the mean SSIM loss on validation for the

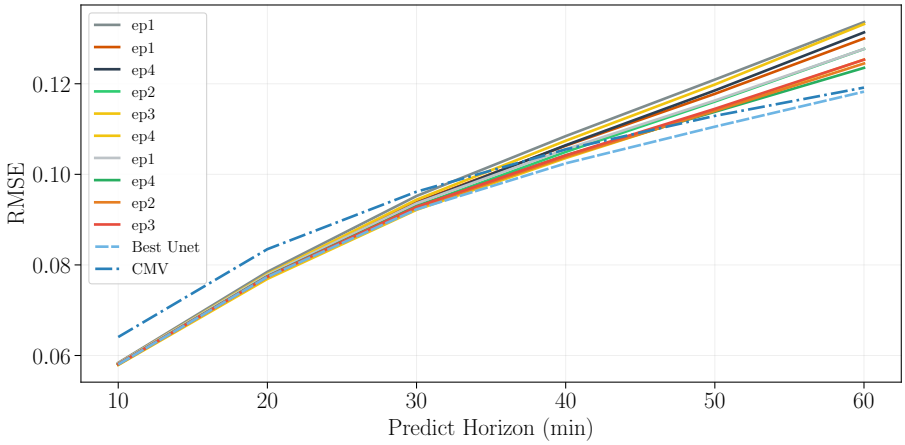


Figure 4.20: RMSE loss on validation evaluated recursive up to 1 hour on the 10 best models according to RMSE loss on validation, in their peak validation performance.

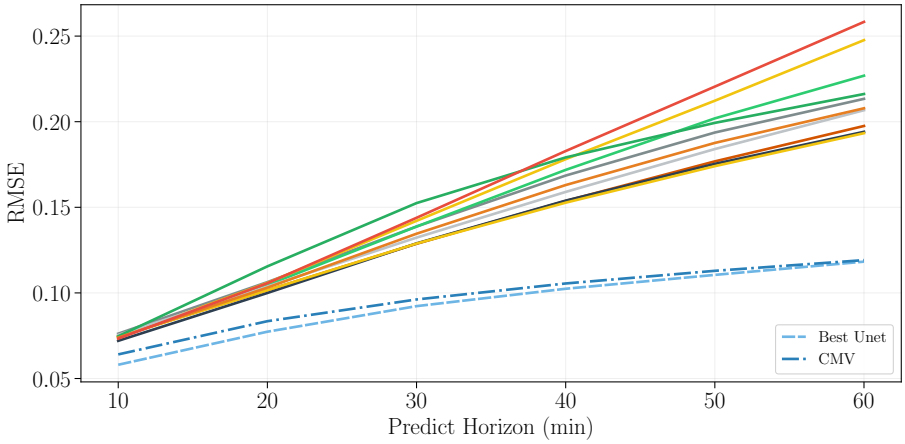
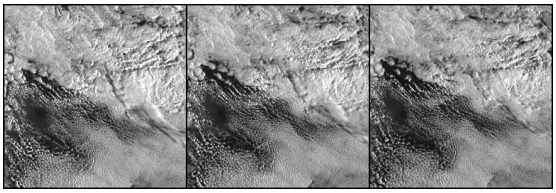


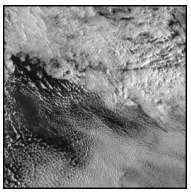
Figure 4.21: RMSE loss on validation evaluated recursive up to 1 hour on the 10 best models according to RMSE loss on validation, trained for 15 epochs.

models in the first and the last epoch of training. As can be observed, it is clear that the models managed to minimize the wGAN loss despite the growing lead time. For validation, on the other hand, some of the models managed to slightly improve this metric in the first epochs of training, but worsens with the epochs, reaching a lower value than at the beginning of training. Refer to Appendix C.6.2 for the detailed training and validation curves for this experiment.

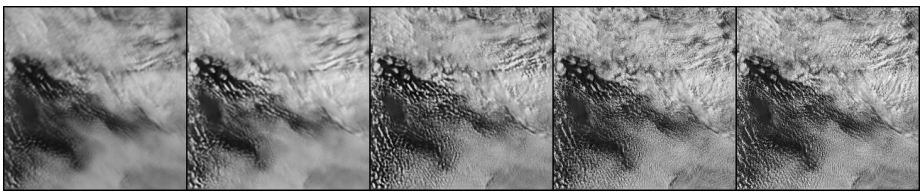
The experiments for the wGAN model were run using a U-Net architecture with 64 initial filters as the generator. To test the effect of lowering the number of initial filters to 16, the ten models previously chosen were trained for a 30 minutes lead time. Given that these are less demanding models in terms of computational resources, they were trained for 40 epochs. This experiment shows that using 16 initial filters lowers the ability of the wGAN to minimize the wGAN loss. Figure 4.23 shows three example outputs from the wGAN in different epochs of the training process. It can be noticed



(a) Input sequence



(b) Ground truth



(c) wGAN outputs. Epochs: 1, 3, 5 ,10, 15.

Figure 4.22: Example of the changes in quality of the outputs of a wGAN from a model from the grid-search as it trains, compared to the ground truth.

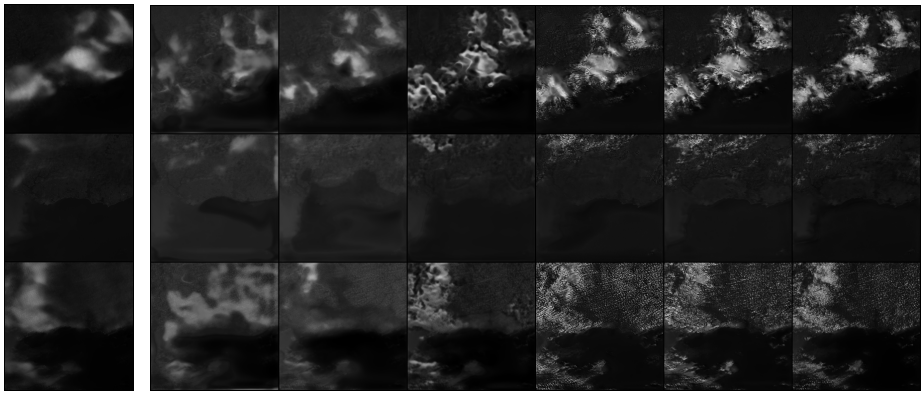
	Epoch	
	1 (first)	15 (last)
wGAN loss	110.40	-6.32×10^{-5}
SSIM val	0.53	0.49

Table 4.11: Mean wGAN loss and SSIM loss in validation for the first and last epochs of training, for the 10 best models from the grid-search, trained for a 60 minutes lead time.

that the outputs from the first epochs present shapes and visual anomalies that do not relate to clouds. The more these models are trained, the outputs resemble more to satellite images, managing to generate more realistic images only in the last epochs. This supports what was previously stated, that lowering the number of initial filters to 16 lowers the ability to minimize the wGAN loss, thus taking a considerably larger number of epochs of training to generate similar quality images.

Models that converge to zero in terms of wGAN loss, do it within a few epochs. As already stated and shown in Table 4.11, minimizing the wGAN loss does not correspond to a better SSIM. This result is also supported by the previous experiment with the 16 initial filters wGAN's outputs. Refer to Appendix C.6.3 for the SSIM validation curves for this experiment. Because of the above, the model that best minimized the wGAN loss at epoch 20 and that better output quality images was chosen as the best model. This model is defined by a set of hyper-parameters. Chosen the best model, the final wGAN models with 16 and 64 initial filters were trained to compare metrics performance. Additionally, a wGAN with 64 initial filters was trained for 100 epochs to predict 60 minutes into the future, to evaluate the performance when training significantly more. Table 4.12 shows the performance for these in their best validation epochs and last epoch (20 epochs) for SSIM, RMSE% and Forecasting Skill, for prediction horizon 60 minutes.

The best SSIM validation performance was achieved in epoch 2 and 3 for the wGAN of 16 initial filters (wGAN16-bestval) and the wGAN with 64 initial filters (wGAN64-



(a) U-Net output

(b) wGAN outputs. Epochs: 1, 2, 3, 5, 10, 40

Figure 4.23: Outputs from the wGAN in different epochs of the training process compared to the output of the U-Net.

Model	SSIM(↑)	RMSE%(↓)	FS(↑)
wGAN16-bestval	0.5086	69.0860	-0.6371
wGAN16-20eps	0.5059	55.3720	-0.0261
wGAN64-bestval	**0.5149**	55.6870	-0.0091
wGAN64-20eps	0.5091	**54.4409**	0.0057
wGAN64-100eps	0.5125	54.7515	**0.0209**

Table 4.12: Validation over SSIM, RMSE% and FS, for the best wGAN model with 16 and 64 filters.

bestval), respectively. The wGAN16 trained for 20 epochs does significantly better on RMSE% and FS and losses by a small margin on SSIM compared to the wGAN16 in the best validation epoch. This happens similarly with the wGAN64. Lastly, the wGAN64-100eps slightly improves some of the metrics but takes a non-negligible larger amount of time to train.

Given that the wGAN64 achieves better images in fewer epochs, better minimizes the wGAN loss (which translates to better minimizing the distance between the distributions of the generated data and the real data), and has the best performance in all four metrics shown in Table 4.12, this model was chosen as the final model to test over the unseen test dataset in the next chapter.

4.5 Selected models for final evaluation

Finally, the deep learning models with the best performance were selected to be evaluated over the test dataset. For the U-Net, the simple U-Net model with 16 initial filters was chosen to test the performance over the REGION3 and MVD datasets. Additionally, the one that predicts the difference (U-Net Diff) was chosen for testing over the REGION3 dataset, and the one that uses the output of the CMV as input (U-Net inputCMV) for testing over the MVD dataset. Both IrradianceNet and IrradianceNet

GEO models were chosen to test on the REGION3 dataset. Lastly, the chosen wGAN model was the one with 64 initial filters, that started training from the pre-trained U-Net weights, and that best minimized the wGAN loss in the grid-search. This model is only going to be tested on the MVD dataset.

Chapter 5

Results

This chapter contains the final results for the performance of the selected models over the testing dataset, unseen in the previous training stages. The focus is on providing the final numerical results to back up the performance assessment conclusions of the work. The evaluation is carried out using the metrics presented in Section 3.3. The results are presented over the MVD and REGION3 test datasets, presented in Section 2.3.2, with special care in avoiding repetitions of the same findings but in different spatial domains. With this in mind, the discussion starts with the REGION3 dataset, which is the more general situation, as none of the forecast horizons are limited by spatial restriction. A separate discussion with the MVD data set is presented afterward. The baseline algorithms used for both datasets are Persistence, CMV, and Blurred CMV. For the CMV algorithm, the evaluation is performed only over valid pixels. On the other hand, for the rest of the models, this is conducted over the entire image, making the comparison unfair from the start, as will be shown in Section 5.1.1.

5.1 Results on the REGION3 Dataset

The deep learning models tested over the REGION3 test dataset are the U-Net, U-Net Diff, IrradianceNet, and IrradianceNet GEO. The evaluations are carried out on a set of metrics to obtain a fair comparison. The REGION3 dataset enables to achieve larger forecast horizons. The size of the image makes the models' training stage more computationally intensive, but after training, the utilization of the models is relatively cheap. The evaluation for this region was performed up to 5 hours ahead with hourly time steps. It should be noted that generating predictions every 10-minutes time step can be easily achieved by training a model for each step and each architecture. However, hourly forecast horizons enable direct comparison with the previously mentioned works related to this section [10] [41]. The results for the different architectures and methodologies are presented in the following.

Figure 5.1 and Figure 5.2 contain the final evaluation in this dataset for each relevant metric and hourly forecast horizon. The first figure contains metrics used in computer vision problems, and the second one metrics related to forecasting.

The largest difference between performances is seen while evaluating with MAE (Figure 5.1(a)), where **the U-Net and U-Net Diff outperform all the other models**. This can be explained since these are the only two models trained to minimize MAE. Observing the RMSE curve (Figure 5.1(b)), this difference is not as relevant, with the Blurred CMV and both IrradianceNet networks having a performance comparable to the U-Nets. This can be attributed to the fact that the enhanced CMV

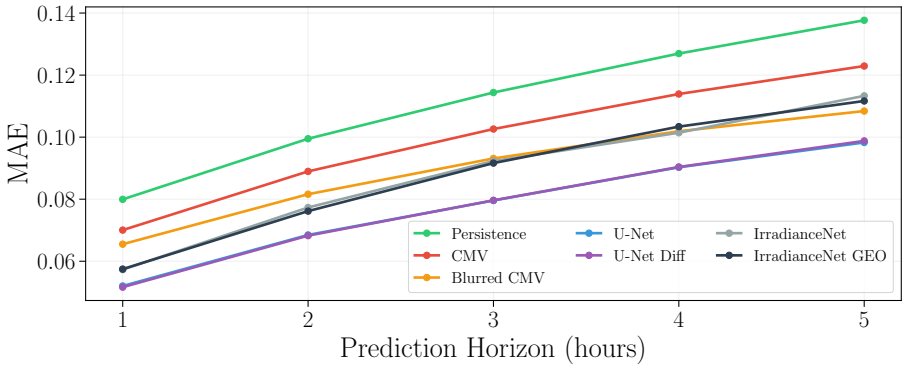
was optimized using RMSE% and the IrradianceNet networks were trained to minimize RMSE. In the evaluation using RMSE, the U-Net Diff outperforms the Blurred CMV by 10% at the one hour prediction horizon, and 4% at the five hours prediction horizon. For MAE, on the other hand, the differences go up to 20% and 9%, respectively. Similarly, the U-Net Diff outperforms the IrradianceNet by approximately 3% in all the horizons for RMSE, and approximately 10% in all prediction horizons for MAE. This effect is not so strong when comparing against a non-optimized algorithm such as the Persistence, with the U-Net Diff having a 35% improvement in performance evaluated with MAE and 32% in RMSE for the one hour mark. Another relevant point is that the Blurred CMV achieves a lower MAE than both IrradianceNet networks for the last forecast horizon (at 5h ahead). On the other hand, this does not happen for the RMSE.

Observing the RMSE% curve (Figure 5.2(a)), and taking into consideration the difference in the datasets and the algorithm optimization done, the performance of the Blurred CMV is consistent with the ones obtained in the work of Aicardi et al. [10]. RMSE% and RMSE show similar results, with the largest difference being the performance of the Blurred CMV, which becomes less competitive in RMSE%. This is caused by the blurring in non-cloudy images, as the ground truth is mostly the region surface, and the prediction is a blurred version. In this case, the RMSE is amplified by being divided by a low value, the mean of the non-cloudy ground truth.

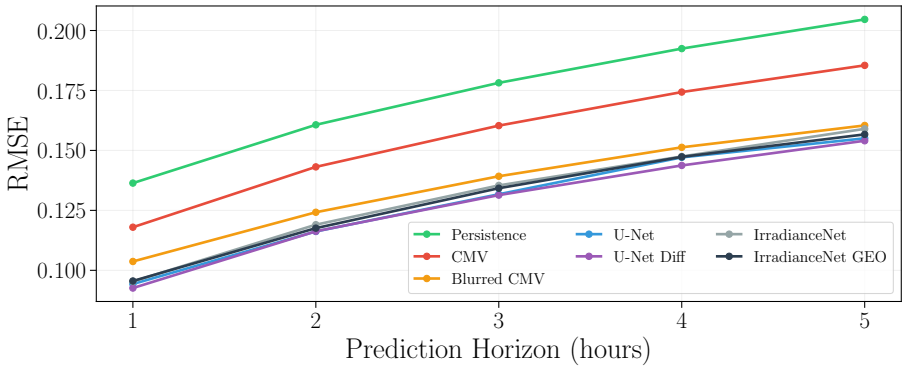
The Forecasting Skill (Figure 5.2(b)) shows how the models perform in RMSE compared to the Persistence algorithm. The U-Net Diff keeps the edge in all the prediction horizons, and the U-Net comes in second place. For this metric, the IrradianceNet networks also outperform the Blurred CMV algorithm.

When using the previous metrics, which always have positive values, it is not possible to know whether the models are mostly predicting values over or under the objective images. The MBD% metric (Figure 5.2(c)) provides this information. As the Persistence algorithm is generating unmodified images it doesn't add a bias to predict lower or higher values, so it is reasonable to see it has MBD% values around zero, which is consistent with previous solar irradiance persistence diagnosis [59] and the work of Aicardi et al. [10]. On the other hand, the deep learning models need to generate a prediction, so they may add a bias. Both the U-Net networks tend to predict values lower than the real ones, similar to the CMV and Blurred CMV. IrradianceNet and IrradianceNet GEO have different behavior, for the first two prediction horizons the errors revolve around zero, but for longer horizons, it tends to predict higher values.

