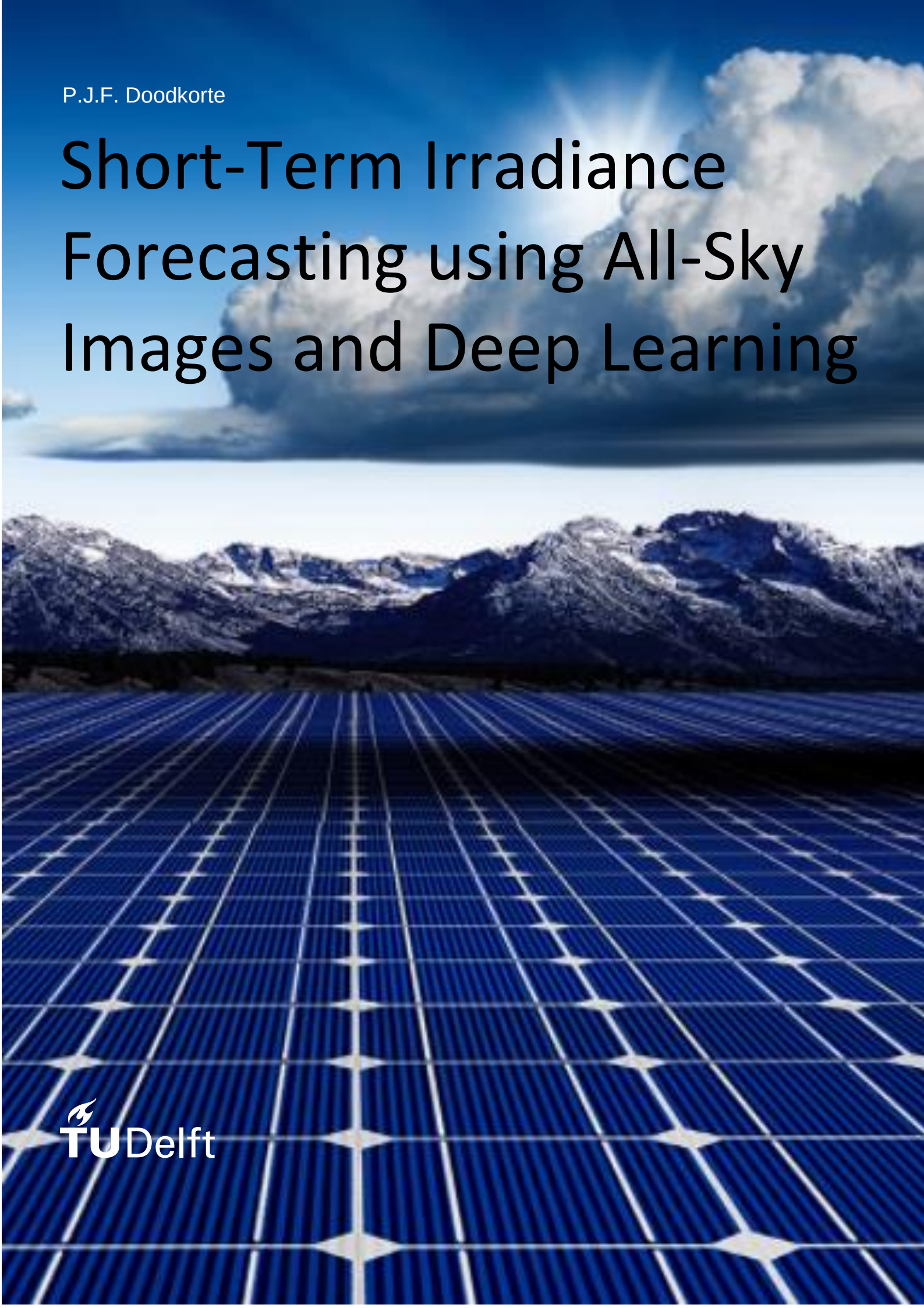
P.J.F. Doodkorte

Short-Term Irradiance Forecasting using All-Sky Images and Deep Learning



TU Delft

Short-Term Irradiance Forecasting using All-Sky Images and Deep Learning

By

P.J.F. Doodkorte

in partial fulfilment of the requirements for the degree of

Master of Science
in Sustainable Energy Technology

at the Delft University of Technology,
to be defended publicly on Wednesday June 18, 2021 at 13:00 AM.

Chairman:	Dr. ir. O. Isabella,	TU Delft
Thesis committee:	Dr. ir. H. Ziar,	TU Delft
	Dr. ir. J.L. Cremer	TU Delft
	Ir. A.M. Lopez,	TU Delft

TU Delft

PVMD
Delft University of Technology

Preface

Prior to the decision of the research topic, it was important that I was intrinsically interested in the research I was going to conduct to bring it to a good result. Within the Photovoltaic Materials and Devices group, several research topics grasped my interest and to obtain a good overview of where I was most interested in, I started studying literature around these topics. I found that my interests were in the field of solar irradiance forecasting using all-sky images. Through my literature study and meetings with my supervisors this resulted in the application of deep learning models to predict the irradiance using all-sky images.

After extensive literature research about methods to tackle the problem, much time spend coding in combination with instructive discussions with my supervisors I am happy to say that the research is finalized. The MSc thesis have tremendously increased my research skills and allowed me to get a better understanding of how deep learning can be applied to solve problems. On top of that I learned to work with large amounts and varieties of data.

Acknowledgements: First, I want to thank my supervisors Hesan Ziar and Arturo Martinez Lopez for guidance during the research. Special thanks go to my daily supervisor Arturo Martinez Lopez for his critical comments during our many meetings and helping me with thinking out of the box in improving the deep learning model. I also want to thank Hesan Ziar for the meetings and discussion that helped me to bring this research to a higher academic level. I also want to thank my MSc Thesis Committee Olindo Isabella, Hesan Ziar, Jochem Cremer and Arturo Martinez Lopez for assessing my work.

Furthermore, I would like to thank the Department of Mechanical and Aerospace Engineering and Centre for Energy Research University of California San Diego for making a comprehensive dataset publicly available which is used in my research.

Pim Doodkorte
11th of June, 2021

Abstract

Short-term solar forecasting is crucial for large scale implementation of solar energy and plays an important role in grid balancing, energy trading, and power plant operation. Cloud movement is the main source of unpredictability within solar forecasting and can be recorded using All-Sky Imagers. Conventional cloud modelling methods using image analysis techniques are unable to extract the spatial configuration and the temporal dynamics of clouds, resulting in poor predictions of the interaction with solar radiation. The goal of this study is to create a deep learning model for short-term irradiance forecasting between 0 and 21 minutes into the future using all sky images combined with auxiliary data. The model performance was assessed by comparing the deep learning model with the persistence model and showed that the deep learning model outperforms the persistence model with 24.8%. A sensitivity analysis to data usage is performed showing that besides using more data, also the variation of using multiple years of data results in better performance. Furthermore, the sensitivity of the model to input variables is assessed, showing that using the clear sky irradiance as input improves model performance with 16% and that meteorological data does not improve performance. Additionally, the model performance was evaluated during different sky conditions showing that the deep learning model outperforms the persistence model for all sky conditions, except overcast conditions. An example of the model behavior is extensively described, showing that the deep learning model tends to predict the trend of the irradiance fluctuations rather than the actual fluctuations. Next to that is in this study shown that the current deep learning model occasional miss important weather events, like obscuration of the Sun, resulting in large irradiance prediction errors. A pathway for future improvements for deep learning models to forecast the short-term irradiance is provided.

Contents

List of Figures

List of Tables

List of Abbreviations

Deep Learning Related	
AI	Artificial Intelligence
CNN	Convolutional Neural Network
DD	Dynamics Dataset
DL	Deep Learning
DNN	Dense Neural Network
LSTM	Long-Short Term Memory
ML	Machine Learning
ReLU	Rectified Linear Unit
ResNet	Residual Neural Network
SD	Static Dataset
Tanh	Hyperbolic Tangent

Error Metrics	
CI	Clearness Index
FS	Forecast Skill
MAE	Mean Average Error
MSE	Mean Squared Error
nMAE	Normalized Mean Average Error
nRMSE	Normalized Root Mean Squared Error
RMSE	Root Mean Squared Error
VI	Variability Index

Irradiance Forecasting	
AM	Air Mass
ASI	All-Sky Imager
CC	Cloud Cover
DHI	Diffuse Horizontal Irradiance
DNI	Direct Normal Irradiance
GHI	Global Horizontal Irradiance
I&P	Ineichen & Perez
NWP	Numerical Weather Prediction
RGB	Red, Green and Blue
TL	Linke Turbidity Coefficient

General	
NaN	Not a Number
NTP	Network Time Protocol
RAM	Random-Access Memory

1 Introduction

Reason of research: Significant technological improvements in photovoltaics combined with economies of scale gave rise to the exponential growth of solar energy. However, the Sun only shines during the day, making it an intermittent energy source. On top of that weather events like obscuration of the Sun by clouds cause significant fluctuations in solar energy yield. This makes solar energy a hard to predict energy source. To maintain grid stability, optimize energy control strategies and develop energy trading algorithms, it is crucial to know how much solar energy can be harvested from solar panels in advance.

An evident approach to forecast the solar energy output is to understand the dynamics of the clouds using satellite derived images or numerical weather predictions (NWP) (Haupt et al., 2017)(Marquez & Coimbra, 2013). These methods predict cloud movements with a large spatial and temporal scale. However, to maintain grid stability and improve short-term energy trading algorithms, a smaller spatial and temporal forecast resolution are required (Neuhoff et al., 2015). To achieve smaller temporal and spatial scale forecasts, ground-based sky camera image analysis have gained popularity for solar energy output forecasting (Haupt et al., 2017). Using multiple image sequences from 2 or more ground-based sky cameras make it possible to build a 3D configuration of the clouds (Marquez & Coimbra, 2013)(Haupt et al., 2017) and to predict the position of the shadows in future timesteps (Chow et al., 2011b). For an overview of ground-based sky cameras applications and research topics the reader is referred to (Kazantzidis et al., 2017). A major problem is that image processing techniques are unable to detect all different kinds of clouds and the cloud movements, resulting in occasional poor predictions (Haupt et al., 2017).

Machine Learning (ML) has been applied for solar energy power forecasting using information extracted from sky images (Al-Lahham et al., 2020). However, due to the limitations of the image processing techniques, the extracted information from sky images is occasionally of low quality. If the low-quality image extracted data is subsequently used in ML algorithms, the ML algorithm is limited by the image processing techniques used. A solution to this problem is deep learning (DL), which is a branch of ML. Within DL, the algorithm can learn what information has to be extracted from the image (Goodfellow, 2016) and can therefore overcome the limitation of image processing techniques. Current state-of-the-art models make use of DL and predict the irradiance or solar energy output using pyranometric measurements in combination with sky images (Paletta et al., 2021) (Kong et al., 2020).

The development of DL models for irradiance and solar power output predictions gained popularity in recent years, shown by the increasing amount and novelty of published papers in this field (Paletta et al., 2021)(Kong et al., 2020). These models make use of dense neural networks (DNN), convolutional neural networks (CNN) or a combination of CNNs and recurrent neural networks (RNN). These state-of-the-art models have shown promising results and outperform conventional short-term irradiance forecasting models, like image processing based models and the persistence model, known as a hard-to-beat model due to the fluctuant behavior of the irradiance during cloudy conditions (Pothineni et al., 2019). However, DL models are still unable to always correctly predict the dynamic behavior of the clouds in partly cloudy skies resulting in occasional significant errors during sudden irradiance drops or spikes when the Sun is obscured by clouds (Paletta et al., 2021). Predicting the interaction between solar irradiance and clouds in the sky remain a complex problem that is not tackled yet. To obtain more insight and a possible solution to this problem the following question must be answered: What is the influence of important factors like input variables, the used dataset, forecast horizon and sky conditions on the performance of DL short-term irradiance forecasting models using all sky images?

Aim of the research: The goal of this research is to answer the question by creating a short-term 21-minute ahead irradiance forecasting model using sky images and auxiliary data. The auxiliary data comprises meteorological measurements, irradiance measurements, and clear-sky irradiance model outputs. The methodology used for short-term irradiance forecasting is deep learning and the influence of input variables, different datasets, forecast horizons and sky conditions on the model its prediction error are evaluated.

In this study a DL model for short-term irradiance forecasting is developed, using significantly more data compared to preceding studies involving DL with sky images as input. The data used in this study is made publicly available by (Carreira Pedro et al., 2019) and is combined with the modelled clear sky irradiance. The original contributions of this study are:

- The development of a DL model for short-term irradiance forecasting using sky images, pyranometric measurements, meteorological data and modelled clear sky irradiances using a significantly larger dataset as preceding studies.
- This study performs an ablation study to evaluate the effect of single modalities within the multi-modal dataset on the predictive performance of the DL model.
- Demonstrate how the predictive performance of the DL model is affected of the by the size of the dataset used.
- Show that training DL models on a dataset comprised of multiple years outperform models trained on a dataset of a single year.
- Sky conditions were classified using the clearness index and variability index to evaluate the performance of the model during different sky conditions.
- Demonstrate that meteorological data does not improve the performance of DL irradiance forecasting with current DL architectures and input data.
- Show that using the clear sky irradiance as an input improves the DL model predictive performance for short-term irradiance forecasting.
- Demonstrate that the DL model does not aim to predict irradiance spikes or valleys but tries to minimize the prediction error by fitting the trend of the irradiance fluctuations.

Structure of the research:
The structure of this study is as the following. In Chapter 2 is the reader informed about background information to understand the application of deep learning methods for short-term irradiance forecasting. In Chapter 3 the methodology used to train the deep learning models, performance evaluation and sensitivity studies are explained. Chapter 4 depicts the deep learning model performance and sensitivity results. Subsequently is in Chapter 5 the research discussed and in Chapter 6 a conclusion of this study given.