It is important to explore which is the difference between the U-Nets and CMV predictions. By doing so, it is expected to observe in which aspects the U-Nets outperform the CMV. For this example, the U-Net Diff is the used model. Many examples in the test dataset were explored. In this section, only one was selected to be shown and described, refer to Appendix D.1 for more examples. Figure 5.3 shows the images used to generate the predictions, the U-Net Diff uses all three images as input, and the CMV the last two. The predictions are shown in Figure 5.4 for all five-time horizons, next to the ground truth image to observe the expected output. There are many aspects to mention about these predictions, the most relevant ones are the following. One of the main drawbacks of the CMV is its inability to predict cloud extinction, the clouds present in the input at the squares *2D* and *2E* wrongly remain in all its predictions while the U-Net Diff manages to predict its reduction and later disappearance. Another example is the cloud in square *4D*, where the U-Net Diff is capable of predicting that it detaches from the cloud in the bottom-right corner at the one-hour mark and that it gradually vanishes for longer horizons. On the other hand, CMV keeps this cloud constant for all horizons. When predicting movement and deformation, the U-Net Diff detects a drifting of the top-left cloud from the northwest to the



(a) MAE

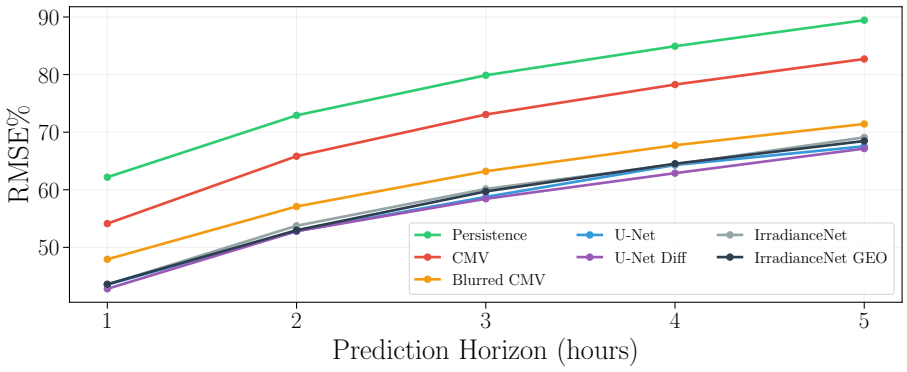


(b) RMSE

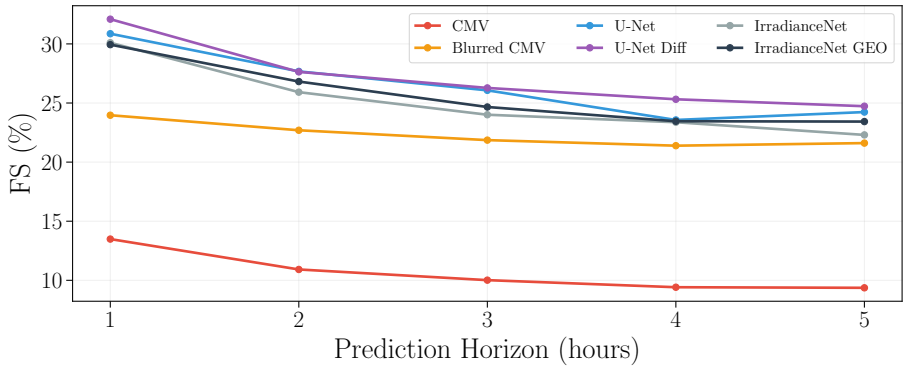
Figure 5.1: Performance comparison in REGION3 test dataset, using absolute metrics. The algorithms evaluated are Persistence, CMV, Blurred CMV, U-Net, U-Net Diff, IrradianceNet, and IrradianceNet GEO. The metrics used to generate the figures are MAE (a) and RMSE (b).

southeast and makes the predictions accordingly. However, it is unable to predict incoming clouds, as seen in the square $4A$ of the five-hour prediction, although this is expected as it does not have the necessary information from the input (studied further in Section 5.1.1). A shortcoming in the U-Net Diff prediction for this example is the vanishing of the bottom-right cloud which persists in the ground truth.

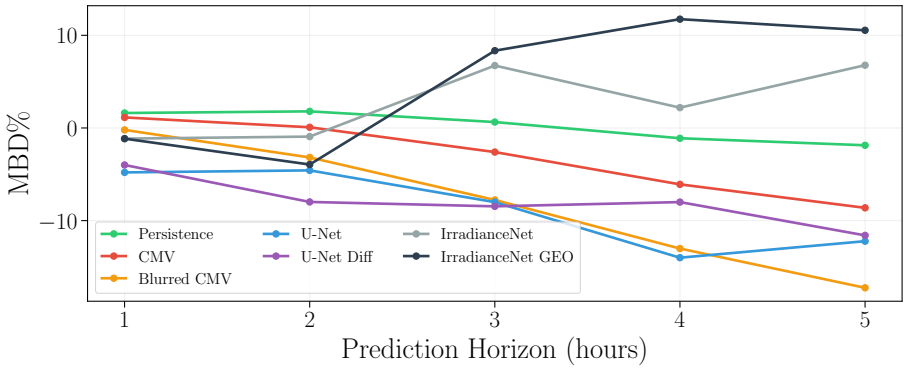
In summary, **the model with the best performance considering all the metrics and time horizons is the U-Net Diff**. About the IrradianceNet architecture, the main takeaway message from its results is that it is surprisingly good for what could be expected from a patch-based model, as this introduces many border errors inside the final prediction (studied further in Section 5.1.1). Finally, it is demonstrated that **the deep learning networks outperform considerably the CMV algorithm, and the U-Net models beat the Blurred CMV in all of the time horizons**.



(a) RMSE%



(b) FS



(c) MBD%

Figure 5.2: Performance comparison in REGION3 test dataset, using relative metrics. The algorithms evaluated are Persistence, CMV, Blurred CMV, U-Net, U-Net Diff, IrradianceNet, and IrradianceNet GEO. The metrics used to generate the figures are RMSE% (a), FS (b), and MBD% (c).

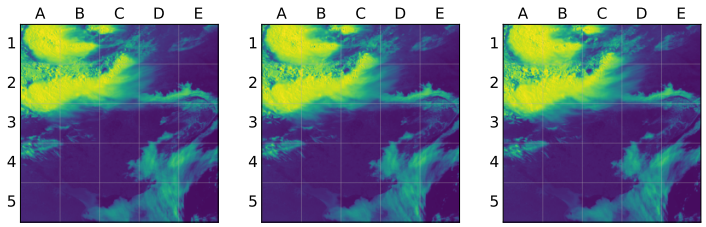


Figure 5.3: Image sequence used as input for the U-Net Diff and CMV algorithm. Captured on October 14, 2020 at 12:20, 12:30, 12:40 (from left to right). Timezone UTC-0.

5.1.1 Temporal and Spatial Analysis

In this subsection, a more detailed analysis of the performance of the final networks is presented. Given that the predictions have a very specific place in time and space, two independent studies on the errors through these two variables are presented. It is important to know when and where the models generate the best and worst predictions, as a diagnosis for further improvements.

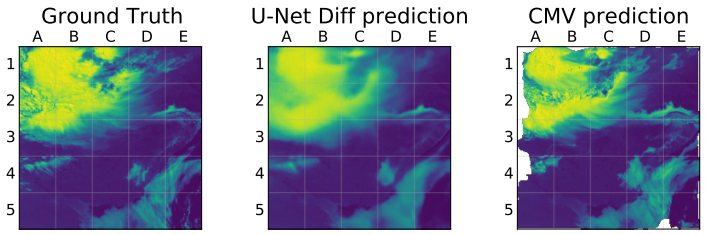
Spatial Analysis

Working with satellite images centered around Uruguay, the region of interest (ROI), sparks the question of whether the different models perform better, equal, or worse in this ROI compared to the rest of the image. It is expected that the models perform better closer to the center of the predictions, as there is less uncertainty on the clouds that could move into or out of that area, while the prediction of incoming clouds through the image borders is a much harder task.

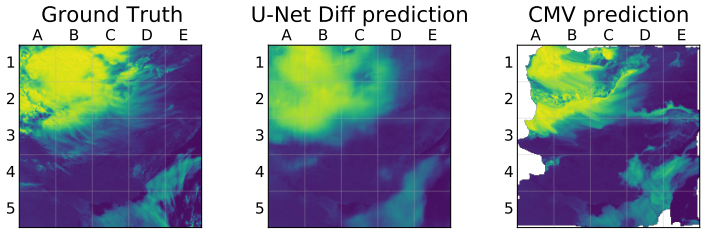
To conduct this experiment, four of the final models were chosen: Persistence, U-Net, U-Net Diff, and IrradianceNet. For each of the five hourly time horizons, an error map is created, composed of the mean error per pixel using the test dataset for three metrics: MAE, RMSE, and RMSE%.

Observing Figure 5.5 and Figure 5.6 it is clear that all four models make on average the largest errors in the top right corner (northeast) of the predicted images, being the persistence algorithm the most disadvantaged. Hence, clouds in that region of the image are harder to predict, and it may be associated with the typical circulation and behavior of cloudiness in that area. It is also noticed that the areas with the highest errors per map are consistent through the five time horizons, with the difference of a systematic increase in the whole map as the prediction horizon grows. When visualizing the error maps of IrradianceNet, the patch-based prediction is very clear because of the presence of a grid-like layout through the region (also visible in the original article [41]). This is caused by the same border issues mentioned before. Apart from the grid effect, the underlying error maps of the IrradianceNet predictions are similar to the ones of the U-Nets.

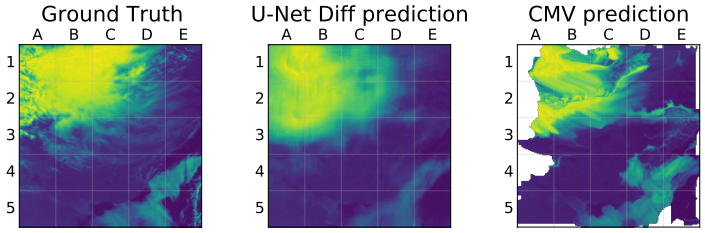
To demonstrate that the networks perform better closer to the image center, using for example the five error maps of the U-Net Diff, the mean error value for windows decreasing in size (as shown in Figure 5.7) was calculated for each prediction horizon. The border sizes go from 0 to 450 pixels, so the evaluation starts at the whole image (1024×1024 pixels) and ends in a window of size 124×124 pixels. The result is shown in Figure 5.8 for all prediction horizons. As the border size increases, the mean MAE for the pixels inside the window decreases. Table 5.1 presents the comparison between evaluating with the smallest and largest border sizes, for each time horizon.



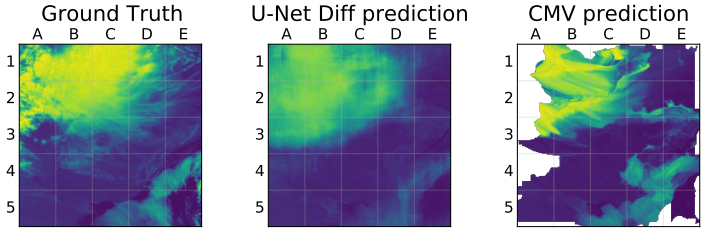
(a) 1 hour.



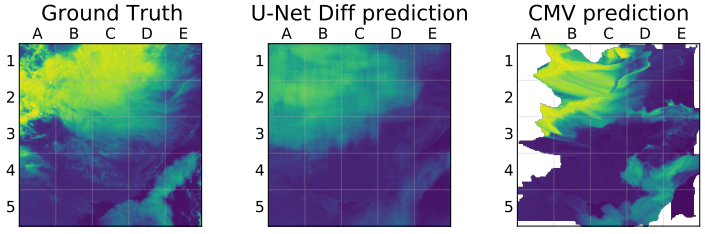
(b) 2 hours.



(c) 3 hours.

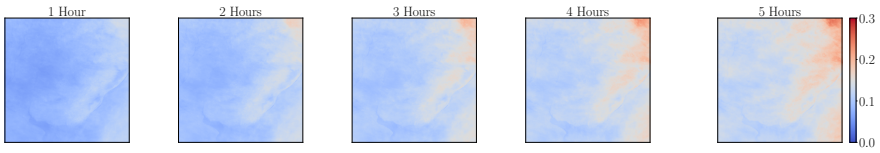


(d) 4 hours.

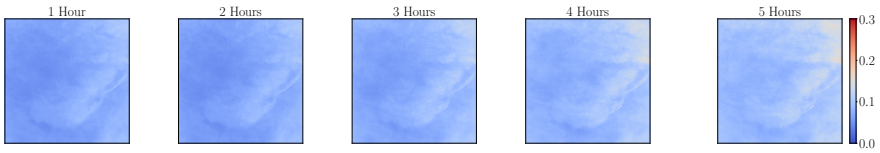


(e) 5 hours.

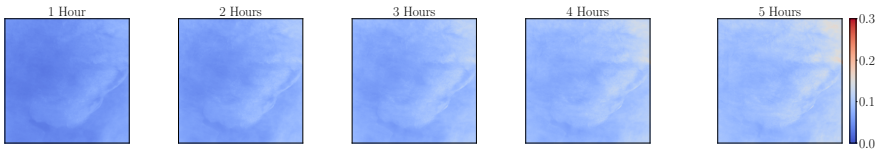
Figure 5.4: Comparison between the ground truth, U-Net Diff prediction, and CMV prediction with inputs from Figure 5.3. For the one to five hours prediction horizons.



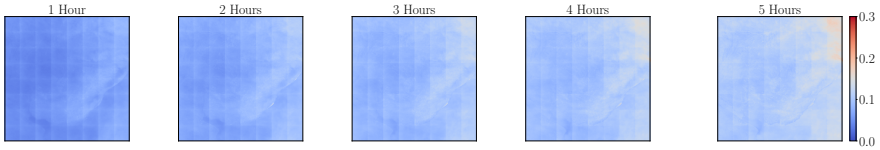
(a) Persistence



(b) U-Net



(c) U-Net Diff



(d) IrradianceNet

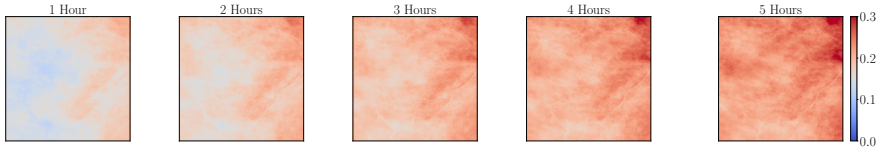
Figure 5.5: Mean absolute error per pixel for all five time horizons. On the REGION3 test dataset.

P. Horizon	Initial value	Final value	Abs. Diff.	Per. Diff. (%)
1 hour	0.0516	0.0436	0.0080	15.4
2 hours	0.0681	0.0572	0.0109	16.1
3 hours	0.0795	0.0672	0.0123	15.5
4 hours	0.0903	0.0763	0.0140	15.4
5 hours	0.0987	0.0836	0.0151	15.3

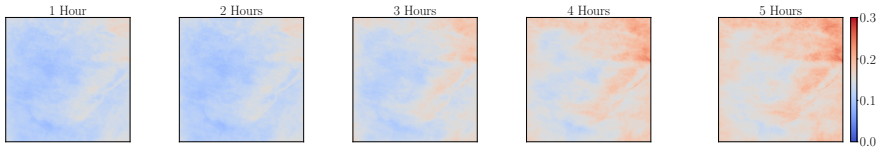
Table 5.1: Difference of the Mean pixel MAE for the whole image against the smallest window (border size = 450).

This shows that as the prediction horizon grows, the absolute improvement is higher. However, the relative improvement stays approximately the same through the five horizons and can be posed as the overall decrease in performance due to the border effect.

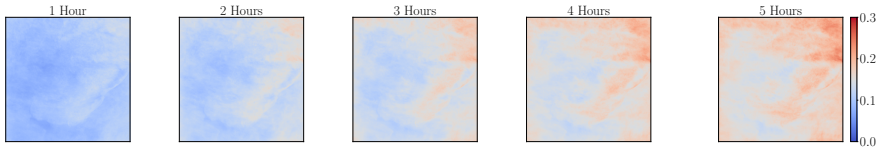
To calculate the RMSE% map for a specific horizon, each pixel in its respective RMSE map is divide by the mean value of that pixel in the ground truth images of the test dataset. Figure 5.9 shows the mean value pixel-wise of all the images used as ground truth for an hour prediction horizon. The mean value map differs slightly



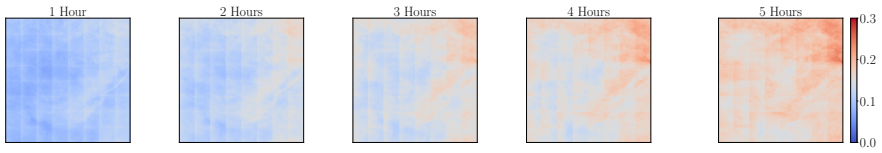
(a) Persistence



(b) Normal U-Net



(c) Diff U-Net



(d) IrradianceNet

Figure 5.6: Mean root squared error per pixel for all five time horizons. On the REGION3 test dataset.

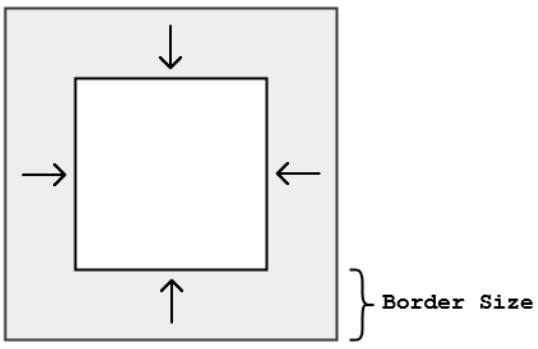


Figure 5.7: Diagram of the window used for evaluation as a function of the border size.

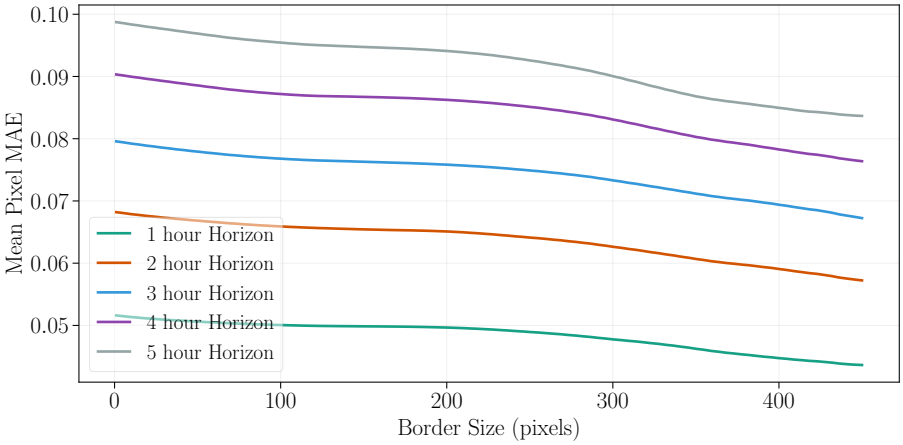


Figure 5.8: Mean error for the U-Net Diff for different windows for the 5 horizons on the REGION3 test dataset.

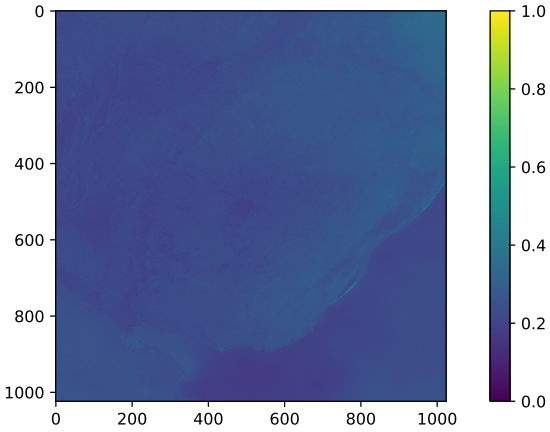
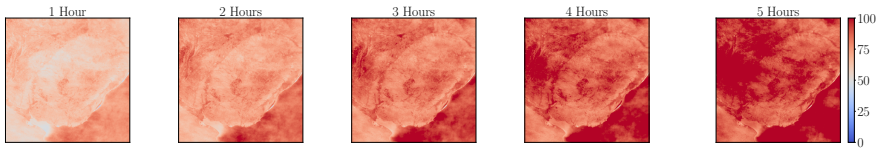


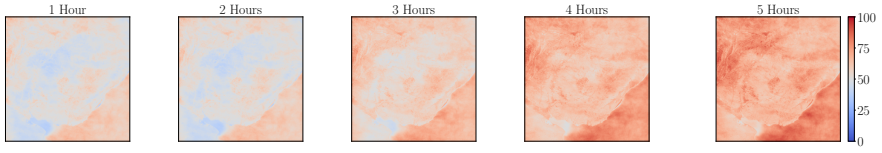
Figure 5.9: Mean value per pixel in an hour prediction horizon.

by the prediction horizon selected, as larger prediction horizons use fewer images. In calculating the RMSE% maps, the mean pixel value map acts as the weights given to each error pixel, suppressing errors in brighter areas and amplifying errors in darker areas. Observing the mean pixel value map (Figure 5.9) and the RMSE error maps (Figure 5.6), a correlation can be found between the brighter areas of both. That correlation is the reason for the more homogeneous relative error maps in Figure 5.10. The errors over the sea (bottom-right corner) are much more relevant than before, and the errors in the top-right corner of the image no longer show the highest values. It can be noticed in Figure 5.10 that the U-Net Diff model achieves better spatial prediction ability than the U-Net at the one-hour lead time.

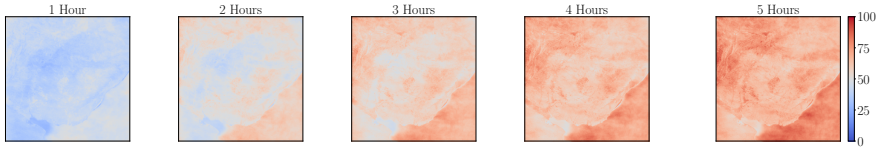
As mentioned before, the RMSE% error maps are more homogeneous than the RMSE ones. This causes that taking smaller windows for evaluation does not have the same impact as before. The curves on Figure 5.11 show how the lines no longer



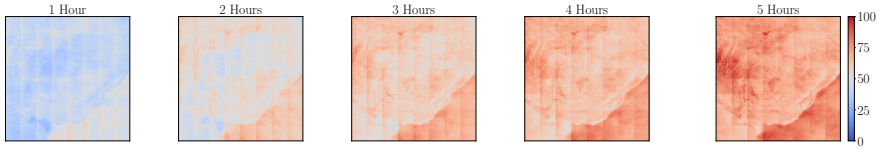
(a) Persistence



(b) U-Net



(c) U-Net Diff



(d) IrradianceNet

Figure 5.10: Mean relative root squared error per pixel for all five time horizons. On the REGION3 test dataset.

P. Horizon	Min RMSE%	Optimal Border Size
1 hour	39.0	273 pixels
2 hours	49.2	359 pixels
3 hours	56.1	359 pixels
4 hours	61.2	359 pixels
5 hours	64.7	368 pixels

Table 5.2: Optimal border size for each prediction horizon using RMSE% as metric.

continually decrease, but now there is an absolute minimum for each horizon (shown in Table 5.2). Comparing the initial and final values on Table 5.3 there are lower values in the last evaluation window but the percentage differences shrink considerably from the Table 5.1.

After the analysis, it is verified that the models make the largest errors in the borders of the output images. This expands the results observed in Section 5.1, as the Deep Learning models are clearly at disadvantage against the CMV ones because these are only evaluated in the valid values of its predictions, basically eliminating

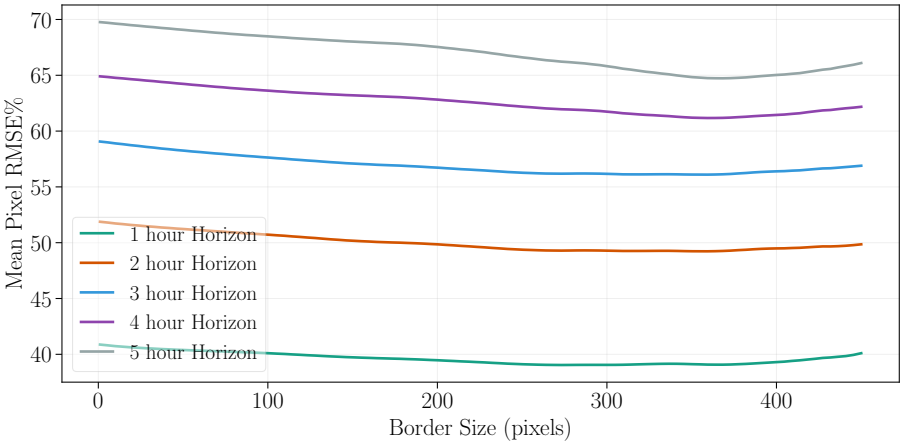


Figure 5.11: Optimal cropping for the test dataset for RMSE%. Done using U-Net Diff.

P. Horizon	Initial value	Final value	Abs. Diff.	Per. Diff. (%)
1 hour	40.9	40.1	0.80	1.89
2 hours	51.9	49.9	2.00	3.90
3 hours	59.1	56.9	2.20	3.67
4 hours	64.9	62.2	2.73	4.20
5 hours	69.8	66.1	3.67	5.26

Table 5.3: Difference of the mean pixel RMSE% for the whole image against the smallest window (border size = 450).

the borders as these are mostly NaN values. Another conclusion is the enormous importance of using larger images, in order to have large borders around the region of interest and be able to reach longer prediction horizons with less uncertainty.

Temporal Analysis

In this section, the performance of the U-Net Diff is measured according to the time of the day the image was captured. The images are divided by the hour they were captured. The hours of the day covered by the predictions start at 10:00 and end at 21:59, using the timezone UTC-0 (Uruguay is in the timezone UTC-3). There are eleven bins, each one representing an hour starting at minute 0 and finishing at minute 59.

As it was made clear before, not all hours of the day are equally important for solar energy production. Because of that, in Section 4.2.6 the decision to remove images captured during sunrise and sunset was taken. Even after that filter, the resulting images in the test dataset still show a tendency to be slightly brighter and darker at the start and end of the day respectively, as shown in Figure 5.12.

Using RMSE to measure errors, Figure 5.13 demonstrates that the model has similar performance for all hours of the day. Except for the 1 hour prediction horizon, the best performance is when the predictions are for images from 21:00 to 21:59 as can be seen in the figure. On average, the hours with the worst performance for the predictions are from 17:00 to 18:59, except for the predictions in the largest time

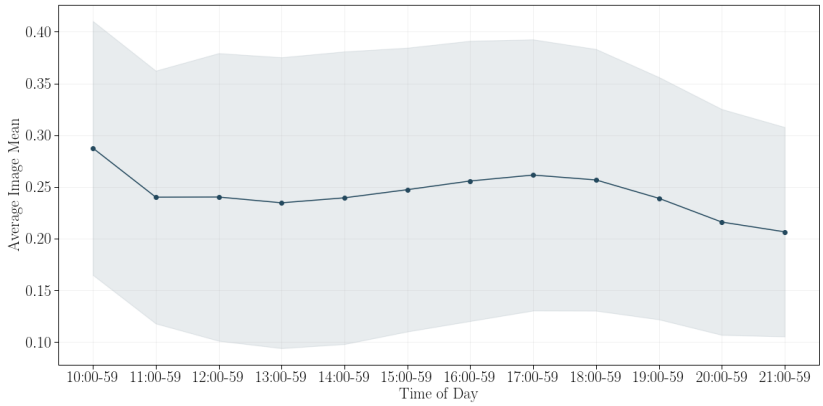


Figure 5.12: Average image mean with the standard deviation for each hour of the day covered in the dataset. Timezone: UTC-0.

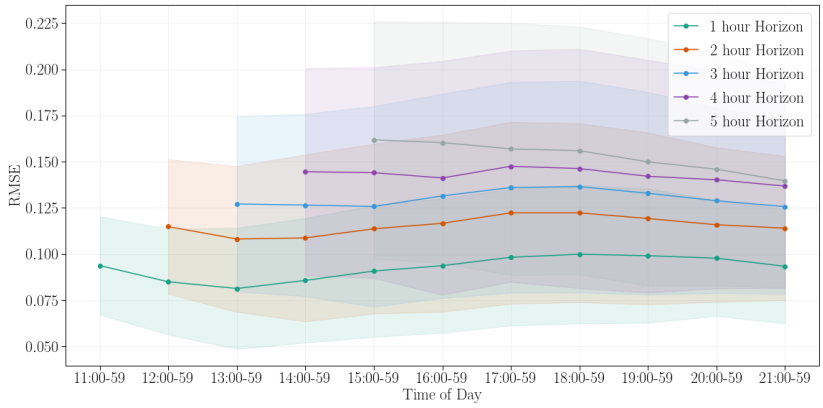


Figure 5.13: Mean RMSE value with the standard deviation for all images in hourly non-overlapping time frames, for all five prediction horizons, using the U-Net Diff for each hour of the day covered in the dataset. Timezone: UTC-0.

horizon, where the worst hour is from 15:00 to 15:59.

The curves when using RMSE% as the evaluation metric have a different shape from the ones using RMSE, as seen in Figure 5.14. The absolute error of the predictions tends to be higher at brighter hours, so dividing by the mean of the image suppresses the error, and dividing by the mean during darker hours (end of the day) amplifies he percentage error.

From the results shown in this section, it is concluded that the U-Net Diff model does have different performances throughout the day. However, the variability from the average value in each hour is not too significant. This was expected as the more difficult images (those containing sunset and sunrise) were filtered out to train and evaluate this model, so they all have very similar properties independent of the hour.

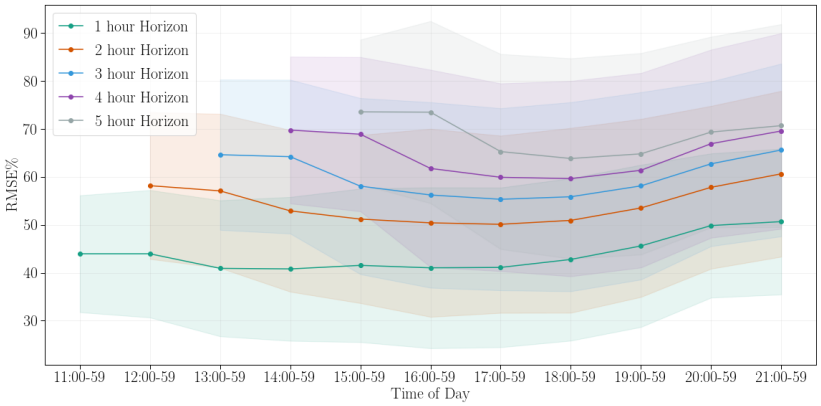


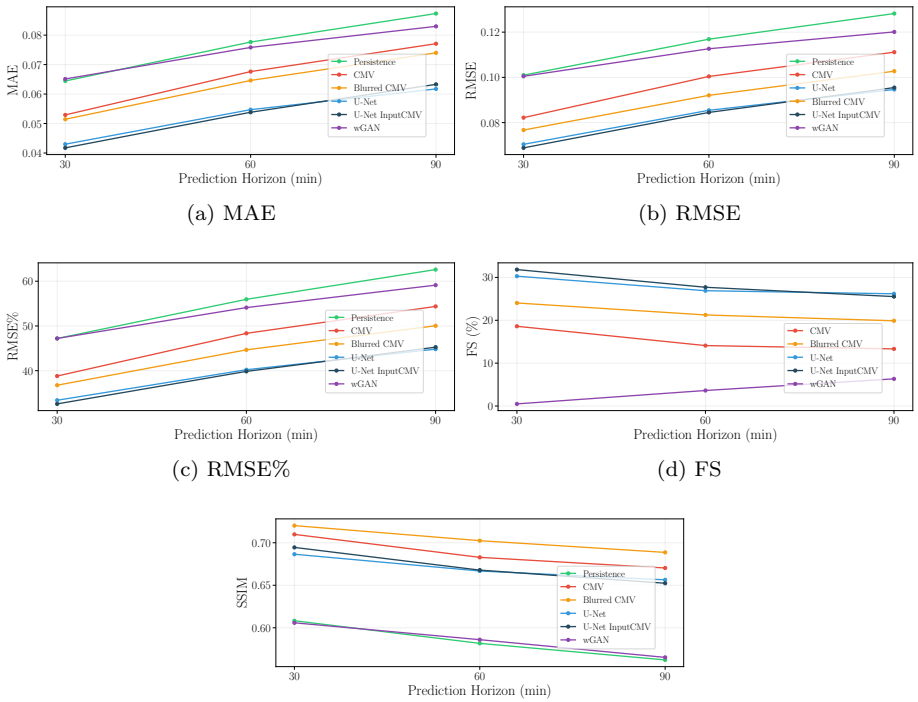
Figure 5.14: Mean RMSE% value with the standard deviation for all images in hourly non-overlapping time frames, for all five prediction horizons, using the U-Net Diff for each hour of the day covered in the dataset. Timezone: UTC-0.

5.2 Results on the MVD Dataset

This section covers a different discussion from the previous section, with the results being on the MVD test dataset. On this dataset, the U-Net, U-Net InputCMV, and wGAN deep learning models were tested. The wGAN model was chosen for this dataset because of the computational restrictions, given that the GANs are two architectures making its training computationally intensive in comparison with the other methods. The U-Net InputCMV was tested over this dataset since it is less computationally expensive than testing over REGION3. The metrics over which the models are evaluated are MAE, RMSE, RMSE%, FS, and SSIM, and given the size of the MVD region (256×256 pixels) and the clouds' speed, the prediction horizons are 30, 60 and 90 minutes. SSIM is included here as a metric that is more adequate to evaluate the texture and realism of the predicted images, as the wGANs are explored with this dataset.

Figure 5.15 shows the results. **The U-Net models show the best performance on all metrics, except SSIM**. The wGAN model shows poor performance, losing in metrics against the other deep learning models. Comparing the RMSE% errors of U-Net InputCMV versus U-Net at 30, 60 and 90 minutes there are the following differences: +2.49%, +0.93%, -0.97%. This means that the U-Net InputCMV model is better at predicting in the shorter prediction horizons but losses at the larger horizon when compared to the normal U-Net. This could be attributed to the CMV prediction losing accuracy for larger forecast horizons. According to these results, using the U-Net InputCMV would be an improvement for the first hour of predictions. Since the interest of the models is to predict also at the longer forecast horizons, which are all of the same relevance, the U-Net InputCMV model cannot be considered an improvement.

It is of interest to analyze examples of the outputs of these models. Figure 5.16 shows a sequence example for the three deep learning models evaluated over the MVD test dataset. For the first two, the input sequence is three images corresponding to 15:30, 15:40, and 15:50, and the ground truth is the image corresponding to thirty minutes ahead, meaning 16:20. The prediction of the models is the rightmost image. For the U-Net InputCMV model, the output of the CMV is concatenated to the input sequence. Examining the outputs, it is clear that the models make different predictions.



(a) MAE (b) RMSE

(c) RMSE% (d) FS

(e) SSIM

Figure 5.15: Performance comparison in MVD test dataset. The algorithms evaluated are Persistence, CMV, Blurred CMV, U-Net, U-Net InputCMV and wGAN. The metrics used to generate the figures are MAE (a), RMSE (b), RMSE% (c), FS (d), and SSIM (e).

For the U-Nets models, the predicted images are blurred when compared to the ground truth. This diminishes the value of the prediction over small-scale details and clouds. On the other hand, it is also visible the ability to predict cloud displacement. There is a clear movement of the clouds from the left part of the image to the right, which both U-Nets manage to capture. Moreover, these models capture the deformation of clouds, seen in the ground truth image in the middle left part as a blue hole, which the U-Net InputCMV manages to imitate. Contrastingly, the wGAN model outputs detailed and sharp images but fails in predicting any kind of visible movement. This can be due to the lack of constraints to predict movement in the learning process and the number of epochs trained, which the more the number of steps the more the generator steps away from predicting, and the more it gets closer to just outputting images resembling the ground truth ones. Additionally, while being clear that the wGAN outputs images resembling satellite images (preserves details of structure) it loses against the U-Net models in SSIM regardless that these models' outputs are visibly more blurred. This supports the claims stated in Section 3.3.3 against using SSIM as a metric of perceptual similarity for predicted images, because it considers the prediction and texture component simultaneously, which forbids extracting clear conclusions on one or the other.