2 Background

2.1 Meteorology

Meteorology is concerned with the dynamics of the atmosphere and what factors are of influence within the atmosphere (Mölders & Kramm, 2014). The atmosphere can be seen as a thick skin around the earth that contains the gasses that people breathe in, consisting 21% of oxygen, 78% of nitrogen, and 1% of other gasses, like methane and carbon dioxide (Spellman, 2013). The atmosphere can be divided into several spheres: troposphere, stratosphere, mesosphere, thermosphere, ionosphere, and exosphere ranging between 0- and 10,000-kilometres height. In the above-mentioned spheres interaction with incoming solar irradiance occurs, depending on the amount and type of gasses within that sphere. The exchange in most spheres is relatively constant and minimal, except for the troposphere. The troposphere ranges from 0 to 14.5 kilometres in height and has the highest air density, and is the sphere in which almost all weather events occur (NASA, 2017). Therefore, in this study, this sphere will be considered as the atmosphere (Mölders & Kramm, 2014).

2.1.1 Meteorological Variables

The weather consists of 9 main essential elements that describe the atmospheric state, called meteorological variables, consisting of: air pressure, wind, air temperature, relative humidity, solar energy, evaporation, precipitation, and clouds (Spellman, 2013). A simplified summary about the interaction between the atmospheric elements is given below. For a more comprehensive view of the interaction, it is advised to look into the book of (Spellman, 2013).

Solar energy heats the earth, and the air gets heated up mainly by the earth and partly by the Sun directly. Due to the heating of the air, the air temperature increases, and the volume expands, resulting in an upward flow. This upward flow causes a negative **pressure** difference underneath the elevated air, and air will get sucked towards the lower pressure area from the surroundings. This sucked air is better known as the **wind**. Besides the air and earth surface, the water bodies get heated up, and water **evaporates** into the air. The amount of water that can be present in the air as water vapor is mainly dependent on the temperature and can be expressed with the **relative humidity.** At 100% relative humidity, water vapor condenses and attaches to microscopic particles within the air. The water condensed air particles group and are heated at the earth its surface, which creates an upward draft and keeps the water condensed particles stay airborne: the airborne particles group and form **clouds**. Within the clouds, water droplets collide and grow in size until they reach a size at which the upward draft is not enough to keep the droplets airborne, and **precipitation** takes place (Spellman, 2013).

2.1.2 Clouds

Clouds can be characterized using their taxonomy that is based on their structure and their altitude. The main varieties of clouds described are:

- Stratus: Flat clouds with no distinguishable structure
- Cumulus: Puffy Clouds
- Nimbus: Rain bearing clouds

High clouds (5-13km height) typically contain ice and have the name ‘Cirro-‘. A cirrostratus cloud is a high flat cloud containing ice crystals. Midlevel clouds (2-7km height) have the name ‘Alto-‘ and the lowest clouds (0-2km height) have no prefix. In Figure 1, the classification of clouds and the corresponding heights and structures are depicted. In Figure 1 is shown that clouds can occur in many forms, where each cloud type behaves differently. Besides, multiple types of clouds may co-occur, adding more complexity to the mix of clouds (Mölders & Kramm, 2014).

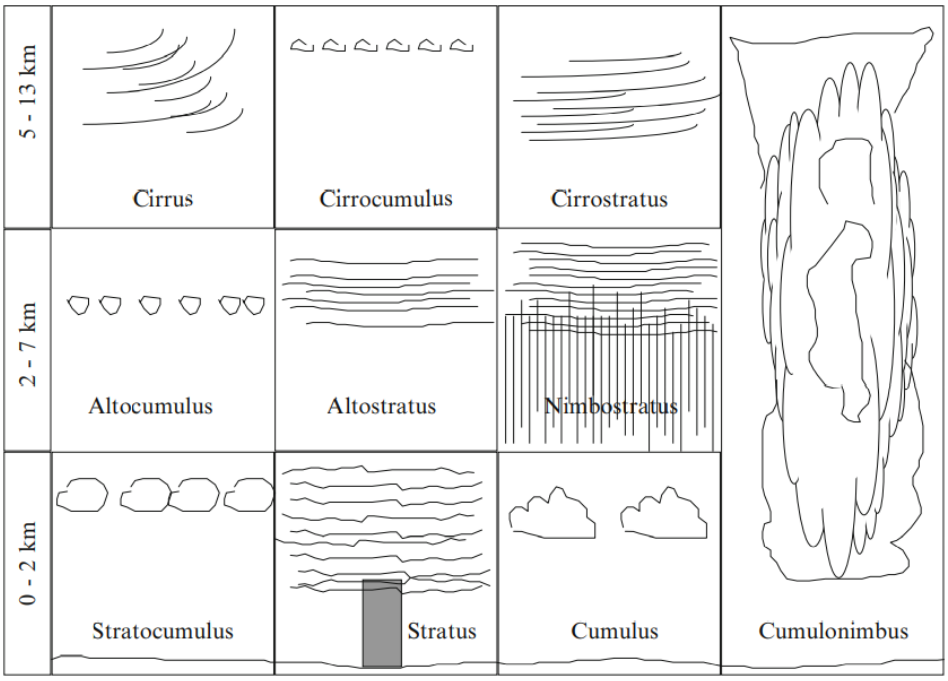


Figure 1: The clouds classification depending on the height within the troposphere and the structure of the clouds is shown. Obtained from (Mölders & Kramm, 2014)**.**

More information about clouds and their dynamics can be found in the book (Mölders & Kramm, 2014). To fully understand how clouds interact with solar irradiance, it is essential to investigate the solar irradiance first.

2.2 Solar Irradiance

Solar radiation is a bundle of waves with different wavelengths, better known as electromagnetic waves forming a light spectrum. When the solar radiation does not traverse through the earth its atmosphere, the light spectrum is called the Air Mass (AM) 0 spectrum, at which the solar irradiance is 1361 W/m^2, which is called the extraterrestrial normal incident irradiance (I_0). Depending on the location of the earth relative to the Sun, a certain distance through the atmosphere must be traversed. The location of the Sun can be expressed with the solar altitude (a_s) and solar azimuth (A_s). A visualization of how a_s and A_s are used to determine the solar position is depicted in Figure 2. When the solar radiation traverses through the atmosphere, it encounters gasses that interact with the electromagnetic waves and reflect, scatter, absorb or transmit the solar radiation, resulting in dissipation of solar radiation (Smets et al., 2016).

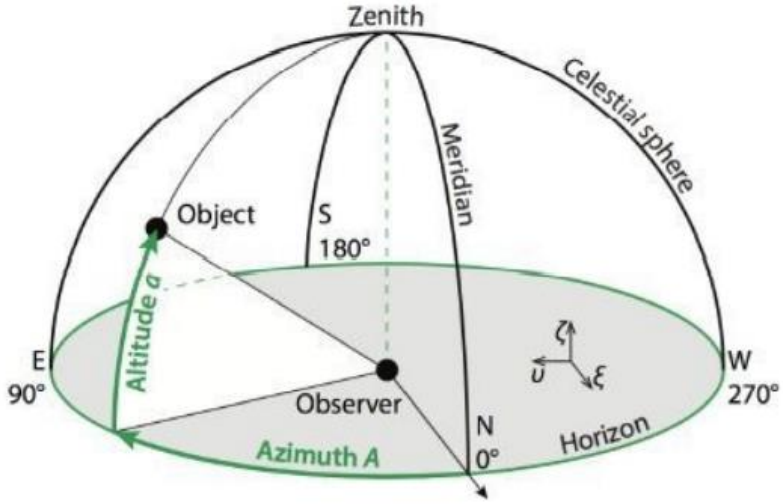


Figure 2: The horizontal coordinate system used for determining the Sun position relative to the observer on the earth is shown. Obtained from (Smets et al., 2016)**.**

Solar radiation travels from the Sun to the earth and experiences minimum interaction outside of the atmosphere. Within the atmosphere, the dissipation of solar energy is mainly due to gasses, aerosols, and clouds. The amount of scattering, reflection, absorption, and transmission of the solar radiation depends on which gasses, aerosols, and clouds are traversed. Besides, the atmosphere is constantly fluctuating due to seasons, weather phenomena, and other irregularities which also impact the solar radiation and atmosphere interactions (Smets et al., 2016). The solar radiation reflections due to atmospheric interactions result in different forms of irradiance. The three main types of irradiances considered in this study are depicted below and depicted in Figure 3.

- Direct normal irradiance (DNI): It is the irradiance at the surface of the earth perpendicular to the Sun.
- Diffuse horizontal irradiance (DHI): It is the irradiance measured at the surface of the earth from radiation scattered within the atmosphere. It includes all incoming radiation, except radiation coming directly from the Sun.
- Global horizontal irradiance (GHI): It is the radiation from the Sun coming at a horizontal surface of the earth. The GHI can be calculated from the DNI and DHI using Equation 2-1.

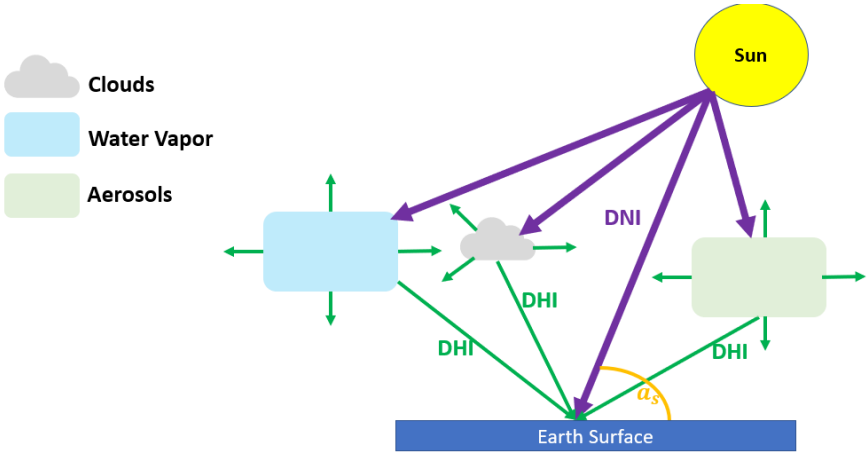


Figure 3: The DNI, DHI are depicted. The GHI can be calculated from the DNI and DHI using Equation 2-1.

$$GHI = DNI * \sin(a_s) + DHI \qquad \textbf{2-1}$$

The amount of direct irradiance and diffuse irradiance is dependent on the atmospheric state and to the relative location of the Sun to the earth. The relative location of the Sun to the earth changes constantly and is dependent on the time and day of a year. Since the trajectory of the Sun is well understood, the relative position of the Sun regarding a location on the earth is also well understood. Two main distinctions between atmospheric states can be distinguished, namely clear sky conditions and cloudy sky conditions. During clear sky conditions, the solar radiation traverses through the atmosphere without encountering clouds, while during cloudy-sky conditions, the solar radiation traverses through the atmosphere while encountering clouds (Smets et al., 2016).

2.2.1 Clear Sky Irradiance

The clear sky irradiance is the amount of irradiance on the surface of the earth in a cloudless atmosphere. Since the solar position relative to the earth and the composition of the atmosphere are well understood, it is possible to approximate the irradiance during a cloudless day, using clear sky models. A large variety of different clear sky models exists, where each model uses different parameters as input. The different inputs can be categorized as solar geometry inputs and atmospheric parameter inputs (Antonanzas-Torres, 2019). Furthermore, clear sky models can be distinguished in empirical models and physical models. In this study an empirical model is used, called the Ineichen & Perez (I&P) model (Ineichen & Perez, 2002). The I&P model includes solar geometry inputs and basic parameters of the atmospheric state, like air pressure, temperature, relative humidity, aerosol content and Rayleigh scattering (Reno et al., 2012).

The solar geometry inputs are mainly used to calculate the distance from the Sun to the earth. When the Sun is directly overhead the solar radiation travels the least distance through the atmosphere and is called the AM1 spectrum. Depending on the distance traversed through the atmosphere the AM is determined. The distance is a product of the angle of the Sun relative to the earth and a simplified approximation for the AM is given by Equation 2-2 (Smets et al., 2016). Multiple other AM approximations are available and will not be discussed in this study, but can be found in the paper of (Rapp-Arrarás & Domingo-Santos, 2011).

$$AM := \frac{1}{\sin(a_s)} \tag{2-2}$$

The I&P model is an addition to the Kasten model (Ineichen & Perez, 2002). The Kasten model uses two coefficients f_{h1} and f_{h2} which relate altitude of the observer with the altitude of the atmospheric interactions. The Equations of f_{h1} and f_{h2} are depicted in Equations 2-3 and 2-4 respectively. In the I&P model, data from clear sky conditions datasets are used to create two additional altitude dependent coefficients c_{g1} and c_{g2}, depicted in Equations 2-5 and 2-6 respectively.

$$f_{h1} = exp(-h / 8000) \tag{2-3}$$

$$f_{h2} = exp(-h / 1250) \tag{2-4}$$

$$c_{g1} = 5.09 * 10^{-5} * h + 0.868 \tag{2-5}$$

$$c_{g2} = 3.92 * 10^{-5} * h + 0.0387 \tag{2-6}$$

Besides the additional coefficients, also the Linke turbidity (TL) coefficient has been used. The TL characterizes the degree of transparency of the atmosphere. Based on older TL coefficients, modification have been performed using data of different locations, altitude, and climates. The I&P model found a turbidity coefficient that is independent of solar elevation and can be found in (Ineichen & Perez, 2002). Using the TL, Kasten coefficient, I&P coefficients, I_0, AM and a_s a clear sky model for the GHI has been formulated as depicted in Equation 2-7. Clear sky models for the DNI and DHI are not used in this study and will therefore not be discussed. Formulations of the clear sky DNI and DHI is elaborated on within (Antonanzas-Torres, 2019). More information about the I&P model can be obtained in the paper of (Ineichen & Perez, 2002).

$$GHI^{Clear\ Sky} = c_{g1} * I_0 * \sin(a_s) * \exp\left(-c_{g2} * AM * \left(f_{h1} + f_{h2}(TL - 1)\right)\right) * \exp(0.01 * AM^{1.8}) \tag{2-7}$$

Modelling the expected GHI during clear sky conditions is accurately done using clear sky models like the I&P model. However, clouds can occur within the atmosphere resulting in additional interaction with the solar radiation. The field of irradiance forecasting explore the cloud solar radiation interactions to make irradiance predictions.