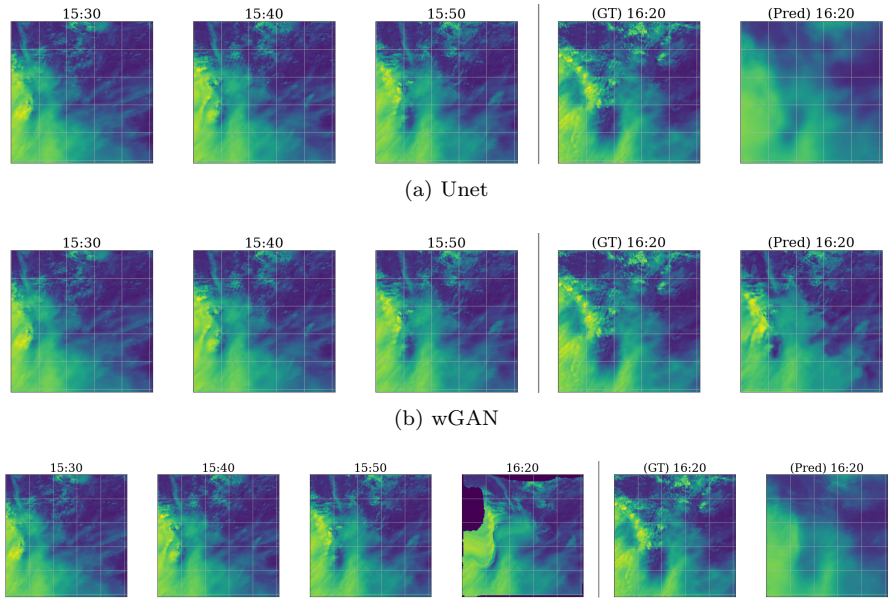


Figure 5.16: Example of a prediction on one hour lead time for the (a) U-Net, (b) wGAN and (c) U-Net InputCMV.

Chapter 6

Conclusions

In this chapter, the ideas and results presented in the previous chapters get a brief closure, and our opinion on the next steps to be considered is also exposed. The conclusions summarize the final state of the work compared to the objectives set at the start, giving a clear reflection on the results presented, and what those results contribute to the R&D area. Finally, there is also a space where our personal experience developing the project is presented.

6.1 Conclusions

Regarding the objectives set in Section 1.6, these were fully accomplished, and to some extent, some were even surpassed. To the best of our knowledge, there was not much research on using deep learning to predict satellite images with the size and resolution used in REGION3, so it was reasonable initially to aim to obtain a competitive model with the current solution provided by the LES. However, in the end, the result was multiple models capable of outperforming the CMV algorithm, even at the five hours time horizon. This proves that deep learning models can be applied to generate a reliable intra-day forecast.

On a technical level, the results show that the U-Net and the U-Net Diff are the best models, with the latter being slightly better. According to the extensive testing, the performance of the U-Net was not improved significantly by adding temporal information, or the CMV algorithm output to the input. The CMV algorithm was notably enhanced by the smoothing technique presented in Section 4.1, which resulted in the Blurred CMV, a model found competitive against the deep learning networks under some metrics. With the GANs, the results were constrained by a trade-off between learning to generate real-looking images or accurate predictions, which resulted in a non-competitive model.

About our contributions to the area, we were capable to adapt a well-known architecture such as the U-Net, proven useful in many types of problems, to our data. With our adaptation, we proved that not necessarily the most complex model has the best performance, as reducing the number of parameters obtained better results. The final model is not computationally expensive, which allows it to generate a prediction in less than a second using a single GPU, and can also run inexpensively on a CPU. This goes against a common trend where research is looking for improvement by using larger models ran on very expensive hardware.

As mentioned in Section 3.4, the article about IrradianceNet [41] was presented at an advanced stage of the project so it was not considered during the initial planning.

Nevertheless, upon reading the article, it was decided it was in our best interest to add this network to our work. The results ended up being a great part of the contributions made, as we could take a state-of-the-art network and train it with large images, and compare its performance against the U-Net.

Finally, we consider that our work could be used by the LES to provide more reliable predictions for Uruguay and the region. With this change, a small step towards a more reliable utilization of our own country's solar energy was achieved. Also, this could be extended to other regions of the world, if provided with the necessary data, and help other countries improve their use of cleaner energies, which in turn can help a greater adoption of solar energy in the world's energy mix. This is immediate for all the area covered by GOES-16 images, which includes North America.

6.2 Future work

Taking into consideration the limited time available, it was relevant to give different priorities to the initial ideas based on difficulty and prospect. This caused some proposals with low priorities to not be tested even if they seemed promising. Next, we explain what we consider should be the following steps if this project was to continue within the initial scope, or as feedback for the current R&D line in the LES.

Probabilistic Forecasting

All the architectures used in this project are trained to output deterministic predictions. This methodology lacks the information about the confidence the network gives to the value assigned to each pixel in the prediction, as each pixel only gets one value. Having the confidence level is useful because it allows the user to know, in a quantitative manner, how accurate the prediction is. For example with the deterministic approach, the prediction at the one-hour horizon is taken with the same confidence as the five hours one when this is intuitively incorrect. The probabilistic distribution of predictions at the larger forecast horizons is wider than at the initial horizons, a feature that is not captured by a deterministic forecast.

More and different kinds of images

As mentioned in Chapter 2, all the images used were from the year 2020. After filtering, the final number of useful images was reduced to approximately ten thousand images, which isn't a lot when using Deep Learning architectures. It would be interesting to test how much can the accuracy improve if one whole year's worth of data (or more) was used exclusively for training and another for validation and testing.

The LES, besides the reflectance, does also save other kinds of satellite images. Among these, Infrared images of the region provide a different look at the clouds that could help the networks make better predictions, specially during sunrise. Also, the satellite GOES-16 provides the Cloud Top Height [60] (CTH), which informs of the top height, cloud top temperature, and cloud top pressure for each cloudy pixel, which seems relevant when trying to predict cloud movement.

Remove background from images

The images, as visualized in some examples, show the surface of the Earth quite clearly. Given that the surface is constant and the focus of the project is set on the clouds, the information the background (the Earth's surface) provides might be useless. However, it is still consuming a part of the model resources as it has to add it to the output.

Removing the background could help in the model training, by keeping focus only on the cloud movement.

Future GAN

Although the experiments with GANs did not reach an acceptable performance, other variations are still worth testing, given the complexity and potential of these architectures. FutureGAN [54] is a progressive growing GAN [61] used to predict future frames in video sequences. The generator, which is an autoencoder, takes an input sequence and outputs future frames. The authors cleverly used a progressive growing GAN that improves the resolution of the output frames, prevents mode collapse, and provides a more stable training.

This architecture is of interest to us since, in opposition to our implemented GAN model, the whole sequence of inputs concatenated with the outputs is processed by the discriminator. Then the discriminator uses the Wasserstein distance to determine if the sequence has ground truth output frames or the ones generated by the generator. Because of the latter, future work for this project would include testing the implementation of FutureGAN over our dataset.

6.3 Group Statement

On a personal level, being part of a project for over a year taught us important lessons on teamwork and self-assessment. Keeping a good organization and communication was hard at some moments, specially in a pandemic scenario, but maintaining regular meetings and an agile methodology allowed us to work in a parallelized manner and still be able to give feedback. Respecting each other's opinions, and listening to our tutors was an important aspect of implementing new ideas, and maintaining a high commitment to the project from start to finish.

There was also an improvement in our technical skills in many areas. We ended up with sufficient knowledge to handle complex deep learning architectures, such as GANs, with ease. The long training and testing times needed by the models made us always take into consideration the need for an efficient code, and to save records about all experiments run. Understanding the topic of solar irradiance forecasting, which neither of us had any previous knowledge of, was crucial to making valuable decisions regarding the models' implementation to make them useful for a production environment and not only useful for a proof of concept.

Appendix A

Relevant info of the data

This appendix contains complementary information for Chapter 2. The first subsection presents a preprocessing option for out-of-range pixels. The second explains the data split for the train/validation/test datasets.

A.1 Data clipping

After the normalization mentioned in Section 2.1.2, some images might contain pixels with values higher than 100. This is not desired as pixel values originally are in the [0, 100] range. This phenomenon is observed mostly in images at the start and end of the day. To solve this issue, two solutions were tried in this regard. The first replaced the outlier values with the mean of the image values, and the other replaced them with the value 100. The first option relied on the assumption that outliers contain no useful information and the best estimate would be the image average. This option was not successful, as the prediction model generated images with undesired anomalies, as seen in Figure A.1. This is because both the context and the fact that outliers represent high values were ignored. Substituting the damaged pixel values to 100, as if they were saturating, worked better, and gave better-looking images.

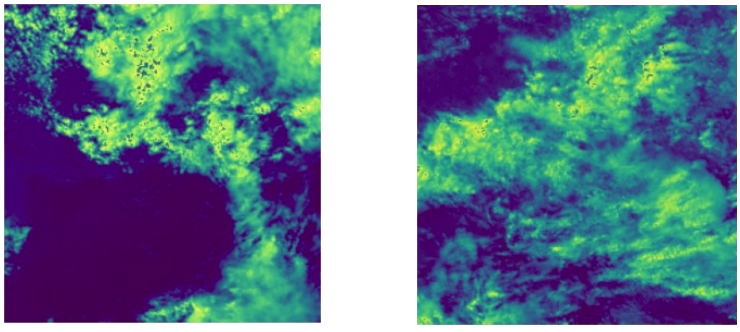


Figure A.1: Two examples of images with pixel values clipped between zero and one hundred, by replacing values over one hundred with image mean.

A.2 Data split

As 2020 was a leap year, the dataset contains images from 366 different days. From those days, 234 were used for training, 40 for validation during training and 92 for testing the final models. Numbering all days from 1 to 366, meaning the 1^{st} of January is day 1, the days were randomly split as it is shown in the next lists. Figure A.2 shows the number of days for each split for each month of 2020.

Train:

[2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 15, 21, 25, 26, 27, 28, 30, 31, 32, 35, 38, 39, 40, 42, 43, 44, 46, 47, 49, 51, 54, 55, 56, 57, 59, 60, 62, 65, 66, 67, 68, 69, 71, 74, 75, 76, 79, 81, 83, 85, 86, 87, 90, 91, 92, 94, 95, 96, 97, 98, 100, 101, 103, 104, 105, 107, 114, 116, 119, 121, 122, 123, 124, 125, 128, 129, 130, 131, 132, 134, 135, 140, 142, 144, 145, 146, 147, 148, 150, 152, 153, 154, 155, 157, 158, 159, 161, 163, 165, 166, 169, 170, 172, 176, 178, 179, 180, 181, 183, 185, 188, 189, 191, 192, 193, 196, 197, 198, 199, 200, 201, 202, 203, 205, 207, 208, 209, 210, 212, 213, 214, 216, 218, 220, 221, 222, 224, 225, 226, 229, 231, 232, 234, 235, 238, 239, 240, 241, 242, 243, 244, 245, 246, 248, 249, 250, 251, 252, 253, 255, 256, 257, 258, 261, 262, 263, 264, 265, 266, 267, 268, 269, 272, 273, 276, 277, 278, 279, 281, 282, 284, 285, 286, 287, 289, 290, 291, 292, 293, 294, 297, 299, 300, 301, 304, 305, 306, 307, 308, 311, 312, 313, 314, 316, 317, 318, 319, 320, 322, 323, 324, 325, 327, 329, 330, 331, 334, 335, 336, 339, 341, 343, 344, 345, 346, 348, 350, 352, 355, 356, 358, 360, 361, 363]

Validation:

[1, 18, 19, 20, 22, 33, 61, 63, 70, 73, 77, 78, 89, 93, 106, 110, 127, 133, 156, 160, 164, 168, 171, 174, 204, 211, 219, 223, 227, 228, 247, 254, 260, 270, 271, 295, 321, 364, 365, 366]

Test:

[4, 13, 14, 16, 17, 23, 24, 29, 34, 36, 37, 41, 45, 48, 50, 52, 53, 58, 64, 72, 80, 82, 84, 88, 99, 102, 108, 109, 111, 112, 113, 115, 117, 118, 120, 126, 136, 137, 138, 139, 141, 143, 149, 151, 162, 167, 173, 175, 177, 182, 184, 186, 187, 190, 194, 195, 206, 215, 217, 230, 233, 236, 237, 259, 274, 275, 280, 283, 288, 296, 298, 302, 303, 309, 310, 315, 326, 328, 332, 333, 337, 338, 340, 342, 347, 349, 351, 353, 354, 357, 359, 362]

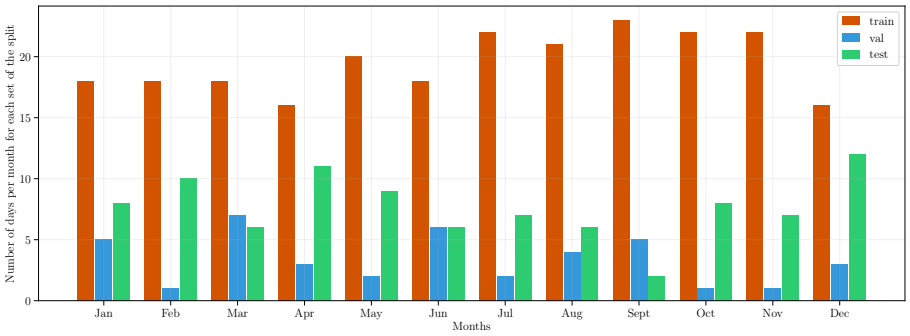


Figure A.2: Number of days for each split for each month of 2020

Appendix B

Technical information

This appendix contains theoretical complementary material for Chapter 3. In it, basic deep learning concepts are introduced, the U-Net variations are explained as well as the mathematics that allows GANs to learn.

B.1 Deep Learning Theoretical Background

This section introduces basic deep learning concepts that aim at helping the reader to better understand this manuscript.

Deep learning algorithms are based on models, also known as architectures. These models are defined by mathematical operations and are what learns in machine learning. The building blocks for these are called layers. These receive an input, operate over it, and output the results to the next layer until reaching the output of the network. For computer vision deep learning algorithms, the inputs are images that mutate from one layer to the next, extracting different features in each one. Examples of models are the convolutional neural networks (CNN) and recurrent neural networks (RNN). The first ones use the convolution operations over the images to extract information at different scales, from colors and edges to more complex structures like shapes and objects. These operations use a kernel that processes the images locally, meaning they have a receptive field. The RNNs on the other hand, are good for processing sequential data such as time series. These networks keep a state at the time step t using the inputs at the time step $t-1$. This way the state is updated recursively with each input, and thus the sequential data at the time step $t-k$ can influence the output at the time step t. Given that the state is constantly updated, if the sequence is too long, the elements from the beginning of the sequence lose importance with each recursion. Because of this is said that the RNNs suffer from "memory loss". To solve this problem, special kinds of cells are used: LSTM [62] y GRU [63]. These have mechanisms called gates that can control the flow of information, allowing longer sequences. Multiple other architectures, such as the ones presented, have been developed to solve problems in different domains.

To learn, deep learning models need a vast amount of data. These models have learnable parameters, called weights, that are tunned in the learning process, also known as training. These weights shape the way the network processes the data. In the training phase, the data is fed forward to a network until it reaches the output. Once this is done, the output is compared against a ground truth under some metric, known as the loss function. Making use of the chain rule, this error is then propagated backward in the network, calculating the gradients of the mathematical expressions

that define the layers, to minimize or maximize the loss function, thus enabling the network to learn from the errors. If the weights become too small, then the gradients are too small to update these, and the networks stop learning. This problem is known as vanishing gradient.

The data can be fed to the network in batches. Meaning in a forward pass a batch of images (in the case of computer vision) is used to calculate the loss and update the weights. The larger the batch size, the more memory is required since more computations have to be performed, and the fewer forward/backward iterations have to be done to complete an epoch (when the entire dataset is passed forward and backward in the network). If the model learns the data too well, meaning it manages to fit against the training data, then it is said that the model overfits. The problem with this is that the model picks up the noise and random fluctuations of a sample of the population of the data, making it generalize poorly. This is a well-known problem in deep learning and several techniques have been developed to avoid it.

The last concept which will be introduced is the concept of a hyperparameter. All deep learning algorithms have a set of these, that shape the ability to learn. These range from the rate at which the model learns (known as the learning rate), to the structure that defines a model, such as the number of layers, and the way the data is inputted into the network such as the batch size.

B.2 U-Net variations

Given the positive preliminary results obtained with the U-Net, further investigation about this architecture was conducted to find possible improvements. As mentioned before, the U-Net architecture inspired multiple versions of itself. In this case, three architectures are tested, that were originally presented in different papers about segmentation in medical images. [1]

Attention U-Net

This architecture was presented in the paper "Attention U-net: Learning Where to Look for the Pancreas" [46]. In it, the authors compare the proposed network against the original U-Net in a dataset of computed tomographies where the objective is to segment the pancreas [64].

One of the main contributions at the time was the implementation of a soft-attention technique in a feed-forward CNN model. Attention applied to CNN tries to detect the parts with the most relevant information of a given input to a layer, to give those regions the most importance when processed later in the convolutional layer. This process of detecting the most relevant regions is done in the "Attention Gates" (AG) shown in Figure B.1 and detailed in Figure B.2.

Two types of attention were considered for the architecture, Soft-Attention, and Hard-Attention. These differ in the way they assign relevance to the elements of the input. Where Soft-Attention allows the attention coefficient to be continuous values between zero and one, hard-attention only allows discrete values for the attention coefficients, zero or one, to keep the most relevant regions and ignore the rest. Soft-Attention was preferred over Hard-Attention as the latter one is often non-differentiable and relies on reinforcement learning for training, making the model training more unstable and difficult. Soft-Attention is differentiable so it is trained with backpropagation.

[1] The code for the networks mentioned in the next subsections was taken from this repository

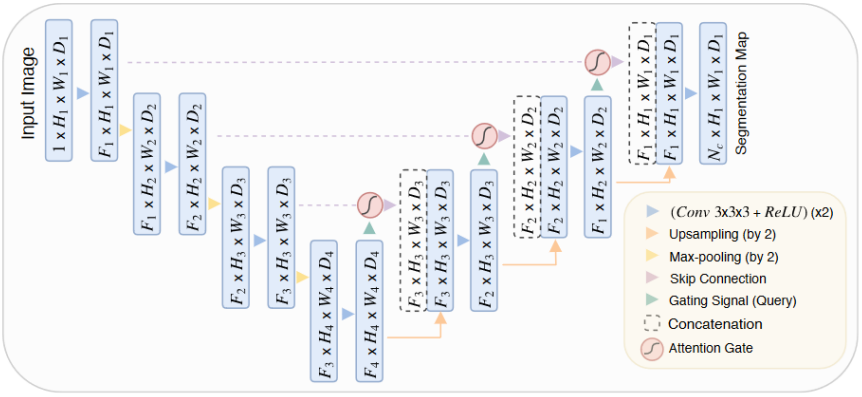


Figure B.1: A block diagram of the proposed Attention U-Net segmentation model. Attention gates (AGs) filter the features propagated through the skip connections. Schematic of the AGs is shown in Figure B.2. Feature selectivity in AGs is achieved by use of contextual information (gating) extracted in coarser scales. Figure reproduced from [46].

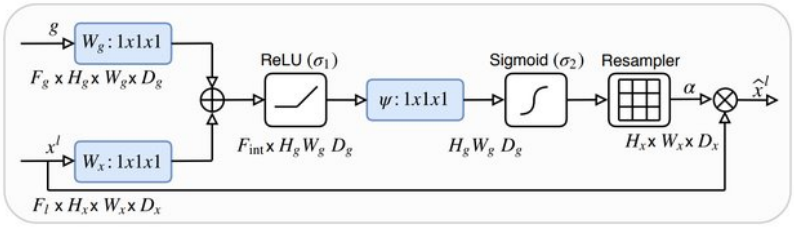


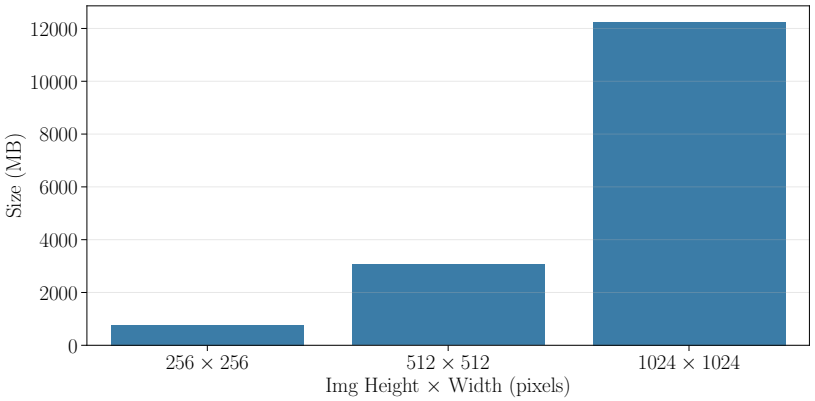
Figure B.2: Schematic of the proposed additive attention gate (AG). Input features (x^l) are scaled with attention coefficients (α) computed in AG. Spatial regions are selected by analysing both the activations and contextual information provided by the gating signal (g) which is collected from a coarser scale. Grid resampling of attention coefficients is done using trilinear interpolation. Figure reproduced from [46].

The attention gates have two inputs, the Gate Signal g comes from the output of the lower level in the Decoder and the input x^l from the skip connection. The input x^l gives the spatial information and g as it comes from deeper into the network has more feature information. To be able to add the two inputs, the x^l input has to be reduced in size by two as it comes from a higher level in the Encoder, so in its first convolution, the stride is 2×2. The attention coefficients ($\alpha \in [0, 1]$), show the importance of each value in the input x^l, so the gate outputs the pointwise multiplication of input feature-maps and the attention coefficients (Equation (B.1)).

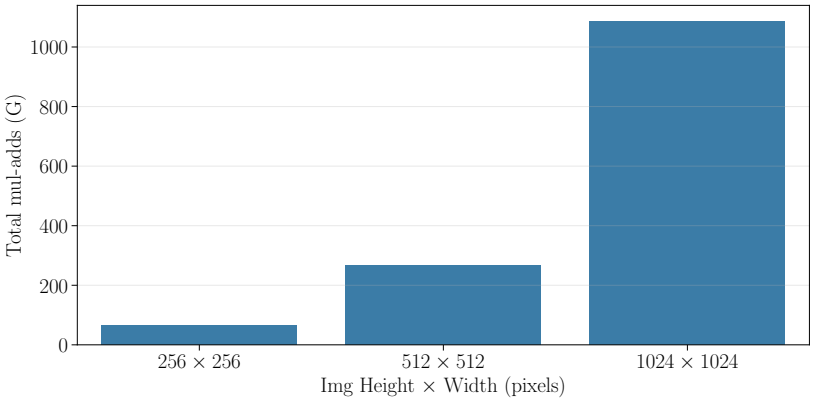
$$\hat{x}^l = x^l \cdot \alpha^l \tag{B.1}$$

As the authors explain about the implementation of the "Attention Gates" in the U-Net, these are incorporated to highlight features that are passed through the skip connections. The information extracted from the gate Signal is used to disambiguate irrelevant and noisy responses in skip connections. This is performed right before the concatenation operation to merge only relevant activations.

The results of the research show that by adding the attention gates, the U-Net benefits from that problem. The highest improvements were found when the organs



(a) Forward/backward propagation calculation space.



(b) Total ops.

Figure B.3: Attention U-Net stats as a function of the image size.

being segmented had a variable size (the case for the pancreas).

The implementation of the Attention U-Net in this work consisted of 34,878,573 trainable parameters, and needs a memory space of 139.51 MB (Figure B.3), while the largest U-Net considered had 17,262,977. It is relevant to note that increasing the number of parameters, makes the model more computationally expensive to train.

Nested U-Net

Another variation of the U-Net is the "Nested U-Net" (Figure B.4), also known as "UNet++" [45]. With this network, the authors try to improve the U-Net performance on four medical imaging datasets. They do this by making changes to the original architecture, one in the skip connections and the other in the way it calculates the loss. On the skip connections, the addition of convolutional layers (the green nodes in Figure B.4) with input from different levels in the Encoder, referred as dense skip connections, are expected to reduce the semantic gap between the feature maps and improve the gradient flow respectively. To calculate the loss, Deep supervision [65] is used, a technique that enables training deeper networks by adding supervision branches

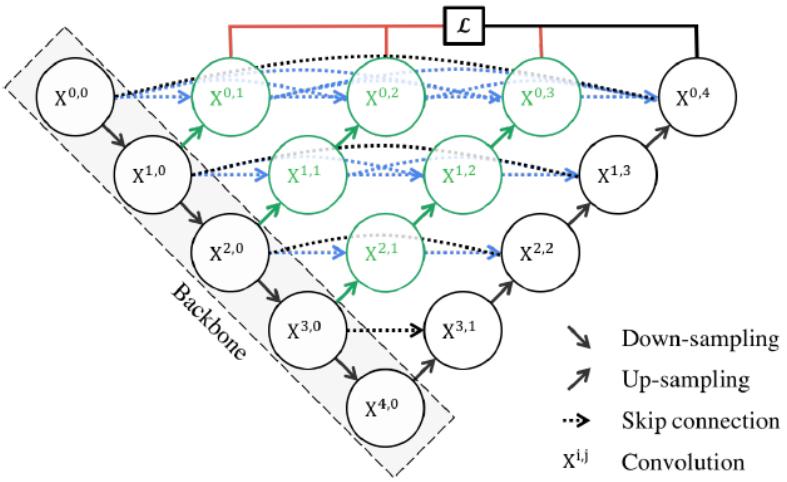


Figure B.4: UNet++ architecture consists of an encoder and decoder that are connected through a series of nested dense convolutional blocks. Figure taken from UNet++ [45].

in intermediate layers (red arrows in Figure B.4).

The network in [45] is compared against the U-Net in four medical imaging segmentation datasets. U-Net++ had a better performance in all four experiments. From those four, in two cases the model trained without Deep Supervision outperformed the one with DS, and in the other two the other way around.

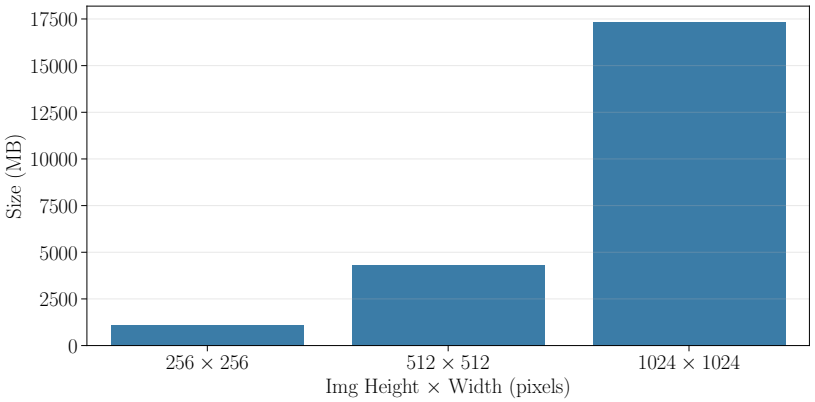
The model used in this project is trained without Deep Supervision, so only the last output is evaluated against the expected image. With 36,629,633 trainable parameters (Figure B.5) the architecture approximately the same number of parameters as the Attention U-Net, however the number of operations needed to generate a prediction almost double because of the dense skip connections. Also obtaining sufficient memory space to train the Nested U-Net with large images becomes an issue.

Recurrent Residual U-Net (R2U-Net)

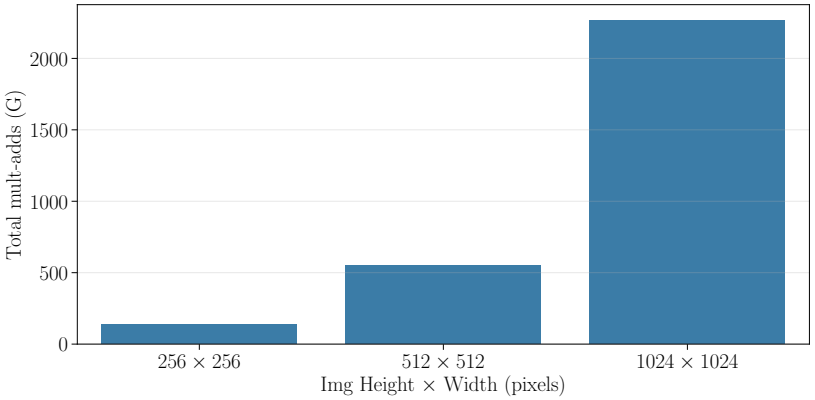
In [47] the authors create another version of the U-Net by adding residual and recurrent layers. The inspiration of using those techniques came from the appearance of residual connections to allow deeper networks [66], and recurrent layers applied to CNN [67]. Instead of the original convolutional layers in the encoder and decoder, the network has Recurrent Residual Convolutional Units (RRCU), shown in Figure B.6. As in the U-Net, the input to each level in the Encoder and Decoder is passed through two convolutional layers, in the RRCU the input is passed through recurrent convolutional layers, getting its recurrent aspect by passing each layer respective output again through it. For the residual part, the output of the second convolutional layer is added to the initial input.

Tested over different segmentation datasets, the R2U-Net shows an improvement over the original U-Net and other variations presented in the same article which only used Recurrent layers (RU-Net) or Residual layers (ResU-Net).

The implementation of the R2U-Net selected has 39,091,393 trainable parameters (Figure B.7), but due to its recurrent nature, the number of operations needed to make a prediction increases considerably against the previously mentioned models, almost tripling the number needed in Attention U-Net.



(a) Forward/backward propagation calculation space.



(b) Total ops.

Figure B.5: Nested U-Net stats as a function of the image size.

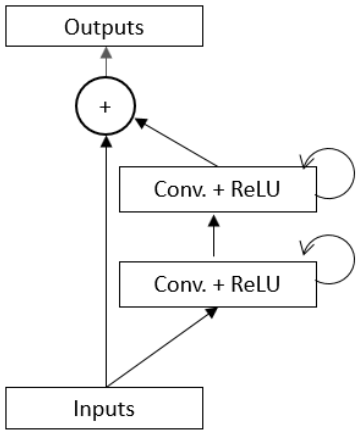
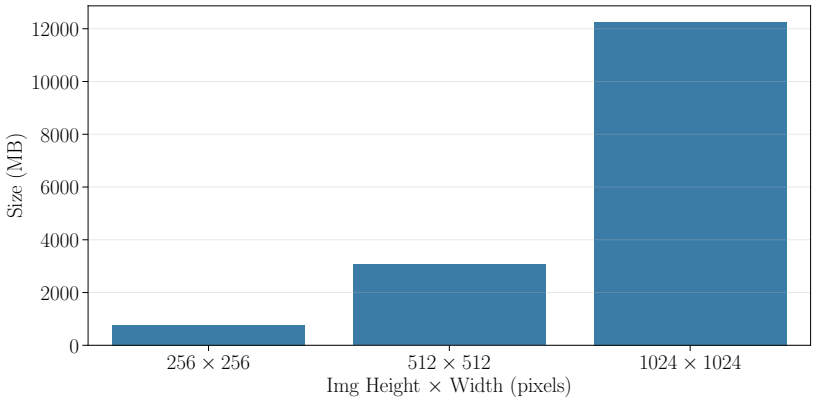
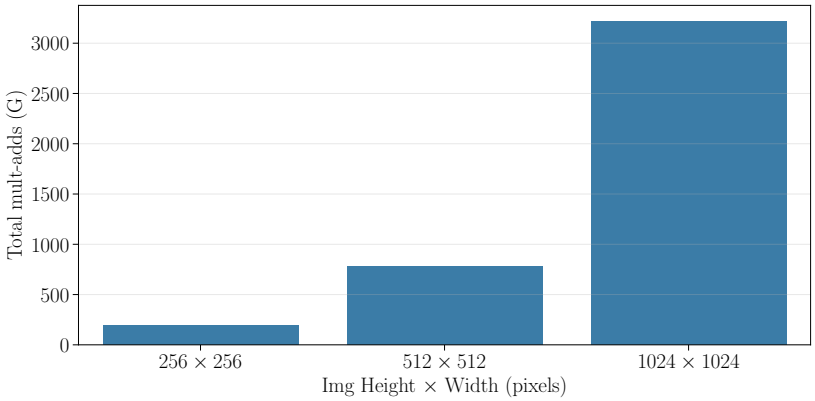


Figure B.6: Recurrent Residual Convolutional Unit (RRCU) diagram. Figure taken from R2U-Net's paper [47].



(a) Forward/backward propagation calculation space.



(b) Total ops.

Figure B.7: R2U-Net Stats.

B.3 The mathematics behind GANs

This section explains the mathematics behind the learning process for the GANs. The difference between the metrics used in the vanilla GAN and the Wasserstein GAN, and other theoretical definitions, needed to understand wGAN's training process, are explained as well.

Vanilla GAN

To understand how the generator and discriminator learn, let $x \sim P_r$ be a sample of the training set and $z \sim P_z$ a sample from the input noise. The Equation (B.2) shows the discriminator's loss function:

$$D_{loss} = log[D(x)] + log[1 - D(G(z))]. \tag{B.2}$$

Looking at Equation (B.2), since x comes from the training set, from the discriminator's point of view we want $D(x)$ to be one, meaning we want $log[D(x)]$ to be zero. In the second term, $G(z)$ is the output from the generator, which will be close to a

real image, so we want $D(G(z))$ to be zero, which means $log[1 - D(G(z))]$ equals zero. This means the discriminator aims at *maximizing* this expression.

The loss function for the generator is given by

$$G_{loss} = log[1 - D(G(z))]. \tag{B.3}$$

The objective of the generator is to deceive the discriminator into thinking $G(z)$ is a real image, meaning we want $D(G(z))$ to be 1 from the generator's point of view. This means the generator aims at *minimizing* the expression.

Therefore if P_r is the distribution of the real data, and P_g is the distribution of the images generated by the generator, the two expressions can be put together as follows:

$$\min_G \max_D L(G, D) = \mathbb{E}_{x \sim P_r} [logD(x)] + \mathbb{E}_{x \sim P_g} [log(1 - D(x))]. \tag{B.4}$$

The training process for G is to maximize the probability that D makes a mistake. This structure corresponds to a *minimax* game between two players.

In the expression B.4, the term $D(G(z))$ with $z \sim P_z$ from Equation (B.3) is simplified for $D(x)$ with $x \sim P_g$.

Wassertein GAN

The difference between the vanilla GAN and wGAN is in the metric that is used. To better understand how these differ it is necessary to delve into some probabilistic jargon. Two metrics used to quantify how different two probability distributions are, are the Kullback-Leibler (KL) divergence and the Jensen-Shannon (JS) divergence. The first quantifies how much a distribution p diverges from another distribution q, and is given by:

$$D_{KL}(p||q) = \int_x p(x) log \frac{p(x)}{q(x)} dx. \tag{B.5}$$

As it is clear from the expression, this metrics is not symmetric and becomes zero only if $p(x) = q(x)\ \forall x$. The JS divergence aims to quantify the same, but this is symmetric and smoother:

$$D_{JS}(p||q) = \frac{1}{2} D_{KL}(p||\frac{p+q}{2}) + \frac{1}{2} D_{KL}(q||\frac{p+q}{2}). \tag{B.6}$$

Operating on the expression B.4, it can be shown that the vanilla GAN's loss function quatifies the similarity between generated data distribution P_g, and the training data distribution P_r by JS divergence when D is optimal.