2.3 Irradiance Forecasting

The interactions between the Sun and the atmosphere result in significant variations of irradiances. The future irradiances can be approximated well during clear sky conditions with clear sky models; however, clouds often occur. The constantly changing atmosphere makes it hard to predict at which moment the clouds obscure the Sun. The field that investigates the atmospheric interaction with the Sun and the relation to the future irradiance is the field of irradiance forecasting. On a temporal scale, a distinction can be made between short-term (0-30 minutes), medium-term (30 minutes – 48 hours), and long-term forecasting (48 hours – 2 weeks+). Besides the temporal scale, the spatial scale also varies between 10 meters to more than 100 kilometres (Haupt et al., 2017).

Short-term forecasting is mainly performed using in-situ measurements; medium-and long-term forecasting relies on satellite images and NWP. However, often combinations of methods are used to increase overall accuracy, called hybrid models. In Figure 4, an overview of the different forecasting horizons, spatial scales, and temporal scales are depicted (Haupt et al., 2017).

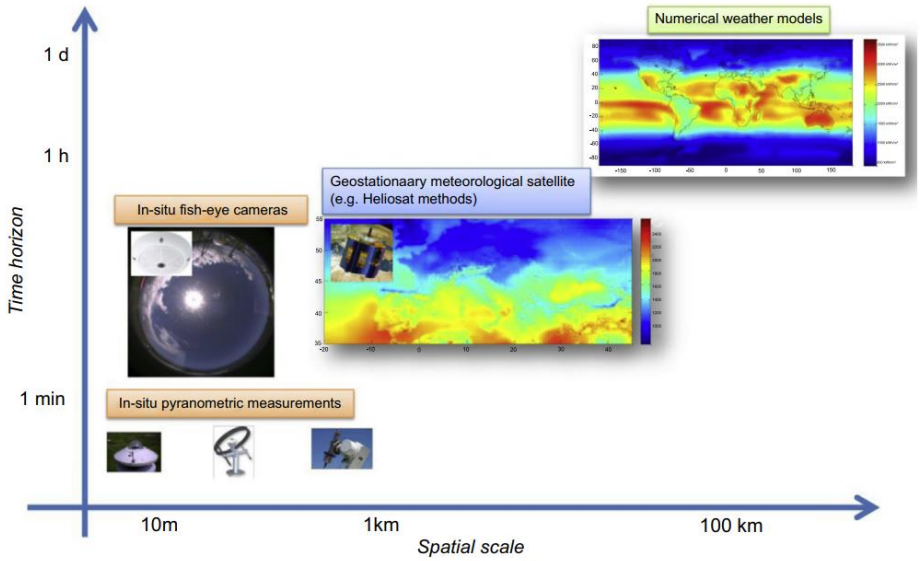


Figure 4: Different forecast horizons on temporal and spatial scale including the type of data used. Obtained from (Haupt et al., 2017)**.**

2.3.1 Numerical Weather Predictions

NWP was first conducted in 1904 (Bjerknes et al., 2009), where future weather events can be forecasted using fundamental conservation laws of nature. Combining these equations in mathematical form creates equations with the same number of unknowns as equations. A solution can be approximated, resulting in a forecast of how the atmosphere develops. To solve the system of equations of motion, the initial state of the atmosphere is crucial, being described by the temperature, humidity, winds, and more meteorological data. Since the analytical solution of the atmospheric equations is unknown, it is numerically approximated. The approximation is called the NWP and consists of two components: dynamic solver and physical parameterization schemes. NWPs are not used in this study and will therefore not be discussed in detail (Haupt et al., 2017).

- **Dynamic solver:** The atmospheric motion is described with differential terms and is algebraically approximated. The solver approximates advection, pressure gradients, Coriolis force, and more (Haupt et al., 2017).
- **Parameterizations:** Not all physical processes can be solved in a certain grid spacing or be represented analytically. These situations are included in NWP models in the form of physical parameterization schemes (Haupt et al., 2017).

Due to high computational requirements for the NWP, the grid spacing of the prediction is often several kilometres (Haupt et al., 2017). As a reference, the Royal Netherlands Meteorological Institute constantly makes an NWP of 48 hours within the future with a 1-hour interval of the Netherlands. The grid size used is 2.5 km^2 and still, the computation of a prediction takes 2.5+ hours using a supercomputer (KNMI, 2021).

2.3.2 Satellite-Based Predictions

Irradiance forecasting can be done using geostationary meteorological satellites for a forecast between 0 and 6 hours. The satellites scan large areas of the earth multiple times per hour, depending on the satellite. This

allows satellites to obtain information on a large temporal and spatial scale about cloud cover and movement. A range of geostationary meteorological satellites exists, namely families of the METEOSAT, GOES, FENG YUN, and HIMA-WARI satellites. Each satellite supplies information of a different geographical area, as depicted in Figure 5 (Haupt et al., 2017).

The frequency of image capturing within satellites is 15-minutes and often has a spatial grid size of 2-10 km^2. Due to the satellites their spatial and temporal resolution, they are ideal for a forecast horizon of 30 minutes to 6 hours, while in shorter forecast horizons in-situ measurements are preferred (Haupt et al., 2017).

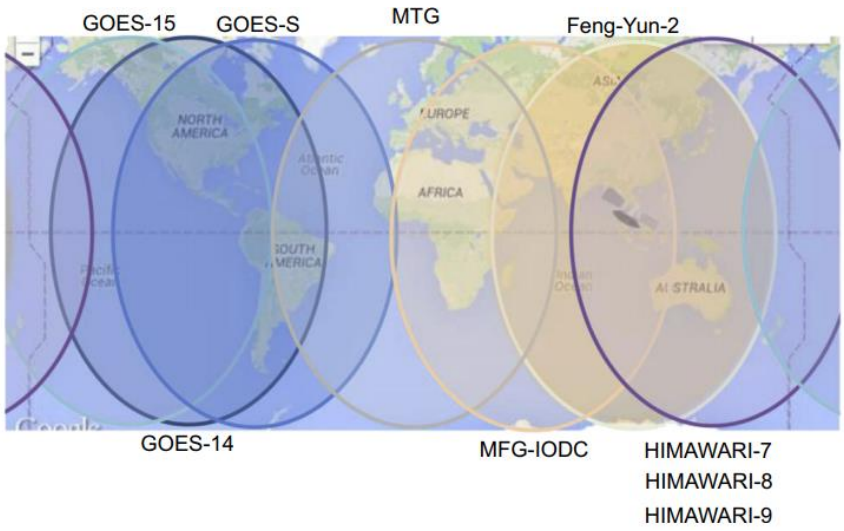


Figure 5: Geostationary satellite coverage for different satellites: MTG stands for METEOSAT. Obtained from (Haupt et al., 2017)**.**

Satellite images can be used in multiple ways for solar irradiance forecasting, but a primary distinction between (semi) empirical and physical-based methods can be made. This study will not explain these methods in-depth, but an in-depth view of these methods can be acquired from Heliosat (Heliosat, 2021). Due to the large spatial (2 km^2+) and temporal (15-minutes +) scales of satellites and NWP based forecasting methods, those methods are not suitable for short-term irradiance predictions on smaller spatial and temporal scales. In-situ measurements are suited for this due to their small spatial scale (~10m^2) and temporal scale (1-min) (Haupt et al., 2017).

2.3.3 All-Sky Imager based predictions.

To obtain a clear view of cloud configurations within a small spatial and temporal scale, All-Sky Imagers (ASI) can be used. ASIs collects images of the whole sky using a fish-eye lens with a 180-degree view. The ASI is positioned within a weatherproof box pointing upward to obtain a clear image of the sky and an example sky image is depicted in Figure 6. Different ASIs are available, where some ASIs come with software to extract essential parameters from the image. However, there are various possibilities to extract information from sky images, which will be discussed in this section. Besides ASIs, pyranometers are often used for short-term irradiance forecasting, due to the small temporal and spatial scales. A pyranometer measures the solar irradiance flux density in W/m^2 for a location (Haupt et al., 2017).

In Figure 6 is an example of an output sky image of an ASI illustrated. This sky image is nothing more than a 3D matrix with a certain width (W), length (L), and depth (d). The width and length determine how much pixels are within the image and the depth is the number of channels of which the image consists of. The depth usually consists of the three color channels red, green, and blue (RGB). Each value within the matrix has a value between 0 and 255, where each number decodes for a specific color of that channel (Haupt et al., 2017).

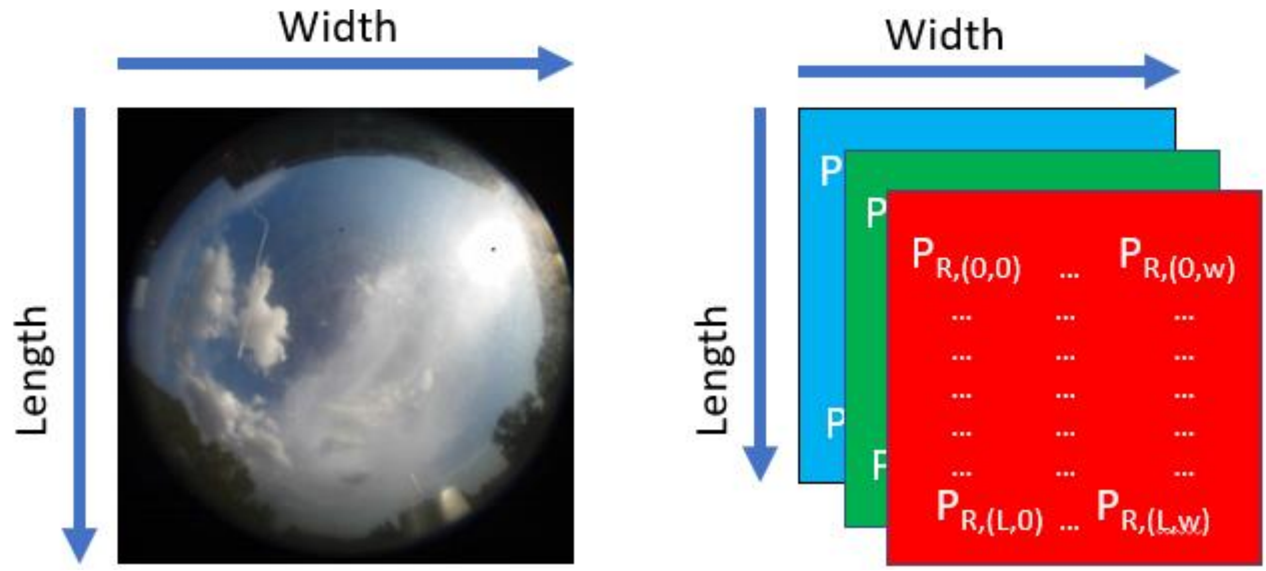


Figure 6: An example of a sky image and how the sky image can be represented as 3D matrix. The sky image is obtained from (Carreira Pedro et al., 2019)**.**

It is easy to distinguish clouds from clear skies with the naked human eye. However, to extract information from images using computers, image processing algorithms are required. An important extractable feature from sky images is the number of pixels that are clouds relative to the number of clear sky pixels, which is known as the cloud coverage (CC) as depicted in equation 2-8.

$$CC = \frac{\text{Cloudy Pixels}}{\text{Cloudy Pixel} + \text{Clear Sky Pixels}} \quad \textbf{2-8}$$

Besides the CC, another unit is often used to describe the cloud cover, called oktas. Oktas range from 0 to 8, where 0 means no clouds and 8 means completely overcast. When the CC is measured to be 50%, this can be expressed as 4 oktas (Spellman, 2013).

Distinguishing cloudy pixels from clear sky pixels is achieved using the RGB of each pixel. The skies are colored blue during clear skies, and therefore clear sky pixels are characterized with high intensity within the blue color channel. Clouds, however, appear white/greyish due to similar scattering within the blue and red channels (Kazantzidis et al., 2017). Several methods exist to use the different RGB pixel values to distinguish clouds and the methods are depicted in Table 1. An example of a sky image where clouds are distinguished from clear skies is depicted in Figure 7.

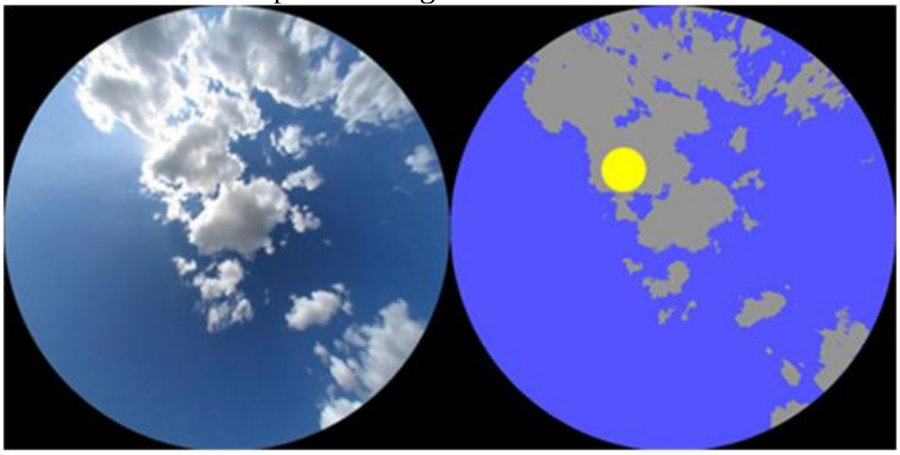


Figure 7: An example of a ASI image, on which clouds have been distinguished from clear skies using RGB values. Obtained from (Luiz et al., 2018)**.**

Table 1: Different methods to distinguish clouds from clear skies.

Paper	Method	Performance
(Long et al., 2006)	R/B > 0.6 → Cloudy ; Otherwise, Clear	Uncertainty of 20%
(Kreuter, 2009)	B /R > 1.3 → Cloudy ; Otherwise, Clear	73% Agreed within 1 okta
(Heinle et al., 2010)	R-B > 30 → Cloudy ; Otherwise, Clear	Outperforms Ratio Methods

Due to the variety of ASIs and lighting, the produced images differ in pixel intensities, and therefore applying suitable thresholds differs per camera and lighting level. Besides, problems arise due to the significant variation of clouds, changing aerosol properties, and solar zenith angles. Therefore, obtaining the cloud cover from RGB values with current methods is limited. A possible solution to this problem will be informed about in Section 2.4 (Haupt et al., 2017). The main limitations are depicted below.

- Cloud variation: The significant problem is that thick cumulus clouds can be distinguished from clear skies, but thin cirrus clouds cannot.
- Aerosols: Aerosols are not constant and different aerosols result in higher blue pixel intensities.
- Solar zenith angles: Due to changing solar zenith angles, the pixel intensities change during the day, which results in different suitable thresholds.

Besides the cloud coverage, other information can be extracted from sky images like raindrop detection, cloud classification, cloud height, aerosol properties and more. These features will not be discussed in this study but can be found in (Kazantzidis et al., 2017). Most image processing techniques are based on convolution, which is a matrix operation over an image (Haupt et al., 2017).

Convolution

As illustrated in Figure 6, an image is a 3-D matrix with a width, length, and depth. To extract information from images A filter or "kernel" can be applied over the image. A kernel is usually a small 2-D or 3-D matrix, and an example is depicted in Figure 8. Different kernels can be used, and the values within the kernel determine what kind of information is filtered. A filtered image is obtained by applying this kernel over a whole image, as depicted in Figure 8. Figure 8 is an example of smoothening of an image where a filter is applied to all locations within the image that takes the average of the central pixel and the four surrounding pixels. Therefore, edges within the image become blurry, and this process is used for smoothening of images. More information about convolutional operations can be obtained within the lectures of (Ludwig, 2007) and in the book of (Goodfellow, 2016).

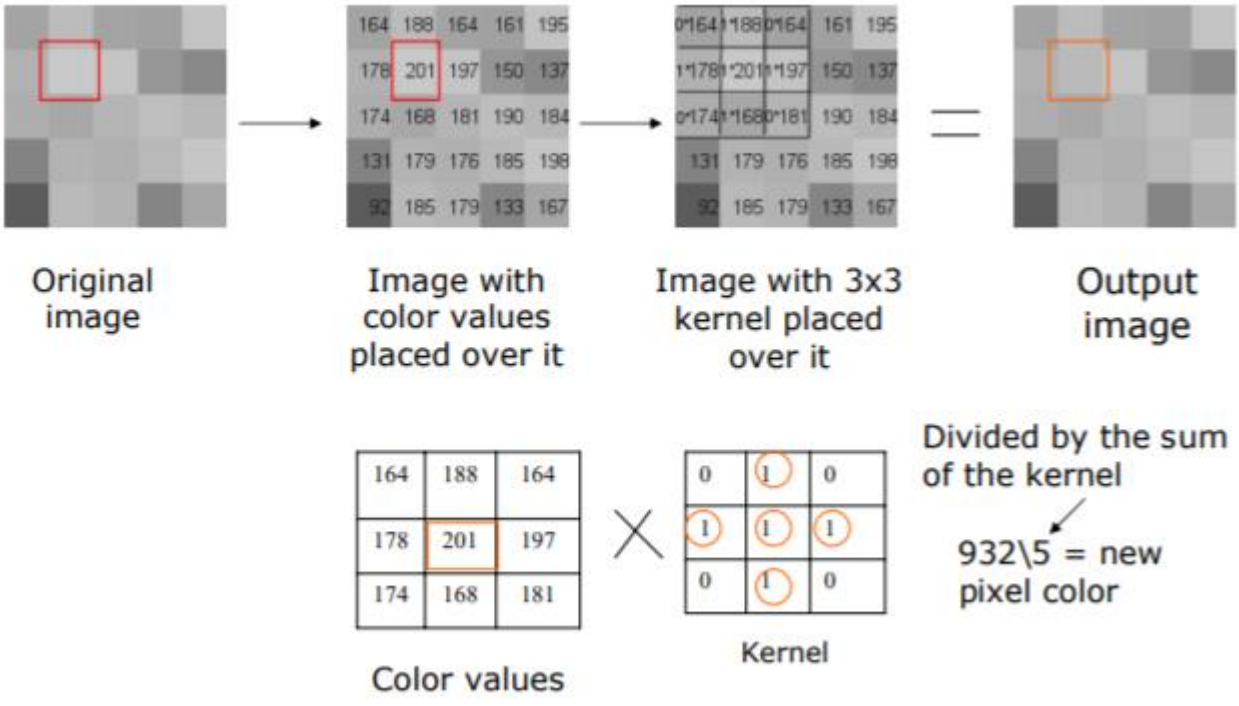


Figure 8: An example of a convolutional operation where the image is smoothed is shown. Obtained from (Ludwig, 2007)

With the detection of clouds in combination with a sequence of images and cloud tracking methods, the movement of clouds can be approximated. Different approaches for image-based cloud movement irradiance forecasting methods exists and are depicted in Table 2 (Yang et al., 2018).

Table 2: Different approaches to use sky images or satellite images to predict cloud movement and to obtain irradiance forecasts (Yang et al., 2018)**.**

Source	Method	Description
(Marquez et al., 2013)	Cloud Motion Forecast	Forecasting the cloud cover in the future.
(Chow et al., 2015)	Cloud Motion Vector	Uses images (satellite or ASI) to create a cloud motion vector field.
(Chow et al., 2011a)	Cross-Correlation Method	Matches blocks of pixels in subsequent images to create a vector of the cloud movement.
(M. Li et al., 2016)	Particle Image Velocimetry	Originally used to measure fluid velocity and can be used to obtain average cloud velocity.

However, the in Table 2 depicted methods suffer from problems like non-linearity within the image, cloud heights, and cloud deformations that the models cannot detect (Chauvin et al., 2016)(Yang et al., 2018).

- Non-linear movement: Using a fish-eye lens creates non-linearities further away from the center of the image, due to the 180-degree fisheye lens. Therefore, determining cloud movements away from the central segments of the image get more inaccurate.
- Cloud height: Within a sky image, the cloud height is unknown and cannot be extracted unless using two or more ASIs (Kazantzidis et al., 2017). The cloud height is essential when looking at the cloud movement, since low clouds and high clouds obscure the Sun differently, imposing different shading profiles.
- Cloud deformations: The movement within clouds is non-linear due to turbulent air flows. Besides, due to the continuous process of condensation and precipitation, the clouds can appear or disappear within minutes (Mölders & Kramm, 2014).

The extracted information from a sky image or a sequence of sky images in combination with irradiance measurements and meteorology data can be used to make predictions of future irradiances. The predictions can be made using irradiance map methods, regressive methods, and machine learning methods. An extensive range of varieties of these methodologies exists and are reviewed in the papers of (Kazantzidis et al., 2017) and (Yang et al., 2018). In this study DL methods are applied, which are a part of ML. This study will therefore first discuss the ML methods followed by the DL methods.

2.3.4 Machine Learning

Within computer engineering, there is the field of artificial intelligence (AI). A branch of AI is Machine Learning (ML), which can learn from data. There are many definitions of ML, but (Mitchell, 1997) provides a short, clear description "A computer program is said to learn from **experience** E with respect to some class of **tasks** T and **performance** measure P, if its performance at tasks in T as measured by P, improves with experience E." –(Mitchell, 1997) (Goodfellow, 2016).

ML **tasks** determine the processing of an example. An example consists of quantitatively measured features that the ML system needs to process. The features are represented as a vector $x \in R^n$ in which each x_i is a different feature. Within a sky image, each pixel value would be a single feature, and the whole image the example (Goodfellow, 2016).

The **experience** of the ML algorithm consists of two categories, namely, unsupervised and supervised. Within supervised learning, the input and outputs are known for the algorithm, and for unsupervised learning the outputs are not known. This study will concern supervised learning, and therefore unsupervised learning will not be discussed but can be found in the book of (Goodfellow, 2016). ML supervised algorithms experience a

dataset (a collection of examples and an example can be seen as a data point) with features correlated with a label (the quantity to predict). The ML algorithm observes vectors consisting of *x* related with a label vector *y* and learns to predict *y* using the inputs *x, p(y | x)* (Goodfellow, 2016).

The **performance** describes how well a task is performed. Depending on the problem, a different metric to evaluate the performance can be used. The main difference is between classification problems and regressive problems. An example of a classification problem is when weather conditions have to be classified into clear sky or cloudy sky. The answer is either clear or cloudy and is therefore best described by looking into how much is correctly classified (accuracy). An example of a regressive problem is predicting the irradiance, which can be a quantity that can have a large range of values and cannot be distinguished into classes. Evaluating the performance of such a model with accuracy would not be suitable, and regressive problem are better evaluated looking at the difference between the predicted and actual values. Common metrics used for regressive problems are the mean average error (MAE), root mean squared error (RMSE) or mean squared error (MSE). In this study the goal is to predict the irradiance, so the focus will be on regressive problems (Goodfellow, 2016).

A simple example of a ML algorithm to predict the irradiance will be given to elaborate on how ML works. The inputs for the algorithm are the measured *GHI* ($GHI^{measured}$) and the CC at t=0, where '*t*' denotes the time. The output label to predict within the ML algorithm is the $GHI^{measured}$ at *t*=1,…,5. The ML algorithm takes the vector $x \in R^n$ as input to predict $y \in R$ as the output using a vector of weights $w \in R^n$ as denoted in Equation 2-9 where the superscript 'T' denotes that the vector or matrix is transposed. For this example two weights are used (Goodfellow, 2016).

$$\hat{y} = w^T * x \qquad \textbf{2-9}$$

$$\widehat{GHI^{measured}_{t=1,\dots,5}} = \begin{pmatrix} w_0 \\ w_1 \end{pmatrix}^T * \begin{pmatrix} CC_{t=0} \\ GHI^{measured}_{t=0} \end{pmatrix}$$

For ML large datasets are used to train the weights that determine the dynamics of the model and for this example a large dataset consisting of a timeseries of sky images and measured GHI is used. The dataset is used to make examples of how the model should behave by making datapoints consisting of GHI and CC at t=0, correlated with GHI at t=1,…,5. The dataset consisting of m examples are split into m_{train} and m_{test} where the ML algorithm will be trained on m_{train} and tested on the m_{test} dataset. The training dataset is used to learn what the optimal configurations of the weights are to map the inputs to the output. Subsequently, the test dataset is used to evaluate how well the model performs on data outside of the training dataset. In this example the *MSE* is used to optimize the weights and evaluate the performance of the ML algorithm, as is depicted in equation 2-10 (Goodfellow, 2016).

$$MSE_{test} = \frac{1}{m_{test}} \sum_i (\widehat{y^{test}} - y^{test})_i^2 \qquad \textbf{2-10}$$

From equation 2-10 it becomes clear that the MSE goes to 0 if the prediction meets the label $\widehat{y^{test}} = y^{test}$. The error of the prediction increases if the Euclidean distance between the label and prediction increases. The key to the ML algorithm is that the weights are configured to an optimal solution to minimize the MSE, which can be done using the experiences captured within the m_{train} dataset. Minimizing the MSE on the m_{train} dataset is reached where the gradient is 0 as depicted in equation 2-11, which can be used to obtain the simple learning algorithm depicted in equation 2-14, using equations 2-12 and 2-13. A more comprehensive deviation is depicted in the book of (Goodfellow, 2016).

$$\nabla_w MSE_{train} = 0 \qquad \textbf{2-11}$$

$$\nabla_w \frac{1}{m} ||\widehat{y^{train}} - y^{train}||_2^2 = 0 \quad \textbf{2-12}$$

$$\nabla_w \frac{1}{m} ||X^{train} w - y^{train}||_2^2 = 0 \quad \textbf{2-13}$$

$$w = (X^{train\,T} X^{train})^{-1} (X^{train\,T} y^{train}) \quad \textbf{2-14}$$

The linear regression example given above is simple and more complicated ML algorithms exist. Minimal complexity can be added by adding a bias term '*b*' that result in the equation in 2-15. The bias term can be seen as an extra input with a value of 1 that can transpose the function, so the ML algorithm can fit the data better. Instead of adding the bias to the model, the bias can be added to the feature vector x with a value of 1. The weight corresponding to the bias will determine the influence of the bias (Goodfellow, 2016). More complicated ML algorithms exists but will not be discussed into depth in this study, since the main topic is deep learning (DL).

$$\hat{y} = w^T * x + b * \quad \textbf{2-15}$$

$$\widehat{GHI}_{t=1,\ldots,5}^{measured} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}^T * \begin{pmatrix} CC_{t=0} \\ GHI_{t=0}^{measured} \\ b_0 \end{pmatrix}$$

A problem that arises from ML algorithms is to determine what information must be extracted. As discussed earlier, image processing techniques can extract the features from sky images but are not always accurate. In the example given above, image processing techniques are used to extract the CC from the sky image. If the image processing techniques fail to be effective, the extracted features are of low quality, and the ML algorithm will most likely fail to make a good prediction. By applying image processing techniques, the ML algorithm is limited in how to interpret information from the sky image. ML algorithms are sufficient to find correlations in some approaches; however, when the problem becomes more complex, ML algorithms fail to be adequate, especially when images are used as input. To overcome the limitations of image processing techniques and allow more complexity in ML algorithms, DL algorithms can be used. DL can be seen as ML algorithms with multiple steps (Goodfellow, 2016).