One of the reasons for mode collapse, a common failure mode for the vanilla GAN, is that the simultaneous SGD does not differentiate between minimizing and then maximizing versus maximizing and then minimizing the expression B.4. If the generator minimizes its loss function first (this means $D(G(z))$ is close to 1), then it is probable that the generator will map noise to fake images, only to images that the discriminator assigns values close to 1 (usually only a single mode of the target distribution) [2]. One way to solve this is to maximize first, but this may lead to vanishing gradients [49].

Following the previous naming convention, let P_r be the probability distribution from the real data and P_g the probability distribution corresponding to the generated images. Equation (B.7) is the Wasserstein distance, where $\Pi(P_r, P_g)$ are all the possible joint distributions between P_r and P_g, and $\gamma \sim \Pi(P_r, P_g)$ is one possible dirt moving plan. Then this expression results in the less expensive plan to go from one source probability distribution to a target one:

[2] https://courses.engr.illinois.edu/ece598pv/fa2017/Lecture12_GAN_AmirT.pdf

$$\mathbb{W}(P_r, P_q) = \inf_{\gamma \sim \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma} \|x - y\|. \tag{B.7}$$

The authors proposed a refactor for this formula to avoid the need to calculate all possible joint distributions using the Kantorovich-Rubinstein duality, leading to

$$\max_{||f||_L \leq 1} \mathbb{E}_{x\sim P_r}\left[f(x)\right] - \mathbb{E}_{y\sim P_g}\left[f(y)\right]. \tag{B.8}$$

This expression is to be maximized from the discriminator point of view (called the critic in the paper) since this means the distance between both distributions is greater than at the beginning. On the other hand, the generator aims at minimizing it.

The constraint $||f||_L \leq 1$ imposed on the discriminator is satisfied if f is 1-Lipschitz continuous. To achieve this, one can clip the values of f but this is not a good way to fulfill the constraint. If the clipping parameter is large, then the weights can take too long to reach their limit values. On the other hand, if we make the clipping parameter small, it can lead to vanishing gradients. This observation is presented in [68]. The authors propose a better way of training WGANs using gradient penalty, which imposes the aforementioned constraint. On top of this, the authors also used Adam [58] as optimizer instead of RMSProp [69] which was used in the original paper.

Appendix B. Technical information

Appendix C

Experimental results

This appendix includes the learning curves from experiments from Chapter 4. A study of the number of NaN values as a function of the prediction horizon for the CMV is presented. Several learning curves complementary to different experiments are also presented in this appendix. Finally, the more technical details for the GAN experiments are presented as well.

C.1 CMV - Number of NaNs as a function of the prediction horizon and window size

Given that the evaluation of the CMV can only be done on not-NaN pixels, and that an option for a fair evaluation of the deep learning models against the CMV is by using a window determined by the number of NaNs generated by the CMV, an analysis of these pixels as a function of the prediction horizon was conducted. For this, a variable window was defined and all pixels outside this window were excluded. A window of size 10 means 10 pixels for each side of the image are ignored, so in total 20 pixels horizontally for each row and 20 pixels vertically for each column. Figure C.1 shows the average percent of defined values (not-NaNs) in the predicted images depending on the size of the window, from 0 (no window) to 60 pixels in steps of 10 pixels between windows, for prediction horizons up to 2 hours. This analysis was performed on the URU validation dataset (images of size 512×512 pixels). As can be observed, with a window size of 50 pixels it is possible to maintain a 95% of valid values for up to 2 hours, which can be considered an acceptable amount. If we assume that this result can be extrapolated to longer prediction horizons then it can be said that for predictions of up to 5 hours a frame size of 125 pixels should be acceptable to minimize the amount of NaNs.

C.2 Training Metric

Figure C.2 shows the validation learning curves for U-Net models trained with MAE, MSE, SSIM, and MAE-SSIM as a loss function, and evaluated with MAE, MSE, and SSIM. The minimum obtained for each model is represented with a dot. It is possible to see that a model does better at the minimum point of the curve when evaluated with the same metric used for training.

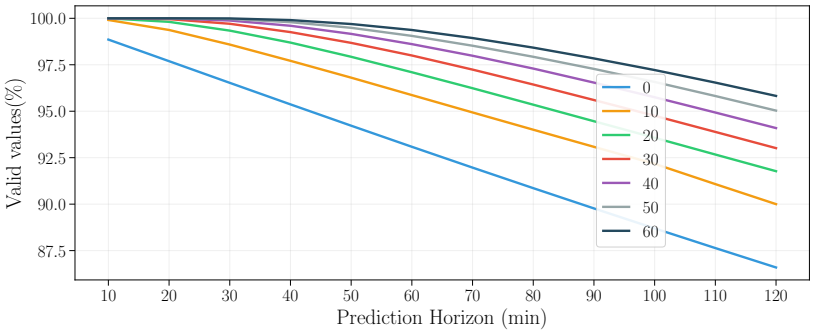


Figure C.1: Percent of valid values of the CMV prediction for different border sizes evaluated on the URU validation dataset.

C.3 U-Net variations learning curves

Figure C.3 shows the training and validation learning curves for the models with an hour prediction horizon. As the one for the ten minutes horizon, the takeaway from these curves is that all three variations showed issues with their capabilities to generalize to the validation dataset, as they have regular decreasing learning curves on the training dataset but the validation ones are noisy.

C.4 U-Net and U-Net Diff learning curves

This section presents the validation learning curves for the final models of the U-Net and U-Net Diff, both with 16 initial filters. The errors in each epoch are measured using MAE, the same metric used as training loss. The evaluation was done up to 5 hours ahead with hourly time steps. From Figure C.4, it is clear that both models have very similar behavior.

C.5 IrradianceNet learning curves

This section presents the train and validation learning curves for the IrradianceNet model. The errors in each epoch are measured using MSE. The evaluation was done up to 5 hours ahead with hourly time steps. Figure C.5 shows a fast improvement in the first epochs for both train and validation, and after a plateau with a slower improvement.

C.6 GANs

C.6.1 Grid-search

This section covers the technical details of the grid-search experiment. Table C.1 shows the values for the hyper-parameters that were taken into account for these experiments. The hyper-parameter *Lamda-GP* imposes the gradient penalty for the 1-Lipschitz continuity explained in Section 3.5.5. *with-S&S-on-train* refers to using the sunrise and sunset images in the training set. *Critic-iterations* is because in the original paper the authors trained the discriminator (critic) more. This hyper-parameter

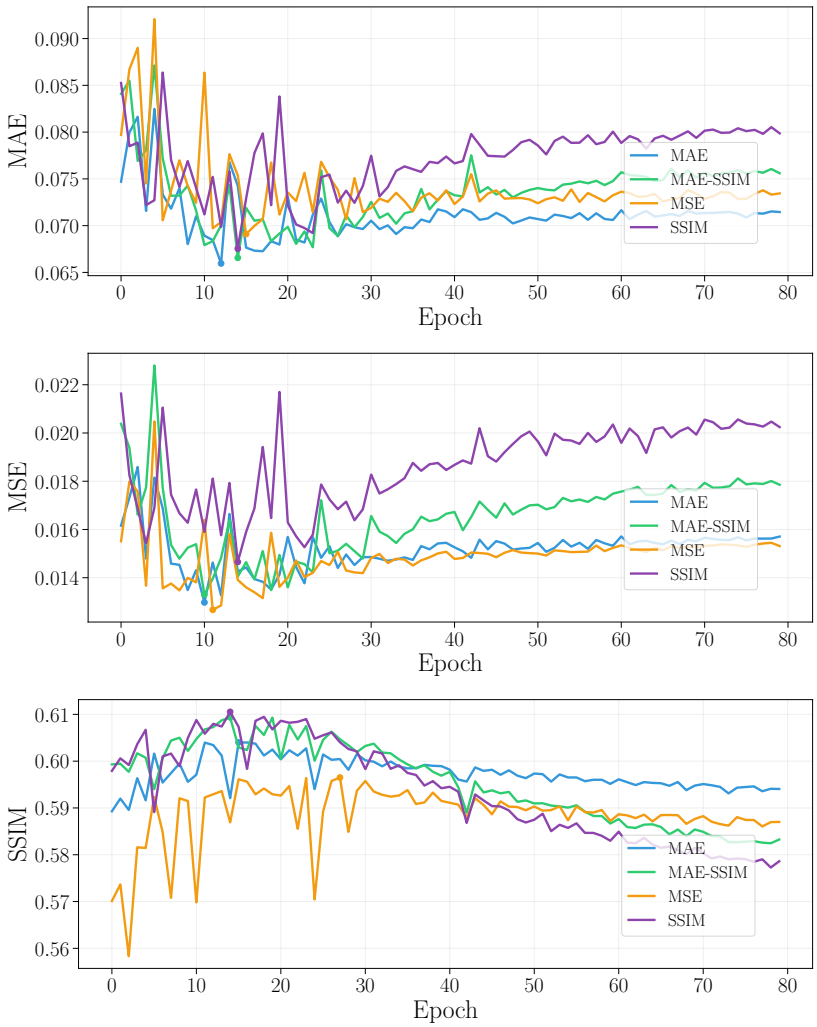
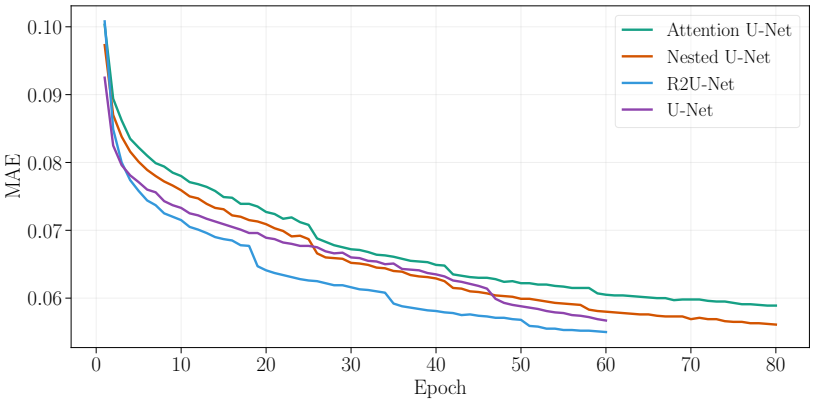
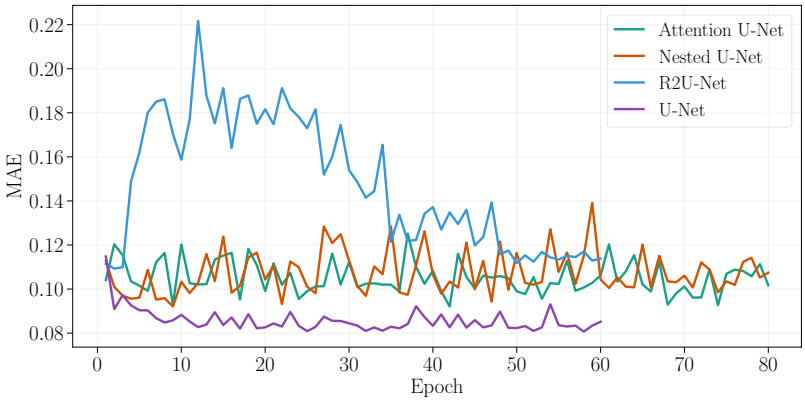


Figure C.2: Validation learning curves for U-Net models trained with MAE, MSE, SSIM, MAE-SSIM and evaluated with MAE, MSE, SSIM, for a sixty minutes prediction horizon on the MVD dataset.



(a) Train.



(b) Validation.

Figure C.3: Training and validation learning curves for the U-Net variations, for a sixty minutes prediction horizon on the URU dataset.

determines how many more iterations. Table C.2 shows the parameters for each model in these experiments. Figure C.6 and Figure C.7 show the wGAN loss and the RMSE validation loss for every model for every epoch of training, respectively. Lastly, Figure C.8 shows the 10 models that best minimized the wGAN loss in the grid-search. The hyperparameters for these can be found in Table C.3.

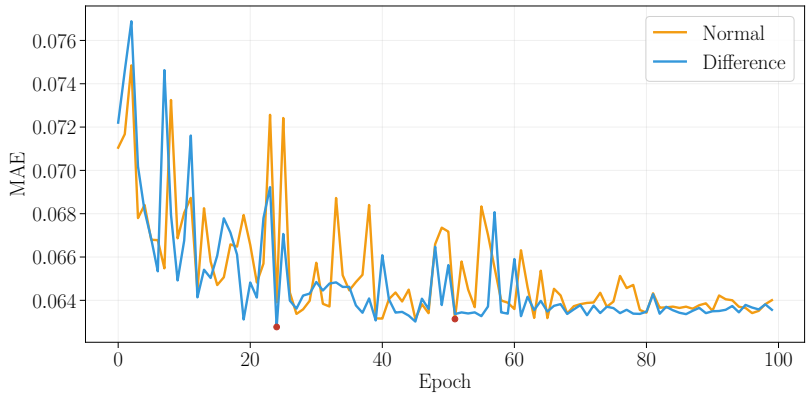
The Figure C.8 is the result of filtering these 10 models from Figure C.6.

C.6.2 wGAN models trained for the 60 minutes prediction horizon

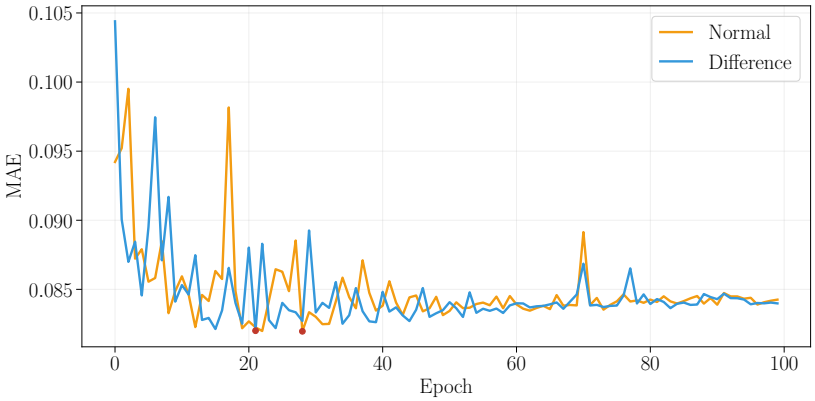
Figure C.9 and Figure C.10 show the wGAN loss and SSIM validation los for the 10 models on a prediction horizon of 60 minutes, for every epoch.

C.6.3 Training wGAN of 16 initial filters for 40 epochs

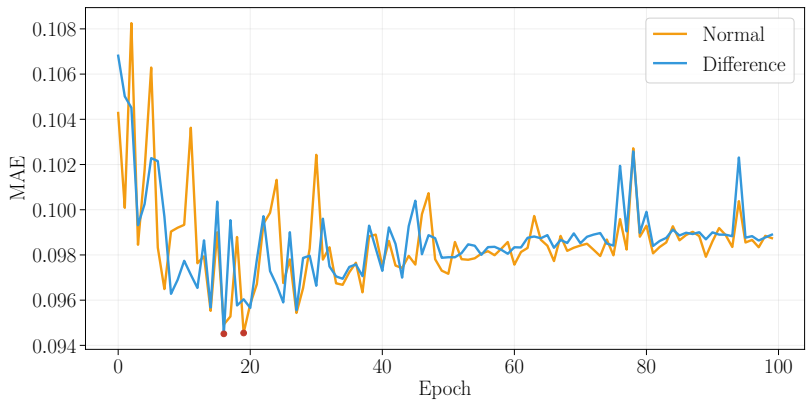
This section contains the training loss and SSIM validation loss for the training of the 10 best models for 40 epochs using a generator with 16 initial filters. Figure C.11



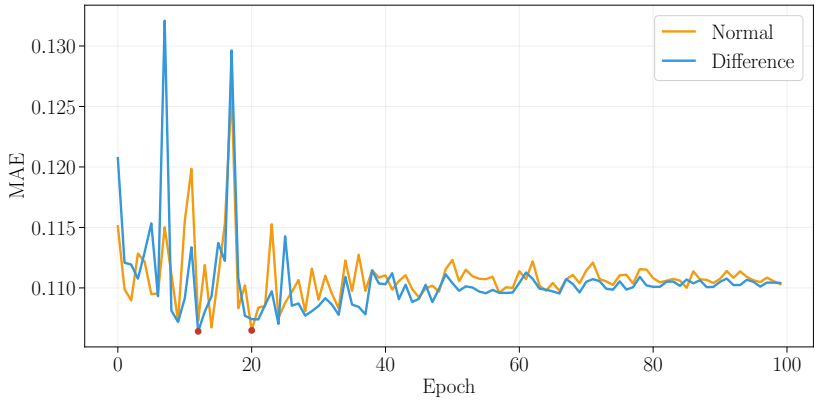
(a) MAE validation learning curve for one hour prediction horizon.



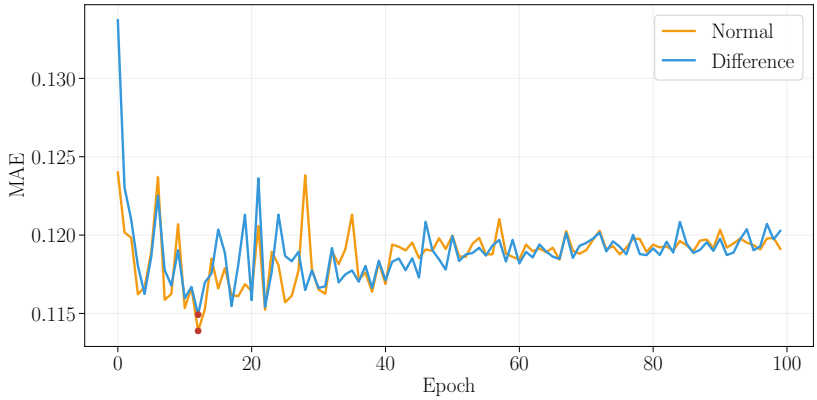
(b) MAE validation learning curve for two hours prediction horizon.



(c) MAE validation learning curve for three hours prediction horizon.



(d) MAE validation learning curve for four hours prediction horizon.