2.4 Deep Learning

As ML is part of AI, DL is a part of ML. DL technology is used in intelligent software and has proven to be effective for several applications, especially using image data. DL is a large field of research, and only the deep learning parts of interest in this study will be discussed. In this study, only deep feedforward networks are explained. In the Appendix certain DL terms are explained for when some terms are not clear to the reader.

2.4.1 Deep learning general

The goal of feedforward neural networks is like ML algorithms, namely, to map a vector input *x* to a label *y* and can be mathematically defined as in equation 2-16. The values of the weights are learned to create a function that best describes the prediction of *y* using input *x* (Goodfellow, 2016).

$$y = f(x; \mathrm{w}) \quad \textbf{2-16}$$

Feedforward networks are called a network since multiple learning algorithms are connected to form a network. Instead of using a single set of weights to map the vector *x* to an output *y*, a feedforward network consists of multiple steps. The input *x* can first be mapped using a set of weights to an intermediate representation (*h*) of the problem. Subsequently the intermediate representation of the problem is used as input

in another learning algorithm. This way, multiple learning algorithms can be stacked, where each learning algorithm is called a "layer". An example of a network consisting of 3 layers can mathematically be described with equation 2-17, where each function ($f^{(3)}$, $f^{(2)}$, $f^{(1)}$) is a different layer of the network. The first layer ($f^{(1)}$) takes the input and creates an intermediate representation ($h^{(1)}$) of the problem. The first intermediate state is fed into the second layer, creating another intermediate representation ($h^{(2)}$). The last layer takes ($h^{(2)}$) as input to map it to the output. In a feedforward network multiple layers can be used, where more layers give more depth to the model, hence the name deep learning (Goodfellow, 2016).

$$f(x) = f^{(3)}\left(f^{(2)}\left(f^{(1)}(x)\right)\right) \quad \textbf{2-17}$$

When a feedforward network is trained, the goal is to create a function that maps the input to the output as good as possible. The feedforward network uses multiple layers to map the input *x* to the output *y* using multiple intermediate representations. When the model is trained, the model knows which inputs result in which outputs y. However, the intermediate representations do not have labels to them and are not specified in the dataset. The learning algorithm must determine the usage of the intermediate layers to be of optimal use to map the inputs to the output. Since the outputs of the intermediate layers are unknown, they are termed hidden layers (Goodfellow, 2016).

If the feedforward networks would only consist of linear operations within multiple layers, the function to map the input to the output would be a linear function. However, regressive problems are not always ideally described using linear functions, so non-linearity needs to be added. Adding nonlinearity to the input of a layer is achieved by transforming the input x using (φ(x)), where φ is a layer that applies a nonlinear transformation. The input x now has a transformed representation determined by φ. A feedforward network can now be represented as depicted in equation 2-18. The weights within the model are used to learn the mapping from φ(x) with the weights to the corresponding output (Goodfellow, 2016).

$$y = f(x, \mathrm{w}) = \varphi(x^T w) \quad \textbf{2-18}$$

A feedforward network is built of multiple layers, where each layer transforms the previous input to an output that can be used in subsequent layers. Two connected hidden layers are depicted in equations 2-19 and 2-20. The first layer takes the input x and transforms it to a transformed representation of the input $h^{(1)}$ using the activation function ($g^{(1)}$), weights ($W^{(1)}$) and biases ($b^{(1)}$) of the first layer. Subsequently, the transformed representation of x denoted as $h^{(1)}$ will be used as input for the second layer to obtain the transformed representation of $h^{(1)}$, namely $h^{(2)}$. This example consists of two layers, but multiple layers can be used in a similar way to create more depth into the DL model (Goodfellow, 2016).

$$h^{(1)} = g^{(1)}(W^{(1)T}x + b^{(1)}) \quad \textbf{2-19}$$

$$h^{(2)} = g^{(2)}(W^{(2)T}h^{(1)} + b^{(2)}) \quad \textbf{2-20}$$

Within a DL model different design decisions can be made. One of the design decisions is the activation function.

Activation Functions

As explained above are activation functions used to add non-linearity within a DL model. In this study two different activation functions are used and will be described. The most used activation function is the rectified linear unit (ReLU) activation function due to its computational efficiency depicted in equation 2-21. The ReLU applies a nonlinear transformation, because it is a piecewise linear function built out of 2 pieces as depicted in Figure 9. If the input *z* into the activation function is positive, the ReLU outputs *z*, and if the input is 0 or negative, the ReLU outputs 0 (Goodfellow, 2016).

$$g(z) = \max\{0, z\} \quad \textbf{2-21}$$

The other activation function used is the hyperbolic tangent (Tanh) function which is mathematically depicted in equation 2-22 and visually represented in Figure 9. Tanh activation function forces values to range between [-1, 1] and is centred around 0. A problem that can occur is the vanishing gradient for high or low values of z. The vanishing gradient problem happens when the gradient of the loss function goes to 0, which limits the training capability of the DL model, since it is used to update weights. The training process of DL models is explained below.

$$\tanh(z) = \frac{2}{1 + e^{-2z}} - 1 \quad \textbf{2-22}$$

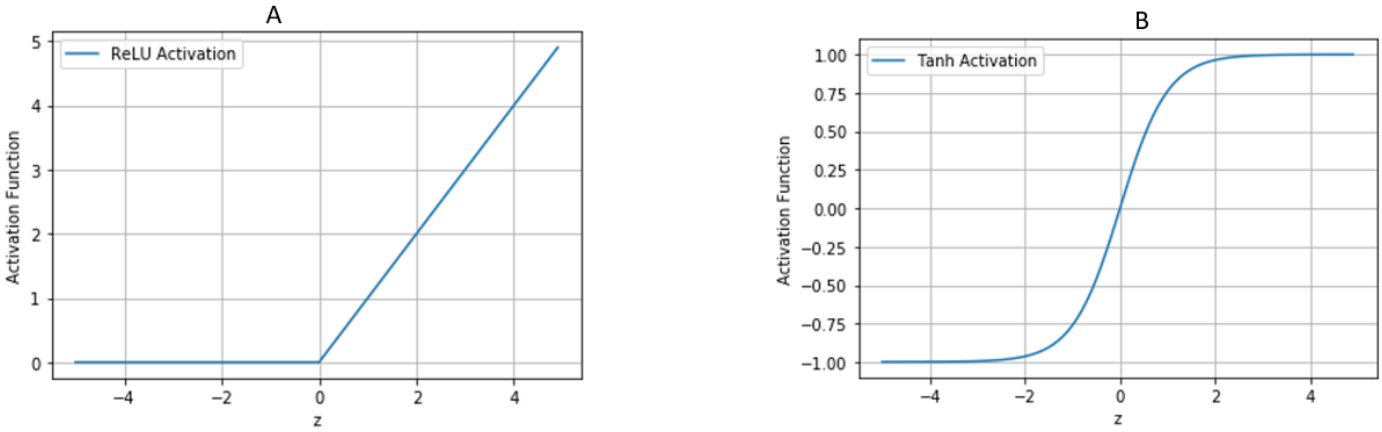


Figure 9: Depicts the behavior ReLU activation function on the left in Figure A, and the of the Tanh activation function on the right in Figure B.

2.4.2 Deep Learning Training Algorithm

The deep feedforward network uses input *x* to generate $\hat{y}$ by moving the input *x* forward through the different layers. The input is transformed within the layers until the final layer produces the output. This process is called forward propagation. Before a DL model is trained all the weights are randomly initialized and the forward propagation will likely result in a poor prediction. To increase the performance within the DL model, the weights must be learned to increase the performance. The weights are adapted using the loss function and stochastic gradient descent with back propagation (Brownlee, 2021).

Loss function: To evaluate the performance of the DL model a loss function is used. Similar as in ML, the performance of a regressive model can be evaluated with the MSE. As explained above, the initial error between the prediction and measured value will likely be large, due to the random initialized weights. The DL model will adapt the initialized weights to optimize the loss function, so minimizing the MSE (Brownlee, 2021). How the DL model assesses the weights and changes them is explained in the following section.

Stochastic gradient descent: This is the optimization algorithm that tweaks weights within the DL model in a way that the inputs for the DL model result in an optimal value for the loss function using the gradient to do so. The first derivative of the loss function determines if the slope of a curve goes up or down. Within DL often a vector of input variables is used, and the first-order derivative of a vector is called the gradient target function. The gradient is a vector of partial derivatives of a loss function using a vector of inputs. The gradient descent algorithm determines if the loss function goes up or down, with respect to input values. The gradient of each input variable is followed downward to adapt the weights within the DL model, resulting in new values. This process is repeatedly done to optimize the weights within the DL model by finding the minima of the loss function as depicted in Figure 10 (Brownlee, 2021).

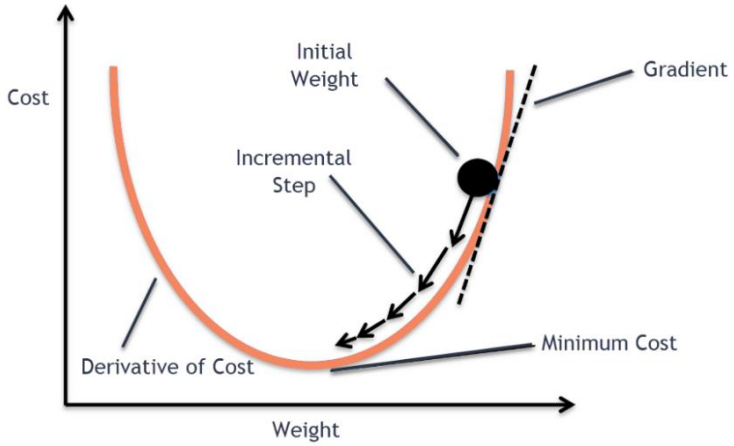


Figure 10: A 2-D example of the stochastic gradient descent model is shown. Within the figure the cost function is similar to the loss function. Obtained from (Lanham, 2021)**.**

Within DL a dataset is used to train the model and minimize the loss function. The inputs of the dataset are used within the DL model to make a prediction and with the label of that datapoint the MSE is calculated. The gradient descent algorithm is used to look how the weights within the DL model should be adapted to increase the performance regarding the MSE loss function. The term 'stochastic' is due to the noise within the data which results that the gradient is sometimes tweaked in the wrong direction. The derivatives of each weight are calculated using the back propagation algorithm (Brownlee, 2021).

Back propagation: Is the algorithm to calculate the gradient of a loss function depending on the weights. It calculates the gradient for each specific weight within a DL model, which is used to update that specific weight. The gradients for each weight within the DL model are calculated backward, starting from the output to the input, hence the name 'backpropagation'. The outcome of the loss function is dependent on the weights. The gradient of the loss function with respect to the weights are the weight gradients and are used to update the weights. Since DL models consist of multiple layers, the weight gradients within the layer are not directly connected with the output and can therefore not be instantly assessed. To obtain the gradient of these weights with respect to the loss function, the chain rule is recursively applied. The derivative of each sub-function is calculated using the derivative of the parent function for which the derivative is known. This way, the derivatives of weights in layers further away from the output layer are assessed (Brownlee, 2021).

With the basic knowledge about how a DL model operates and learns from data the different types of DL layers will be depicted. Not all layers within DL are discussed, only the layers that are of interest for this study. For a deeper understanding of how DL works, the reader is referred to the book of (Goodfellow, 2016).

2.4.3 Deep Learning Layers

Dense Layer:

A dense layer is the most common and frequently used layer within DL. A visual representation of a simple DL models consisting of two dense layers is depicted in Figure 11. Within Figure 11 the weights are denoted with 'w', the bias is depicted as 'b', the hidden states as 'h', the output as 'y' and the input as 'x'. The example consists of two inputs x_0 and x_1 mapped to the output y_0 using two hidden layers, layer one and layer two, each consisting of 3 units. The inputs are first multiplied by the weights of layer one and summed to be transformed using the activation function of layer one to obtain the hidden states of layer 1 ($h_{0,0}$; $h_{0,1}$; $h_{0,2}$). The hidden states of layer 1 are then a transformed representations of x_0 and x_1 and are used as input for layer 2, following a similar procedure. An example of how a single hidden state ($h_{0,0}$) of layer 1 within a dense layer is calculated is depicted in equation 2-23 where 'g' denotes the activation function (Goodfellow, 2016). In the example given, small layers of only 3 units are used, but in practice usually a significantly greater number of units are used. Similar as in ML algorithms, the bias term has a value of 1 and is an additional input. Each dense layer has a bias and is used to guarantee that a neuron will be activated, even if all other input values are 0 and to

correct for systematic errors in predictions (De Luca, 2019). The reasons to use bias holds for all different DL layers.

$$h_{0,0} = g_0(x_0 * w_{0,0} + x_1 * w_{0,3} + B_0 * w_{0,6}) \quad \textbf{2-23}$$

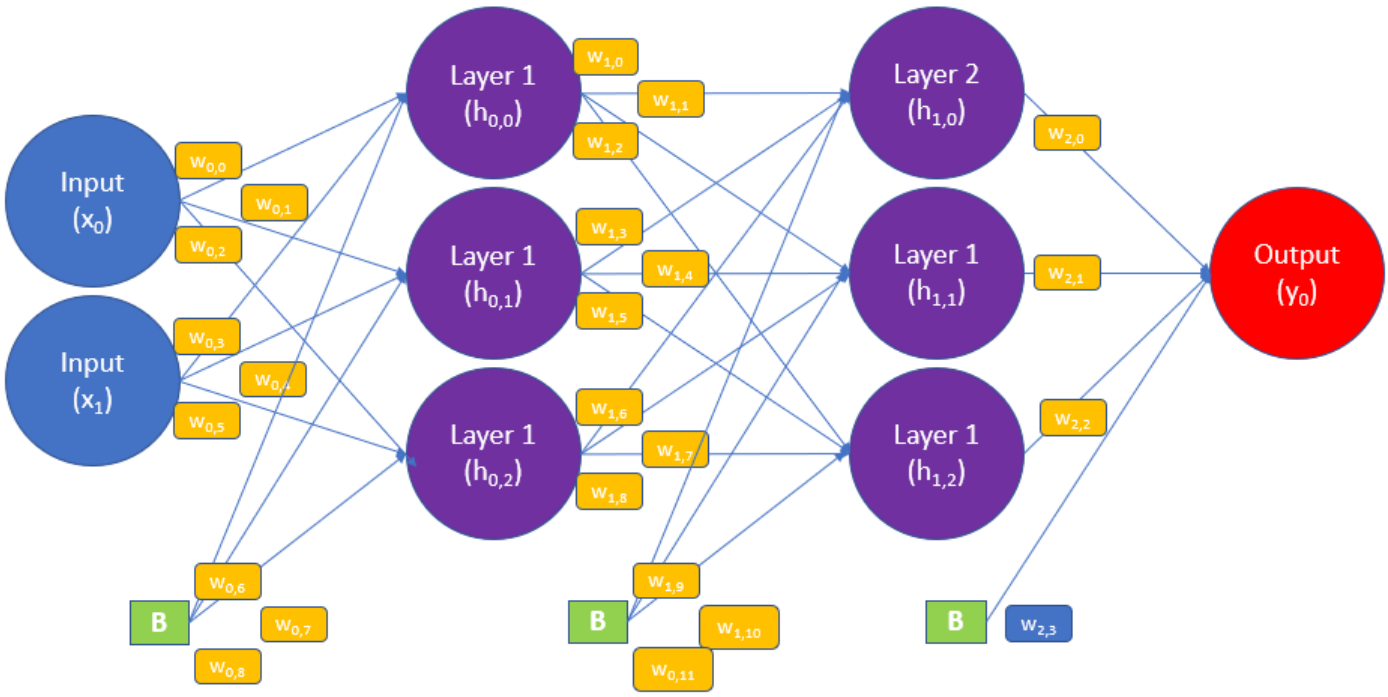


Figure 11: A simple example of a DL model with two dense layers where two inputs (x_0 & x_1) are mapped to the output y_0. Each hidden layer consists of 3 units.