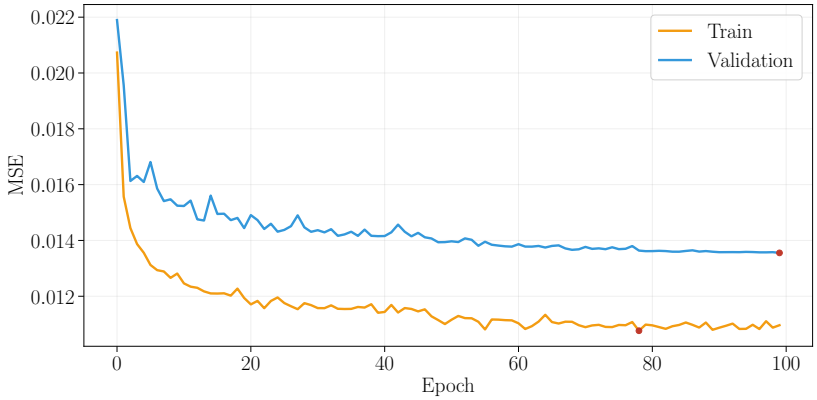


(e) MAE validation learning curve for five hours prediction horizon.

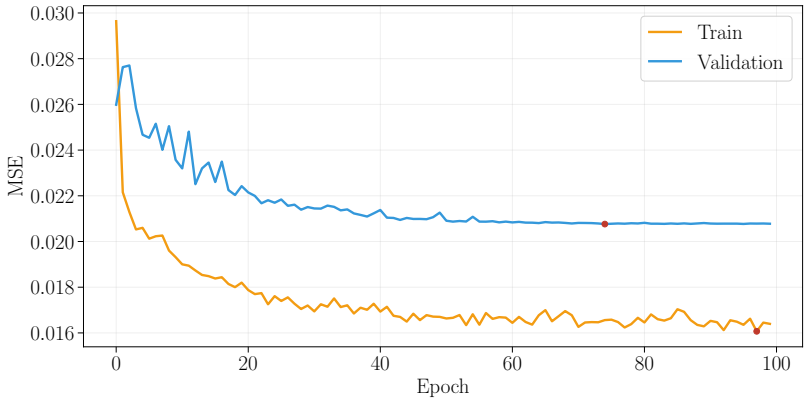
Figure C.4: Validation learning curves for the U-Net and the U-Net Diff. The evaluation was done up to 5 hours ahead with hourly time steps.

Hyper-parameters	**Values**			
Learning rate	1×10^{-4}	3×10^{-4}	9×10^{-4}	5×10^{-5}
Lambda-GP	5	10		
Features-D	16	32		
with-S&S-on-train	True	False		
Critic-iterations	0	5	10	

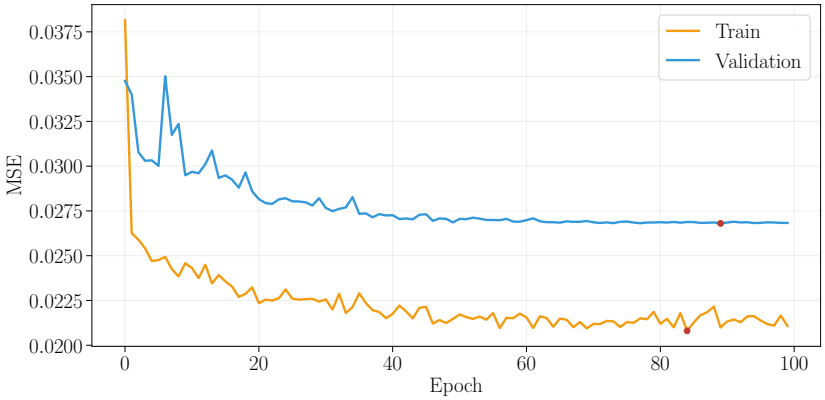
Table C.1: Grid Search hyper-parameters



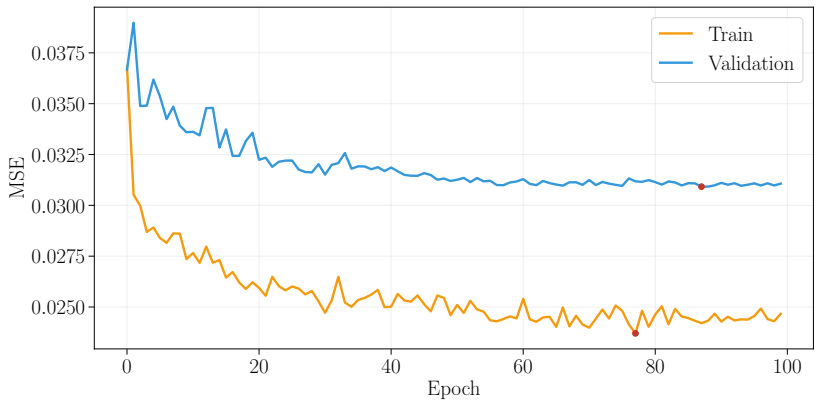
(a) MSE train and validation learning curve for one hour prediction horizon.



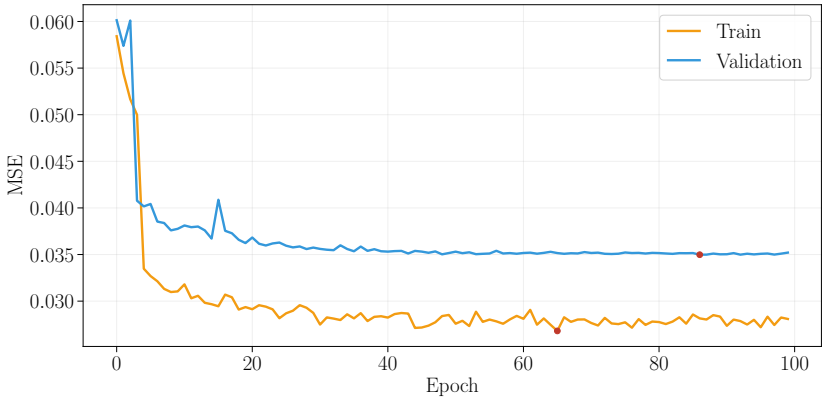
(b) MSE train and validation learning curve for two hours prediction horizon.



(c) MSE train and validation learning curve for three hours prediction horizon.



(d) MSE train and validation learning curve for four hours prediction horizon.



(e) MSE train and validation learning curve for five hours prediction horizon.

Figure C.5: Training and validation learning curves for the IrradianceNet. The evaluation was done up to 5 hours ahead with hourly time steps.

Parameters	**Values**
Optimizer	RMSProp
Starting weights	Best U-Net (64 filters) @ ph=10min over MVD
Trained for	15 epochs
Prediction horizon	10 min
Validation loss	RMSE

Table C.2: Grid-search training configuration.

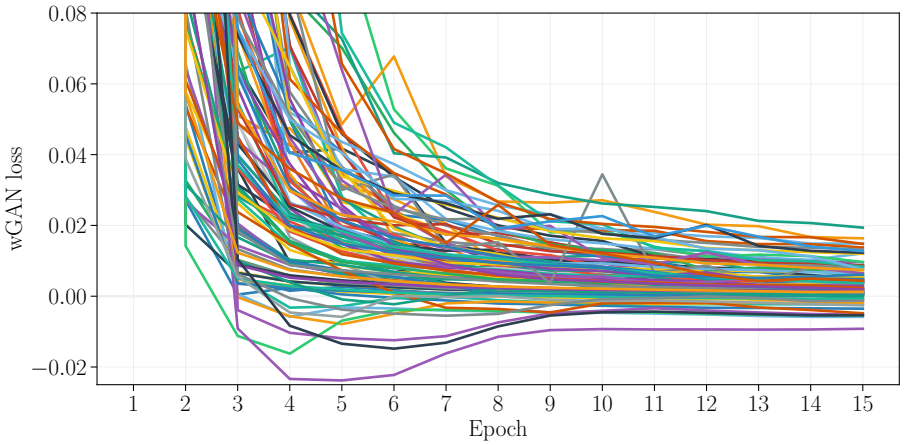


Figure C.6: wGAN loss for the grid-search.

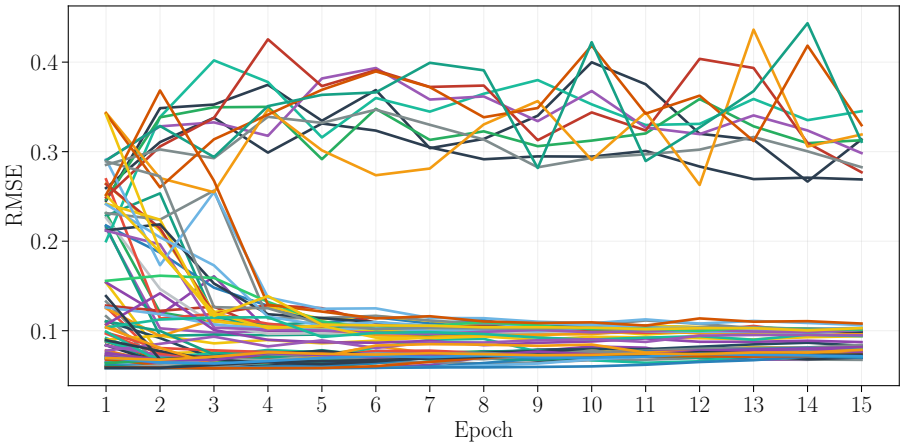


Figure C.7: RMSE loss on validation for the grid-search.

and Figure C.12 for validation shows that learning in a GAN scope does not mean an improvement in SSIM loss, and that around 10 epochs are enough for improving the image quality while maintaining the validation metrics equal or slightly worse as the U-Net.

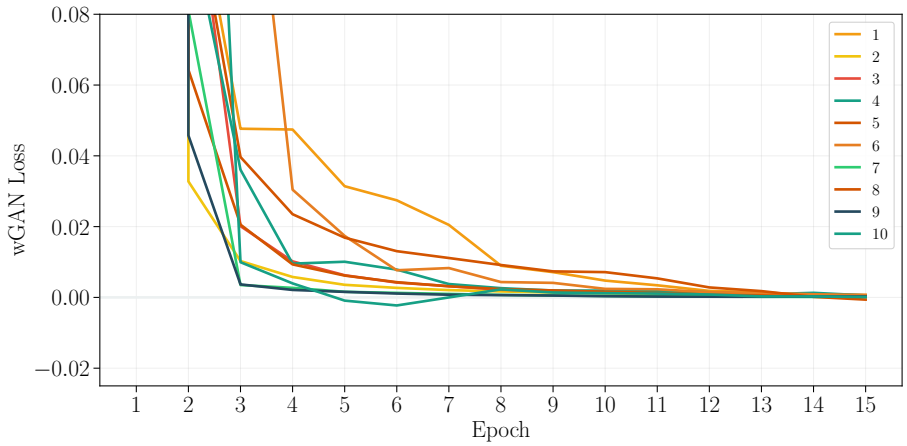


Figure C.8: wGAN loss for 10 best models according to wGAN loss. The numbers in the legends correspond to the hyper-parameters in Table C.3.

	Hyper-parameter				
	LR	λ_{GP}	$features_D$	with-S&S-on-train	Critic-iters
1	0.0001	5	32	False	5
2	0.0001	5	32	False	0
3	0.0003	5	32	False	5
4	0.0003	5	32	False	0
5	0.0003	5	16	False	10
6	0.0003	5	32	False	10
7	5e-05	5	16	False	0
8	5e-05	5	32	False	5
9	5e-05	5	16	False	0
10	5e-05	10	16	False	5

Table C.3: Hyper-parameters of the 10 best models of the grid-search according to wGAN loss.

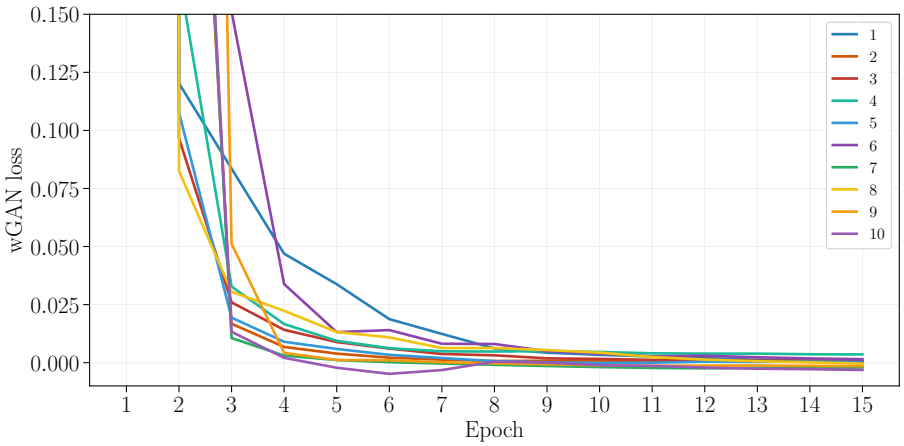


Figure C.9: Training learning curves using wGAN loss for every epoch for the 10 best models trained to predict 1 hour into the future.

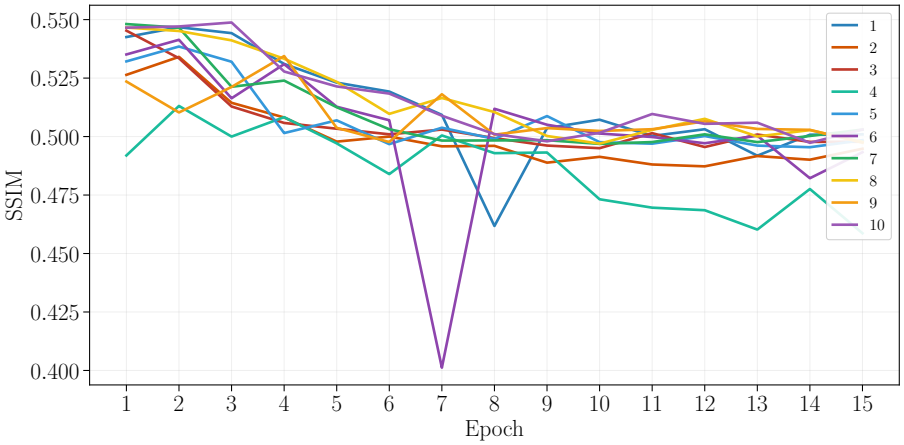


Figure C.10: Validation learning curves using SSIM loss for the 10 best models trained to predict 1 hour into the future.

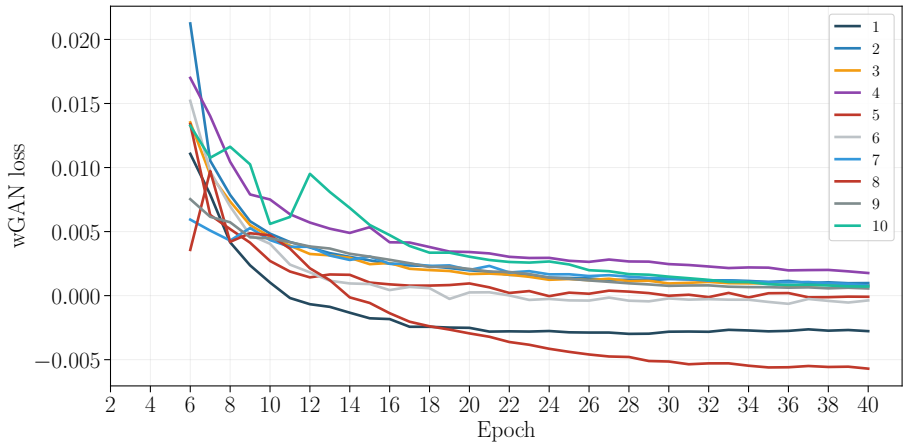


Figure C.11: Training learning curves using wGAN loss for every epoch for the 10 best models, using U-Nets with 16 filters trained for 40 epochs to predict 30 minutes into the future.

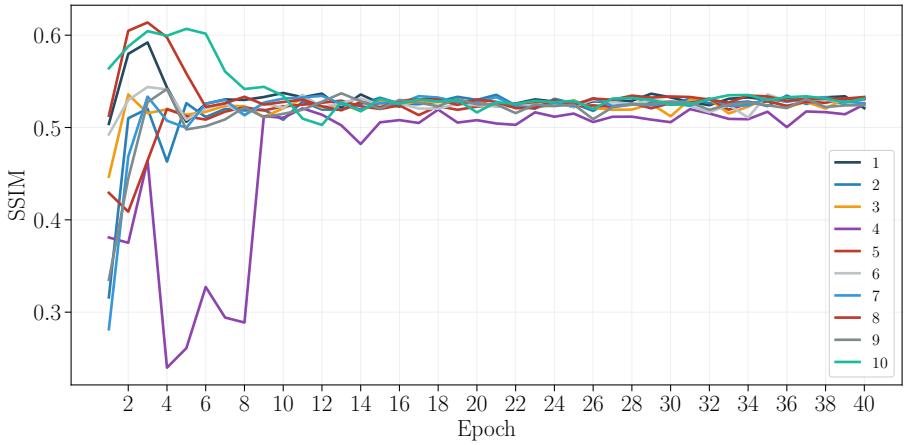


Figure C.12: Validation learning curves using SSIM loss on validation for every epoch for the 10 best models, using U-Nets with 16 filters trained to predict 30 minutes into the future.