If a sky image is used as input, the input vector becomes large, since each pixel will be a separate input. This results in a large number of weights, and therefore long training times. To extract valuable information from images without having long training times a convolutional layer can be used, which is explained in the next section.

Convolutional Layer:

Convolutional neural networks (CNN) have proven to be effective in processing images due to the powerful feature learning ability and are efficient using weight sharing. The name comes from the linear convolutional operation and instead of using a kernel with fixed values determining what information to extract from the image, the values within a kernel of a CNN can be seen as group of weights that are trained using the DL learning algorithm. The DL model is therefore able to learn what set of weights within the kernel are ideally used to optimize the loss function. In other words, the DL model learns what information to extract from the image to reduce the loss function. In this study a 3-D color image is used as input and applying convolution over a color image requires 3-D convolution. This process is like 2-D convolution as explained in Section 2.3.3, but instead of using a kernel of weights with a length and width, the filter also has a depth with the same number of dimensions as the input. Besides the weights, each kernel also has a single bias term added for similar reasons as in the dense layers. The powerful feature learning ability is enabled through the convolutional operation and the weights are shared since each kernel consisting of weights is applied over the whole image, thus sharing the same weights for multiple pixels (Mayank, 2020).

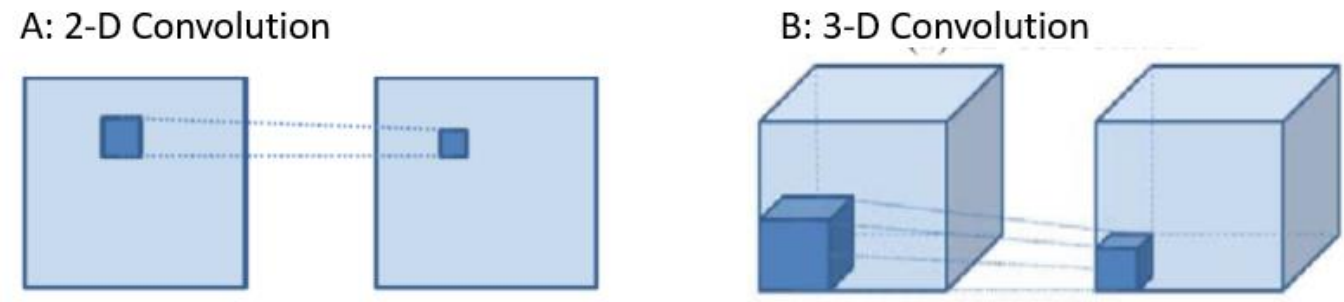


Figure 12: Shows the process of 2-D convolution on the left and 3-D convolution on the right. It is an example of how a feature map is obtained by applying convolutional over a 2-D matrix and 3-D matrix. Obtained from (Mayank, 2020)**.**

Within a convolutional layer multiple kernels can be used, where each kernel creates a separated feature 2-D map. The feature map of each kernel is stacked to form a 3-D matrix consisting of a width, length, and a depth. The 3-D output of a convolutional layer can be used into a subsequent convolutional layer as input, again performing 3-D convolution. This way convolutional layers can be stacked to create more depth and allow to extract more complex information (Mayank, 2020). To clarify this an example regarding a sky image is given.

A sky image consists of pixels where some pixels show the Sun and other pixels the clouds and clear sky. To extract complex information from this image 3 convolutional layers are applied.

- The first convolutional layer learns to use a kernel that can create a feature map that consists solely of cloudy pixels, thus creating a feature map representing the cloudy pixels within the image.
- The cloudy pixel feature map is subsequently used in a second convolutional layer to extract more information. The second convolutional layer learns to use a kernel that extracts information about the opacity of the cloudy pixels.
- The feature map containing information about the opacity of the clouds is subsequently used in a third convolutional layer. The third convolutional layer learns to use a kernel that extract information about the structure of the cloud.

Important to note is that the example above is imaginary, and it is unknown what information the convolutional layers are extracting. However, the hidden state outputs of convolutional layers can be visualized to get a better intuition of what information is extracted. An example of intermediate states of convolutional layers extracted from sky images within this study after the first convolutional layer is depicted in Figure 13.

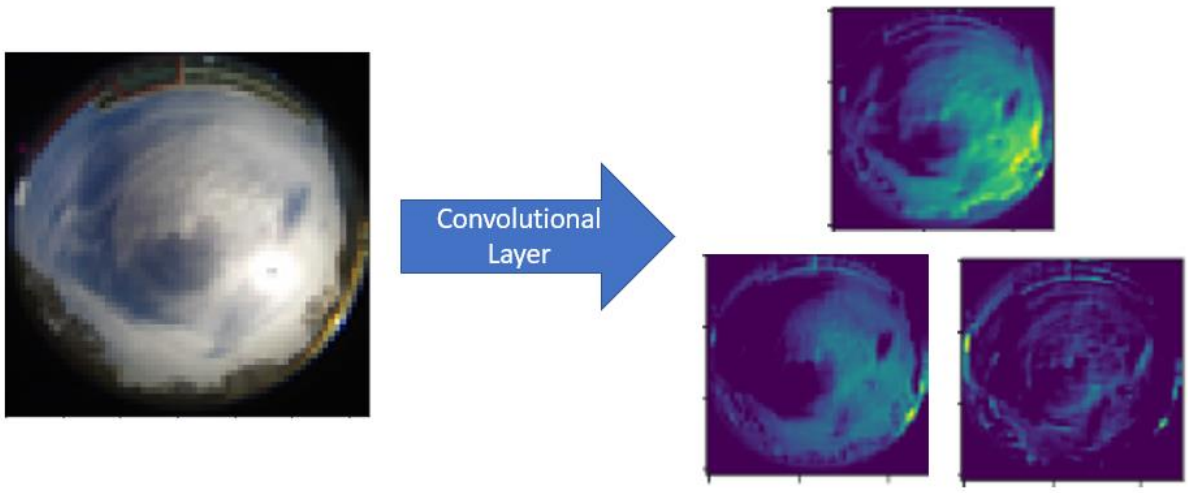


Figure 13: An example of extracted information from the first convolutional layers using a sky image as input.

Aside from the convolutional layer, a convolutional network is often comprised of multiple other layers to make the convolutional process more efficient. These layers are pooling layers and flatten layers and will be discussed in the following section.

Pooling & Flatten Layer:
Convolutional layers are more effective at interpreting images compared to dense layers, however applying many kernels still results in many weights to train and computations to perform, especially for large image dimensions. To reduce the number of trainable parameters, while losing minimal information about the image, pooling layers are used. The combination of a convolutional layer with a pooling layer is called a CNN block.

Typically, a CNN block has three stages. First, the number of kernels is applied to convolve the input to create a set of feature maps. The linear activations go through a nonlinear activation function, like the ReLU. After the nonlinear activation layer, the output is often pooled, reducing the output size. Pooling is a straightforward process in which a kernel is applied, but instead of doing convolution, the maximum value within the kernel size is taken while reducing the size of the matrix depicted in Figure 14 (Goodfellow, 2016).

$$\begin{matrix} 12 & 4 & 31 & 0 \\ 21 & 8 & 0 & 2 \\ 33 & 73 & 41 & 5 \\ 98 & 102 & 22 & 8 \end{matrix} \rightarrow Max\ Pooling(2x2) = \begin{matrix} 21 & 32 \\ 102 & 41 \end{matrix}$$

Figure 14: An example of how the process of pooling works within deep learning.

A DL model can consist of multiple CNN blocks, where the output of a CNN block is used as input for the next CNN block. During the convolutional layers combined with pooling, the size of the matrix decreases. Using multiple CNN blocks allows the extraction of more complex features from the image. Clarification is given with an example using a 64x64 matrix.

When using a 64x64 image as input into a CNN layer with 20 kernels of size 7x7, the output of this layer is a matrix of 20 x 58 x 58, as depicted in equation 2-24. The output of this layer is activated and then pooled with a kernel size of 2x2, resulting in a layer of 29 x 29 x 20 shown in equation 2-25. The output of the first CNN block is used in the second CNN block that applies 30 kernels of size 5x5, which results in an output matrix of 30 x 25 x 25, depicted in equation 2-26. The output of this layer is activated and pooled, resulting in a matrix of 30 x 12 x 12 shown in equation 2-27. In this way, multiple CNN blocks can be used for more complexity, but the kernel size cannot exceed the input size of the convolutional block.

CNN Block 1

$$20 \times \big(64 - (7-1) \times 64 - (7-1)\big) = 20 \times 58 \times 58 \qquad \textbf{2-24}$$

$$20 \times 58\,x \times 58 \rightarrow Max\ Pooling\ (2,2) = 20 \times 29 \times 29 \qquad \textbf{2-25}$$

CNN Block 2

$$30 \times \big(29 - (5-1) \times 29 - (5-1)\big) = 30 \times 25 \times 25 \qquad \textbf{2-26}$$

$$30 \times 25 \times 25 \rightarrow Max\ Pooling\ (2,2) = 30 \times 12 \times 12 \qquad \textbf{2-27}$$

The output of the final CNN block within a DL model outputs the extracted features from the image. To use the image features to obtain more complexity, the output of the final CNN block can be used within a dense layer. However, the output of the CNN block is 3-or 2-D, while a dense layer can only use a 1-D input. To convert the 3-or 2-D output to a 1-D vector, a flatten layer is applied. A flatten layer converts multi-dimensional matrix to a 1-D vector and is depicted in Figure 15. If the output of the CNN Block 2 in equation 2-27 is flattened, the 3-D matrix of 30x12x12 would be converted to a 1-D vector with a size of 4320.

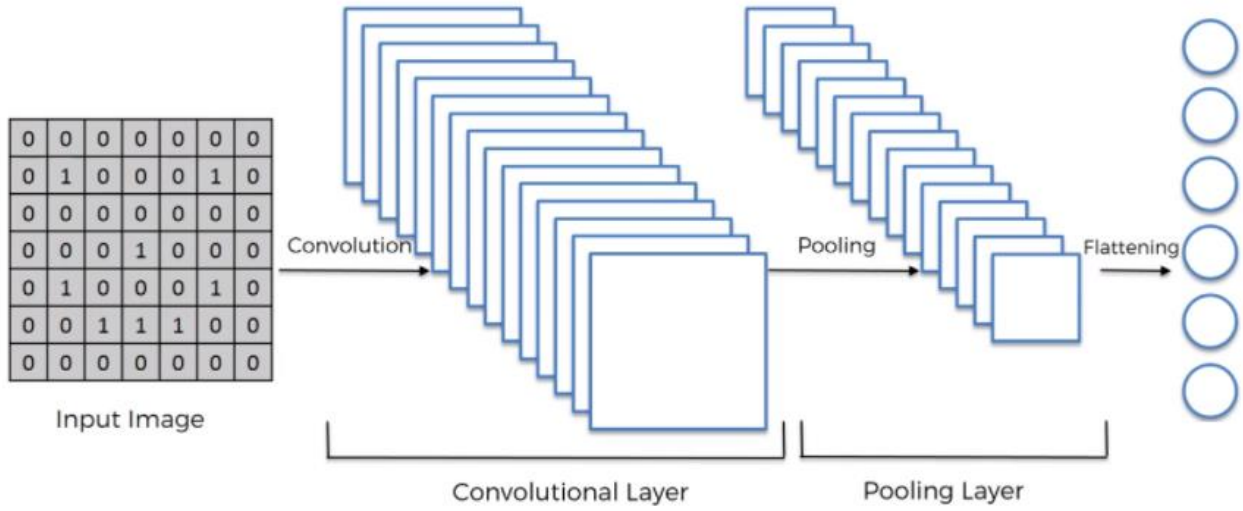


Figure 15: An illustration of how other layers are used within a CNN block to output a 1-D feature vector. Obtained from (Panadda, 2019)**.**

Recurrent Neural Network Layers:

Where dense neural networks excel to interpret 1-D vector inputs and CNNs in interpreting images, another kind of neural network is specialized in handling sequences of data $x^{(0)},\ldots, x^{(t)}$. The layer specialized in sequence data is the recurrent neural network (RNN). Sequential data is time series data, which are data points ordered in time. In dense layers and convolutional layers is only the current input considered to create an output. However, within a RNN the input goes through a loop using the current input and what is learned from previous inputs. The main difference between a RNN and a dense layer are depicted in Figure 16. A RNN layer has internal memory, that can be used to store information from previous timesteps to subsequently use in later timesteps. The inputs to a RNN network are therefore the new input combined with stored information from previous inputs (Donges, 2021).

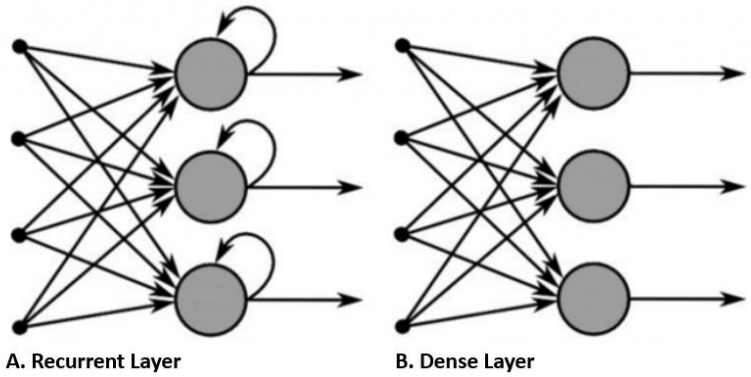


Figure 16: The difference between a recurrent layer depicted in A, and a dense layer depicted in B. Obtained from (Donges, 2021)**.**

To clarify why RNNs are important, an example is given concerning clouds in a sky image. If a sky image is used to predict what the irradiance is in the future, it is important to know when a cloud obscures the Sun. If a single image is used, it is possible that the cloud is in proximity of the Sun but does not obscure it yet. From that single image it is not possible to know in which direction the cloud moves. However, if a sequence of sky images is used a RNN can be used to store information from previous images. Within the RNN the direction of the cloud movement can be extracted and stored, so it can be used to predict the irradiance into the future. If the cloud moves away from the Sun, the model will likely predict that the irradiance is high, but if the clouds move towards the Sun, the model will likely predict a lower irradiance.

As explained, RNNs operate on a sequence of input vectors $x^{(t)}$. Any feedforward neural network where the function shows recurrence is an RNN. The equation for RNN is depicted in equation 2-28 in which h denotes the hidden state, which is a product of the previous hidden state ($h^{(t-1)}$), the input x, and the internal weights (Goodfellow, 2016).

$$h^{(t)} = f(h^{(t-1)}, x^{(t)}; \mathrm{w}) \qquad \textbf{2-28}$$

The RNN is trained to predict the future from a past sequence and learns to use the hidden states (h^t) to summarize relevant aspects observed in the past sequence. The input x can be a long sequence, where the output summary of relevant aspects is denoted with a fixed vector with a length of $h^{(t)}$. An unfolded representation of an RNN is depicted in Figure 17 (Goodfellow, 2016).

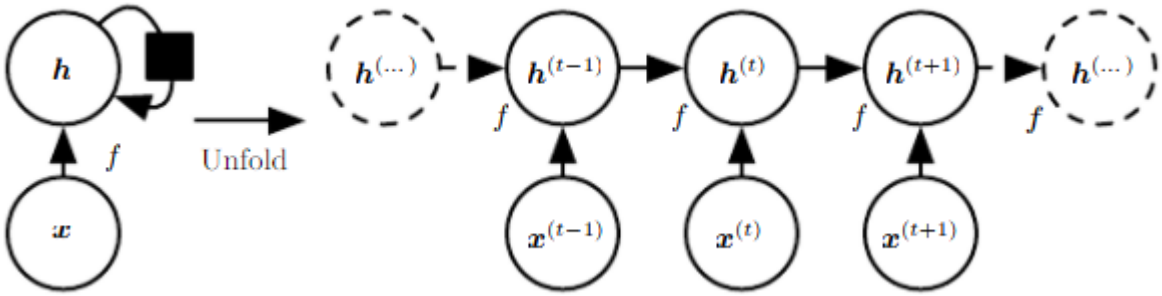


Figure 17: Shows an unfolded RNN. (Goodfellow, 2016)

Mathematically the following operations are performed within the RNN. First, the previous state is multiplied by the weights of the state w_s and the input is multiplied by the input weights w_x which are then added with the bias b, denoted in equation 2-29. The output a^t is transformed to the hidden state using the tanh activation function as depicted in 2-30. The output of the RNN is obtained by multiplying the latest hidden state with the output weights $\mathrm{w_y}$ as depicted in equation 2-31 (Goodfellow, 2016). For clarification the following example is given. If a sequence of 10 timesteps is used, the information of the first timestep ($x^{(0)}$) is ideally remembered for the output y_t. The hidden state of the first sequence input ($h^{(0)}$) is combined with the next sequence input ($x^{(1)}$) to create a new hidden state ($h^{(1)}$). Which information of the previous hidden state ($h^{(0)}$) is kept and discarded for the next hidden state ($h^{(1)}$) is determined by the weights. This way information from the first timestep can be stored to use in the last timestep and produce an output.

$$a^t = b + h^{(t-1)} w_h + x^{(t)} w_x \qquad \textbf{2-29}$$

$$h^{(t)} = \tanh(a^t) \qquad \textbf{2-30}$$

$$y_t = w_y h^{(t)} \qquad \textbf{2-31}$$

More variations on the RNN are possible, like Long Short Term Memory (LSTM) cells, Gated Recurrent units, and transformers, which will not be discussed in this study but can be found in the book of (Goodfellow, 2016).

3 Methodology

3.1 Dataset and Pre-processing

All the data is obtained within California in the Folsom Area at the coordinates 38.642° - 121.148° depicted in Figure 18, made publicly available by (Carreira Pedro et al., 2019). Folsom is a city in Sacramento Country in the Central Valley of California, characterized by a temperate climate and dry, hot summers. All the measurement instruments were installed in 2012 and consist of pyranometers, an ASI, and an all-in-one weather sensor. All the measurement instruments are positioned near each other, as is illustrated in Figure 19. All the measurements were logged with a CR1000 datalogger that records data with 1-minute average rates with synchronized clocks using on-site Network Time Protocol (NTP) to improve consistency. The data collection took place during the years 2014, 2015, and 2016. (Carreira Pedro et al., 2019). More information about the used measurement tools can be found in the Appendix.

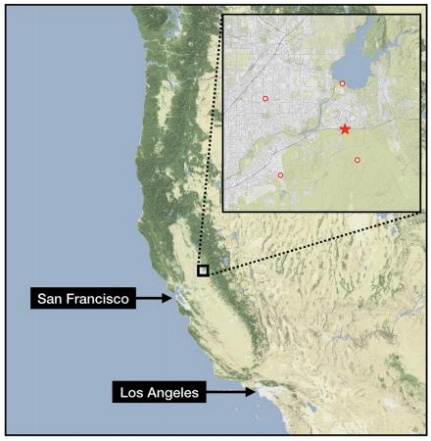


Figure 18: The map of California and in which area of California the measurement instruments are located. The exact location is the red cross. Obtained from (Carreira Pedro et al., 2019)**.**

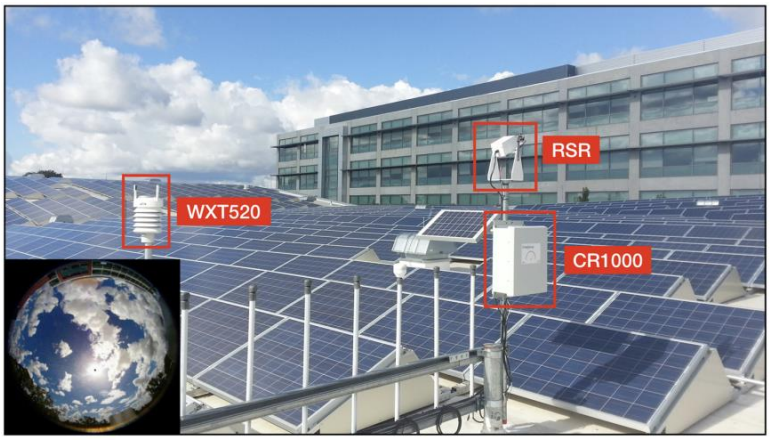


Figure 19: The setup of the measurement instruments used for acquiring the dataset is shown. Obtained from (Carreira Pedro et al., 2019)**.**

The used dataset in this study comprises ASI images, measured GHI, clear sky GHI, and meteorological data with a 1-minute interval. Temporal alignment was achieved by connecting the timestamps of the data points. Besides temporal alignment, the data required pre-processing to make it suitable for deep learning applications. After pre-processing, the data needed conversion to a dataset with correct sequencing for input and output data.

3.1.1 All Sky Images

The original All-Sky image has a resolution of 1537x1537 pixels, where each image consists of 3 channels (RGB), which resulted that a single image is stored as a 3D matrix with a width of 1537 pixels, a height of 1537 pixels, and a depth of 3 color channels (1537x1537x3). One image consists of 7,087,107 values, requiring a large amount of storage. Processing such high-resolution images results in many convolutional multiplications to extract valuable information, which results in longer training times. Besides, loading many images into Python would require large amounts of random-access memory (RAM). A workaround would be.

- allocating more memory.
- working with a smaller sample.
- using a computer with more memory.
- changing the data format.
- or using a big data platform.

Since the first three solutions require extensive additional hardware and software complexity, a smaller sample is used. The original 1537x1537x3 image is downscaled to 64x64x3, thereby reducing the required memory by a factor of 24^2. A comparison between the original and downsized sample is illustrated in Figure 20.

Besides downscaling, the image pixel values have also been normalized by dividing each pixel value by 256. Resulting in a matrix with values between 0 and 1, which are more efficiently handled by the deep learning algorithm.

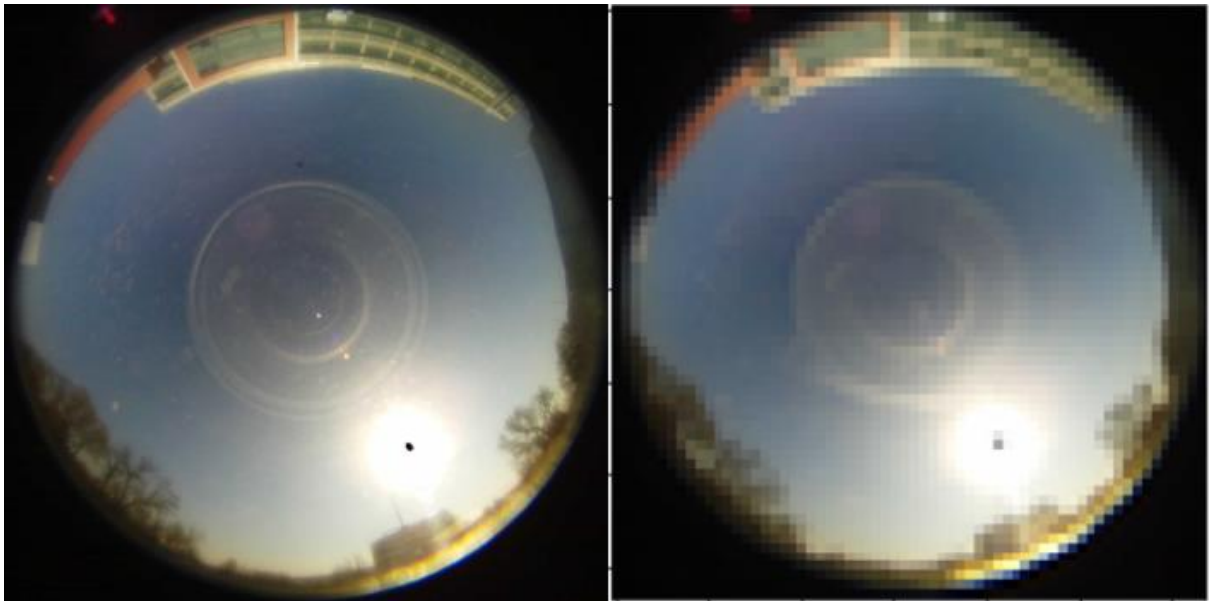


Figure 20: The original image on the left and the downscaled image on the right are illustrated.

3.1.2 Measured GHI & Clear sky GHI

The 1-minute clear sky GHI was obtained using the 'get_clearsky()' function included in the pvlib Python library (F. Holmgren et al., 2018). This function calculates the solar position, extra-terrestrial irradiance, airmass, atmospheric pressure and uses it to return the clear sky GHI in W/m^2. The implemented clear sky model used in the pvlib library is the Ineichen & Perez model, elaborated in Section 2.2.1.