Appendix D

Analysis of predictions

D.1 U-Net Diff vs CMV on REGION3

In this section, two examples of the predictions made by the U-Net Diff and the CMV algorithm are presented. The idea is to understand in which aspects the U-Net Diff outperforms the CMV and vice versa. The first example uses the input shown in Figure D.1 to generate the predictions which are shown in Figure D.2. At the center of the inputs, there is a cloud, that the U-Net Diff can predict its reduction and later disappearance. The CMV fails to predict this disappearance, keeping a stable cloud in the center. In the input, there are two clouds on the top border, primarily at the squares *1A* and *1C*. The U-Net Diff correctly detects the movement towards the center of the image from the first cloud and it extrapolates that motion by expanding the cloud. The second cloud, on the contrary, fails at predicting the movement and vanishes the cloud. The inputs for the last example are shown in Figure D.3 and the ground truth and outputs from the models are in Figure D.4. In this, the U-Net Diff accurately predicts the disappearance of the clouds on squares *1D*, *1E* and *4D*, while the CMV does not, keeping them constant in shape and position. Lastly, for this example, both models fail at predicting the movement to the left of the bottom-right cloud which can be seen at the five-hour horizon at the square *5C* where the ground truth has a portion of the cloud, while the models do not.

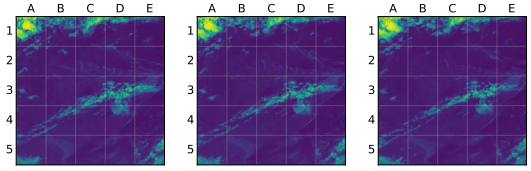
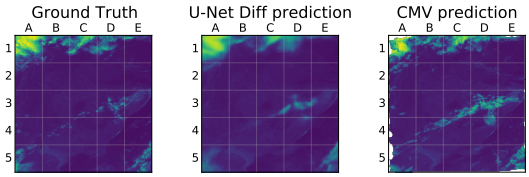
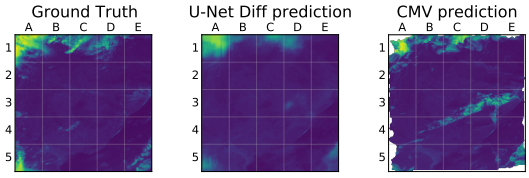


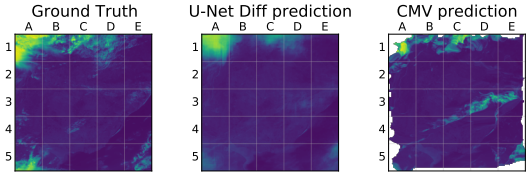
Figure D.1: Image sequence used as input for the U-Net Diff and CMV algorithm. Captured on January 13, 2020 at 12:00, 12:10, 12:20 (from left to right). Timezone UTC-0.



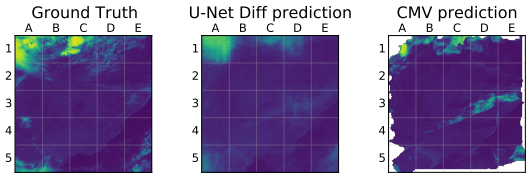
(a) 1 hour.



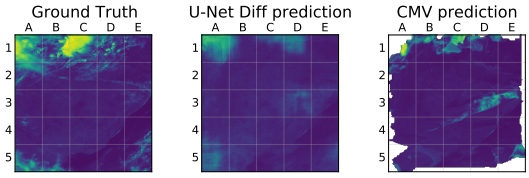
(b) 2 hours.



(c) 3 hours.



(d) 4 hours.



(e) 5 hours.

Figure D.2: Comparison between the ground truth, U-Net Diff prediction, and CMV prediction with inputs from Figure D.1. For the one to five hours prediction horizons.

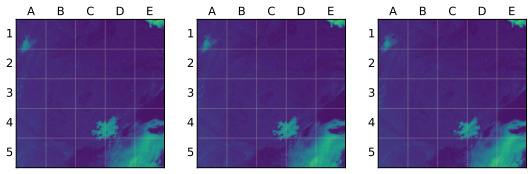
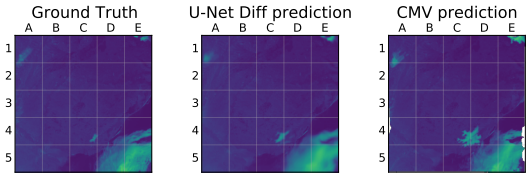
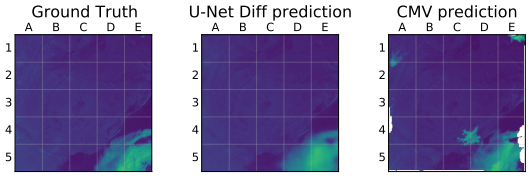


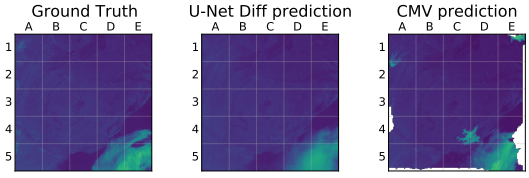
Figure D.3: Image sequence used as input for the U-Net Diff and CMV algorithm. Captured on August 4, 2020 at 14:20, 14:30, 14:40 (from left to right). Timezone UTC-0.



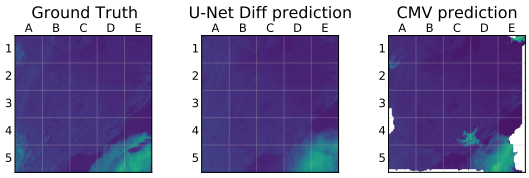
(a) 1 hour.



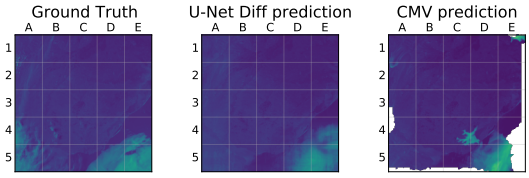
(b) 2 hours.



(c) 3 hours.



(d) 4 hours.



(e) 5 hours.

Figure D.4: Comparison between the ground truth, U-Net Diff prediction, and CMV prediction with inputs from Figure D.3. For the one to five hours prediction horizons.

Bibliography

[1] R. Perez and M. Perez, "A fundamental look at supply side energy reserves for the planet," *Int. Energy Agency SHC Program. Sol. Updat*, vol. 62, pp. 4–6, 2015.

[2] D. Arvizu, P. Balaya, L. Cabeza, T. Hollands, A. Jäger-Waldau, M. Kondo, C. Konseibo, V. Meleshko, W. Stein, Y. Tamaura, H. Xu, and R. Zilles, "Direct solar energy. In IPCC special report on renewable energy sources and climate change mitigation," 2011.

[3] International Energy Agency, "Renewable electricity growth is accelerating faster than ever worldwide, supporting the emergence of the new global energy economy 2021," Dec 2021.

[4] International Energy Agency, "Technology roadmap - solar photovoltaic energy," *Technology report*, 2014.

[5] Uruguayan Ministry of Industry, Energy and Mining, "Energía solar en Uruguay," *Programas de políticas y gestión.*, 2017.

[6] "Decreto n^o 173/010, autorizacion a suscritores conectados a la red de distribucion de baja tension a instalar generaciones de fuentes renovables," *Registro Nacional de Leyes y Decretos.*, vol. 2, p. 1103, 2010.

[7] R. Alonso-Suárez, *Estimación del recurso solar en Uruguay mediante imágenes satelitales*. PhD dissertation, Universidad de la República (Uruguay). Facultad de Ingeniería., 2017.

[8] P. Bauer, A. J. Thorpe, and G. Brunet, "The quiet revolution of numerical weather prediction," *Nature*, vol. 525, pp. 47–55, 2015.

[9] G. Giacosa, "Pronóstico de la energía solar a partir de imágenes satelitales," Master dissertation, Universidad de la República (Uruguay). Facultad de Ingeniería., 2020.

[10] D. Aicardi, P. Musé, and R. Alonso-Suárez, "A comparison of satellite cloud motion vectors techniques to forecast intra-day hourly solar global horizontal irradiation," *Solar Energy*, vol. 233, pp. 46–60, 2022.

[11] R. Alonso-Suárez, F. Marchesoni, L. Dovat, and A. Laguarda, "Satellite-based Operational Solar Irradiance Forecast for Uruguay's Solar Power Plants," in *2021 IEEE URUCON*, pp. 182–187, 2021.

[12] L. Berthomier, B. Pradel, and L. Perez, "Cloud cover nowcasting with deep learning," *CoRR*, vol. abs/2009.11577, 2020.

[13] C. K. Sønderby, L. Espeholt, J. Heek, M. Dehghani, A. Oliver, T. Salimans, S. Agrawal, J. Hickey, and N. Kalchbrenner, "MetNet: A neural weather model for precipitation forecasting," *CoRR*, vol. abs/2003.12140, 2020.

[14] T. Mitchell, "Elevation data in netCDF," Dec 2014.

[15] P. Salio, M. Nicolini, and E. J. Zipser, "Mesoscale convective systems over southeastern South America and their relationship with the south american low-level jet," *Monthly Weather Review*, vol. 135, no. 4, pp. 1290 – 1309, 2007.

[16] K. L. Rasmussen, M. D. Zuluaga, and R. A. Houze Jr., "Severe convection and lightning in subtropical South America," *Geophysical Research Letters*, vol. 41, no. 20, pp. 7359–7366, 2014.

[17] R. Alonso-Suárez, M. David, V. Branco, and P. Lauret, "Intra-day solar probabilistic forecasts including local short-term variability and satellite information," *Renewable Energy*, vol. 158, pp. 554–573, 2020.

[18] M. Bertalmío, A. L. Bertozzi, and G. Sapiro, "Navier-stokes, fluid dynamics, and image and video inpainting," in *2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001), with CD-ROM, 8-14 December 2001, Kauai, HI, USA*, pp. 355–362, IEEE Computer Society, 2001.

[19] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[20] J. Sola and J. Sevilla, "Importance of input data normalization for the application of neural networks to complex industrial problems," *IEEE Transactions on Nuclear Science*, vol. 44, no. 3, pp. 1464–1468, 1997.

[21] I. S. Nesmachnow S., "Cluster-uy: Collaborative scientific high performance computing in uruguay," *Torres M., Klapp J. (eds) Supercomputing. ISUM 2019. Communications in Computer and Information Science.*, vol. 1151, 2019.

[22] S. V. Ravuri, K. Lenc, M. Willson, D. Kangin, R. Lam, P. Mirowski, M. Fitzsimons, M. Athanassiadou, S. Kashem, S. Madge, R. Prudden, A. Mandhane, A. Clark, A. Brock, K. Simonyan, R. Hadsell, N. H. Robinson, E. Clancy, A. Arribas, and S. Mohamed, "Skillful precipitation nowcasting using deep generative models of radar," *CoRR*, vol. abs/2104.00954, 2021.

[23] M. S. Nazir, F. Alturise, S. Alshmrany, H. M. J. Nazir, M. Bilal, A. N. Abdalla, P. Sanjeevikumar, and Z. M. Ali, "Wind generation forecasting methods and proliferation of artificial neural network: A review of five years research trend," *Sustainability*, vol. 12, no. 9, 2020.

[24] A. McGovern, K. L. Elmore, D. J. Gagne, S. E. Haupt, C. D. Karstens, R. Lagerquist, T. Smith, and J. K. Williams, "Using artificial intelligence to improve real-time decision-making for high-impact weather," *Bulletin of the American Meteorological Society*, vol. 98, no. 10, pp. 2073 – 2090, 2017.

[25] J. Kuehnert, E. Lorenz, and D. Heinemann, *Satellite-Based Irradiance and Power Forecasting for the German Energy Market*, pp. 267–295. 08 2013.

[26] R. Perez, E. Lorenz, S. Pelland, M. Beauharnois, G. Van Knowe, K. Hemker, D. Heinemann, J. Remund, S. C. Müller, W. Traunmüller, G. Steinmauer, D. Pozo, J. A. Ruiz-Arias, V. Lara-Fanego, L. Ramirez-Santigosa, M. Gaston-Romero, and L. M. Pomares, "Comparison of numerical weather prediction solar irradiance forecasts in the US, Canada and Europe," *Solar Energy*, vol. 94, pp. 305–326, 2013.

[27] R. Perez, S. Kivalov, J. Schlemmer, K. Hemker, D. Renné, and T. E. Hoff, "Validation of short and medium term operational solar radiation forecasts in the US," *Solar Energy*, vol. 84, no. 12, pp. 2161–2172, 2010.

[28] E. Lorenz, A. Hammer, and D. Heinemann, "Short term forecasting of solar radiation based on satellite data," *EUROSUN2004 (ISES Europe Solar Congress)*, Jul 2004.

[29] D. Yang, S. Alessandrini, J. Antonanzas, F. Antonanzas-Torres, V. Badescu, H. G. Beyer, R. Blaga, J. Boland, J. M. Bright, C. F. Coimbra, M. David, Âzeddine Frimane, C. A. Gueymard, T. Hong, M. J. Kay, S. Killinger, J. Kleissl, P. Lauret, E. Lorenz, D. van der Meer, M. Paulescu, R. Perez, O. Perpiñán-Lamigueiro, I. M. Peters, G. Reikard, D. Renné, Y.-M. Saint-Drenan, Y. Shuai, R. Urraca, H. Verbois, F. Vignola, C. Voyant, and J. Zhang, "Verification of deterministic solar forecasts," *Solar Energy*, vol. 210, pp. 20–37, 2020. Special Issue on Grid Integration.

[30] D. Yang, "A guideline to solar forecasting research practice: Reproducible, operational, probabilistic or physically-based, ensemble, and skill (ropes)," *Journal of Renewable and Sustainable Energy*, vol. 11, p. 022701, 03 2019.

[31] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[32] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss functions for image restoration with neural networks," *IEEE Transactions on Computational Imaging*, vol. 3, no. 1, pp. 47–57, 2017.

[33] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," *CoRR*, vol. abs/1801.03924, 2018.

[34] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015.

[35] X. Su, T. Li, C. An, and G. Wang, "Prediction of short-time cloud motion using a deep-learning model," *Atmosphere*, vol. 11, no. 11, 2020.

[36] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," *CoRR*, vol. abs/1506.04214, 2015.

[37] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[38] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 1, pp. 185–203, 1981.

[39] A. H. Nielsen, A. Iosifidis, and H. Karstoft, "Cloudcast: A satellite-based dataset and baseline for forecasting clouds," *CoRR*, vol. abs/2007.07978, 2020.

[40] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," *CoRR*, vol. abs/1506.04214, 2015.

[41] A. H. Nielsen, A. Iosifidis, and H. Karstoft, "Irradiancenet: Spatiotemporal deep learning model for satellite-derived solar irradiance short-term forecasting," *Solar Energy*, vol. 228, pp. 659–669, 2021.

[42] P. Uwe, K. Steffen, Trentmann, Jörg, H. Rainer, F. Petra, K. Johannes, and W. Martin, "Surface radiation data set - Heliosat (SARAH) - edition 2.1," 2019.

[43] J. Sánchez Pérez, E. Meinhardt-Llopis, and G. Facciolo, "TV-L1 Optical Flow Estimation," *Image Processing On Line*, vol. 3, pp. 137–150, 2013. https://doi.org/10.5201/ipol.2013.26.

[44] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Image Analysis* (J. Bigun and T. Gustavsson, eds.), (Berlin, Heidelberg), pp. 363–370, Springer Berlin Heidelberg, 2003.

[45] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, "Unet++: Redesigning skip connections to exploit multiscale features in image segmentation," *IEEE transactions on medical imaging*, vol. 39, no. 6, pp. 1856–1867, 2019.

[46] O. Oktay, J. Schlemper, L. L. Folgoc, M. C. H. Lee, M. P. Heinrich, K. Misawa, K. Mori, S. G. McDonagh, N. Y. Hammerla, B. Kainz, B. Glocker, and D. Rueckert, "Attention u-net: Learning where to look for the pancreas," *CoRR*, vol. abs/1804.03999, 2018.

[47] M. Z. Alom, M. Hasan, C. Yakopcic, T. M. Taha, and V. K. Asari, "Recurrent residual convolutional neural network based on u-net (r2u-net) for medical image segmentation," *CoRR*, vol. abs/1802.06955, 2018.

[48] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[49] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," *arXiv preprint arXiv:1701.04862*, 2017.

[50] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *CoRR*, vol. abs/1511.06434, 2016.

[51] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR*, vol. abs/1411.1784, 2014.

[52] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CoRR*, vol. abs/1611.07004, 2016.

[53] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*, pp. 214–223, PMLR, 2017.

[54] S. Aigner and M. Körner, "Futuregan: Anticipating the future frames of video sequences using spatio-temporal 3d convolutions in progressively growing autoencoder gans," *CoRR*, vol. abs/1810.01325, 2018.

[55] Y.-H. Kwon and M.-G. Park, "Predicting future frames using retrospective cycle gan," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1811–1820, 2019.

[56] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, Curran Associates, Inc., 2019.

[57] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, JMLR Workshop and Conference Proceedings, 2010.

[58] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[59] G. Giacosa and R. Alonso-Suárez, "Desempeño de la persistencia para la predicción del recurso solar en uruguay," *Revista Brasileira de Energia Solar*, vol. 9, no. 2, pp. 107–116, 2018.

[60] "Data products: Cloud top height/cloud layer." https://www.goes-r.gov/products/baseline-cloud-top-height-cloud-layer.html. Accessed: 2022-03-28.

[61] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *CoRR*, vol. abs/1710.10196, 2017.

[62] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[63] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014.

[64] H. R. Roth, A. Farag, E. B. Turkbey, L. Lu, J. Liu, and R. M. Summers., "Data from pancreas-ct. the cancer imaging archive. (2016)." https://doi.org/10.7937/K9/TCIA.2016.tNB1kqBU.

[65] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," *arXiv preprint arXiv:1409.5185*, 2014.

[66] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[67] M. Liang and X. Hu, "Recurrent convolutional neural network for object recognition," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3367–3375, 2015.

[68] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," *CoRR*, vol. abs/1704.00028, 2017.

[69] T. Tieleman and G. Hinton, "Divide the gradient by a running average of its recent magnitude. COURSERA Neural Netw," *Mach. Learn*, vol. 6, pp. 26–31, 2012.

This is the last page.
Compiled on Wednesday 15th June, 2022.
http://iie.fing.edu.uy/