The measured GHI ($GHI^{measured}$) and clear sky GHI ($GHI^{clear\ sky}$) were both normalized by dividing the irradiances by the same value, called the irradiance normalization factor (I_{norm}). I_{norm} is the highest value present in all the observed measured GHIs and clear sky GHIs of the training dataset, as depicted in equation 3-1. In equation 3-1 't' denotes a data point and 'T_{total}' denotes the total amount of measured GHI data points.

$$I_{norm} = \max\left(GHI^{measured}_{t=0,\dots,T_{total}}, GHI^{clear\ sky}_{t=0,\dots,T_{total}}\right) \quad \textbf{3-1}$$

Using the I_{norm}, the measured GHI and clear sky GHI were normalized using equations 3-2 and 3-3, respectively. The normalized measured GHI and normalized clear sky GHI are denoted with $GHI_{t,norm}^{measured}$ and $GHI_{t,norm}^{clear\ sky}$ respectively.

$$GHI_{t,norm}^{measured} = \frac{GHI_t^{measured}}{I_{norm}} \quad \textbf{3-2}$$

$$GHI_{t,norm}^{clear\ sky} = \frac{GHI_t^{clear\ sky}}{I_{norm}} \quad \textbf{3-3}$$

3.1.3 Meteorological Data

The meteorological data was processed in several ways. First, the NaN (Not a Number) values were replaced with the previous non-NaN measured value. Subsequently, the wind variables (wind speed and direction) were used to create wind vectors, which are more efficiently interpreted within deep learning models. The wind direction in degrees is converted to radians, which are subsequently used to calculate the wind in the X and Y-axis direction using equations 3-4 and 3-5, respectively. After conversion, the wind vectors were in the unit of ms^{-1}. In equation 3-4 and 3-5 the wind direction is denoted with 'ϕ' and the windspeed with 'V'. The subscript 't' denotes a certain data point within the dataset and the subscripts 'wx' and 'wy' denote the absolute wind speed in the X-axis and Y-axis direction, respectively.

$$V_{t,wx} = V_t * \cos(\phi_t(radians)) \quad \textbf{3-4}$$

$$V_{t,wy} = V_t * sin(\phi_t(radians)) \quad \textbf{3-5}$$

Subsequently, all the meteorological variables are normalized using equation 3-6. In equation 3-6 'Var' stands for one of the meteorological variables and the subscript 'a' denotes which meteorological variable is used. In total 6 meteorological variables are used, namely:

- Pressure.
- Ambient Temperature.
- Relative Humidity.
- Precipitation.
- Absolute wind speed in X-direction.
- Absolute wind speed in Y-direction.

The subscript 'norm' stands for normalized and the subscript 't' stands for a certain data point within the dataset. Min(Var_a) depicts the minimum value of the variable present within the dataset, and max(Var_a) is the maximum value of the variable present within the dataset. Applying normalization ensures that values are in between 0 and 1.

$$Var_{a,norm,t} = \frac{Var_{a,t} - \min(Var_{a,t=0,\dots,T})}{\max(Var_{a,t=0,\dots,T}) - \min(Var_{a,t=0,\dots,T})} \quad \textbf{3-6}$$

3.1.4 Dataset Merging

A dataset was merged to train the DL models from the normalized and pre-processed time series of ASI images, measured GHI, modelled clear sky GHI and meteorological data. The time series contained gaps since data is only acquired during daytime. The time series gaps result in errors when taking sequences from the time series. The time series gap problem is resolved by splitting the dataset into separated days using the clear sky GHI. The logic used is that if the clear sky GHI is decreasing and starts to increase again, it means that a day has

ended or started. The logic is depicted in equation 3-7 and is visualized in Figure 21. From each separate day, sub-time series are made and used to create sub-datasets, which were concatenated to create a single dataset.

$$\text{if } GHI_{t-1}^{clear\ sky} \geq GHI_{t}^{clear\ sky} \text{ and } GHI_{t}^{clear\ sky} \leq GHI_{t+1}^{clear\ sky} \rightarrow \text{Start or New Day} \quad \textbf{3-7}$$

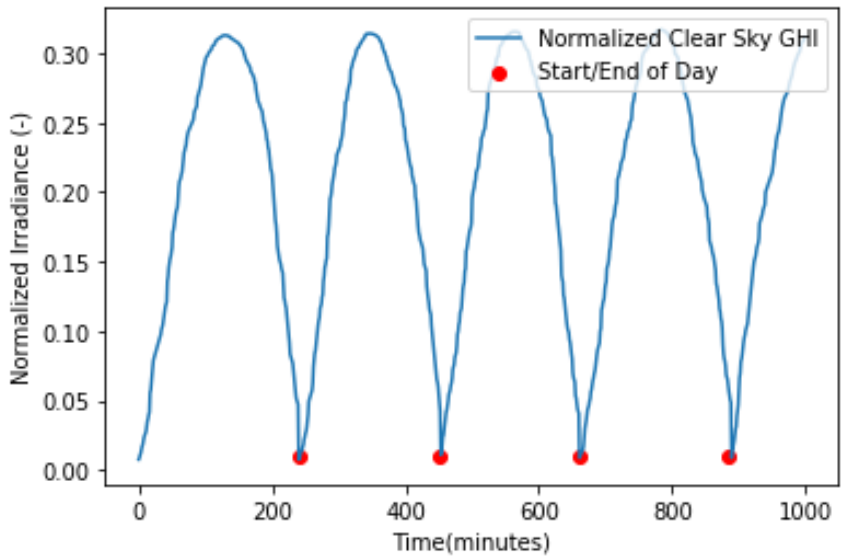


Figure 21: The figure shows how the timeseries dataset is split using the clear sky GHI.

Depending on the deep learning model that is trained a dataset has been created. Each dataset follows the logic explained above. The main distinction between created datasets is the Static Datasets (SD) and the Dynamic Datasets (DD). The SD takes a single point in time as an input, and the DD takes a sequence of the time series as input. The combination of the measured GHI, clear sky GHI, and meteorological data is referred to as 'auxiliary data'. Training a DL model requires that inputs be labelled with the outputs. The inputs given into the DL models is a single time step (t=0) or a sequence of the last three timesteps (t=-2, -1, 0). The output is a sequence of measured GHI from time step 0 to 21. The difference between inputs for the SD and DD models is depicted below.

Static Dataset
Input:
- All-Sky Image(t) : t=0
- Auxiliary Data(t) : t=0

Output
- Measured GHI(t) : t=0,…,21.

Dynamic Dataset
Input:
- All-Sky Image(t) : t=-2,…,0.
- Auxiliary Data(t) : t=-2,…,0.

Output
- Measured GHI(t) : t=0,…,21.

3.1.5 Training, Validation & Testing

The deep learning models have been trained, validated, and tested on different parts of the total dataset. The entire dataset comprises data of the years 2014, 2015, and 2016.
Training: All the models were trained on 40% of the data of 2014 and 40% of the data of 2015. Comprising in total of 211,882 datapoints. Before training, the training dataset was randomly shuffled.

Validation: All the models were validated on 5% of the data of 2014 and 5% of the data of 2015. Comprising in total of 26,485 datapoints. Before validation, the validation dataset was randomly shuffled.
Testing: All the models were tested on 100% of the data of 2016. Comprising in total of 242,038 datapoints.

The training, validation, and testing were similar for all the models. apart from certain models to investigate the model's sensitivity to the used dataset, on which is elaborated in Section 3.3. The batch size for all models used is 40 datapoints per batch.

All the described models are implemented using Keras with TensorFlow as backend. The used computer is an Aspire A515-54 with 8 GB RAM and an i5-8265U processor without a suitable graphical processing unit (GPU) to speed up the training process.

Depending on the learning curves of the models, the training time for the models differed. When the model performance on the training dataset outperformed the model performance on the validation set, one more epoch is trained after which training is stopped. However, after training the learning curves were evaluated, and if the model still improved in subsequent epochs on the validation and training dataset, the training process was continued until no further improvement was achieved. The required training time was approximately 6 hours per model, depending on the performance on the training dataset and validation dataset. Typical learning curves for trained models in this study are visible in the Appendix.

3.1.6 Loss Function & Activation Function

Besides the dataset used for training, validating, and testing the DL models, there are more parameters to set, called hyperparameters. Hyperparameters describe the architecture of the DL models, like the number of layers and the number of units in each layer. Optimization of hyperparameters is not done in this study since it is a time-consuming process, especially on a computer without NVIDIA GPU (Nvidia Corporation, 2015). The hyperparameters are based on a previous study that did hyperparameter optimization for a DL model that uses ASI images for GHI prediction (Paletta et al., 2021). The different layers and the number of units within each layer are described in Section 3.2. Two important DL decision choices are described below, namely the loss function and activation function.

The internal weights within the model are updated to increase the performance of the model. The model performance is assessed using the loss function MSE as depicted in equation 3-8. In equation 3-8 '$\hat{Y}$' is the predicted GHI by the model and 'Y' is the measured GHI.

$$MSE(\{\hat{Y}\},\{Y\}) = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_t - Y_t)^2 \qquad \textbf{3-8}$$

The MSE loss function is chosen since it penalizes larger errors more than minor errors due to the squared difference between the predicted and measured values. Within the field of short-term irradiance forecasting, the main problem is to predict partially cloudy sky conditions, which result in large fluctuations in short time horizons and therefore large errors. Clear sky conditions often result in minor errors, due to the constant and easy to predict behavior. Besides, short-term predictions for clear skies are already well described by clear sky models. To mitigate the large error outliers and increase performance during the partial cloudy sky conditions, the MSE is used. On top of that, it has already been proven that MSE loss functions show promising results with the use of ASI images for short-term GHI predictions and outperform other loss functions (Paletta et al., 2021).

A more comprehensive explanation about activation functions and how they work are given in Section 2.4. The used activation function for the hidden layers within the DL models is the ReLU activation function for its computationally efficiency and it does not suffer from the vanishing gradient problem.

3.1.7 Performance Evaluation

The DL models are trained using the loss function MSE, but the output unit of the MSE is W^2/m^4. Since W^2/m^4 is not the standard International System of Units (SI) and a clear metric to understand how well the model performs, a different error metric has been used. The standard SI is obtained by using the MAE as depicted in equation 3-9. Besides the MAE also the RMSE error is used depicted in equation 3-10.

$$MAE(\{\hat{Y}\},\{Y\}) = \frac{1}{n}\sum_{i=1}^{n}\left|(\hat{Y}_t - Y_t)\right| \quad \textbf{3-9}$$

$$RMSE(\{\hat{Y}\},\{Y\}) = \sqrt{\frac{\sum_{i=1}^{n}(\hat{Y}_t - Y_t)^2}{n}} \quad \textbf{3-10}$$

The MAE and RMSE are chosen as main error metric due to the good generalization properties when comparing different models and datasets. Besides, those are the most used error metric within solar irradiance forecasting performance evaluation and is the standard SI unit. (Yang et al., 2020)

Besides the MAE and RMSE also the normalized MAE (nMAE) and normalized RMSE (nRMSE) are calculated. The nMAE and nRMSE give the relative difference in predictions in percentages instead of the actual error in W/m^2. Multiple ways exist to normalize the MAE and RMSE, and in this study, the mean of all the GHI measurements is used to obtain the nMAE and nRMSE, as depicted in equation 3-11 and 3-12 respectively. The values of '$GHI_{measured}^{2016}$' are all the measured GHI of 2016 and the mean GHI of 2016 was 436.92 W/m^2. The nMAE and nRMSE can be used to compare the performance with other papers that have different mean values of the GHI.

$$nMAE = \frac{MAE}{GHI_{measured}^{2016}} \quad \textbf{3-11}$$

$$nRMSE = \frac{RMSE}{GHI_{measured}^{2016}} \quad \textbf{3-12}$$

Since the model predicts a forecast horizon, the performance of the model for each timestep is different. The performance of the DL model has been evaluated over the whole horizon and for each separate timestep. The MAE for the entire forecast horizon is calculated using equation 3-13 and the MAE for each separate timestep is calculated using equation 3-14. Within equations 3-13 and 3-14 't' denotes a single prediction and 'T_{total}' the total amount of predictions made. 'τ' denotes separate timesteps within the forecast horizon between 0 and 21 minutes and T_τ denotes the total amount of timesteps within the forecast horizon.

$$MAE = \frac{1}{T_\tau * T_{total}}\sum_{\tau=1}^{T\tau}\sum_{t=1}^{T}\left|\hat{Y}(\tau,t) - Y(\tau,t)\right| \quad \textbf{3-13}$$

$$MAE_\tau = \frac{1}{T_{total}}\sum_{t=1}^{T}\left|\widehat{Y_\tau(t)} - Y_\tau(t)\right| \quad \textbf{3-14}$$

The forecast skill (FS) assesses the performance of models relative to each other. The models are compared using MAE over the whole forecast horizon. The MAE of the model that is compared with is called the baseline model (MAE_{base}) and the model to compare is called the new model (MAE_{new}), as is depicted in equation 3-15.

$$FS = \frac{MAE_{base} - MAE_{new}}{MAE_{base}} \quad \textbf{3-15}$$

Besides evaluating the model performance over the forecast horizon, the model's performance is assessed during different sky conditions. In this study, the sky conditions have been classified using a similar method presented in (Hartmann, 2020), This method classifies sky conditions based on the clearness index (CI) and variability index (VI). The calculation of the VI differs slightly from (Hartmann, 2020) since this study calculated the 1-D Euclidean distance and within (Hartmann, 2020) the 2-D Euclidean distance is used. In practice this results in minimal differences, since the differences in GHI measurements are significantly bigger than the differences between timesteps. CI is calculated using equation 3-16, and VI is computed using equation 3-17. In equations 3-16 and 3-17 $GHI_{t,\tau}$ stands for the Global Horizon Irradiance for a specific datapoint (t) at a time within the forecast horizon (τ) of the prediction.

$$CI_t = \frac{\sum_{i=0}^{21} GHI_{t,\tau(i)}^{Measured}}{\sum_{i=0}^{21} GHI_{t,\tau(i)}^{Clear\ Sky}} \quad \textbf{3-16}$$

$$VI_t = \frac{\sum_{i=0}^{21} \left|(GHI_{t,\tau(i)}^{Measured} - GHI_{t,\tau(i+1)}^{Measured})\right|}{\sum_{i=0}^{21} \left|(GHI_{t,\tau(i)}^{Clear\ Sky} - GHI_{t,\tau(i+1)}^{Clear\ Sky})\right|} \quad \textbf{3-17}$$

Based on the CI and VI for a prediction horizon, the conditions are classified into the five classes: clear, overcast, mild, moderate, and high denoted in Equations 3-18, 3-19, 3-20, 3-21, and 3-22, respectively.

$$Clear: \begin{cases} CI \geq 0.5 \\ VI < 2 \end{cases} \quad \textbf{3-18}$$

$$Overcast: \begin{cases} CI < 0.5 \\ VI < 2 \end{cases} \quad \textbf{3-19}$$

$$Mild: 2 < VI < 5 \quad \textbf{3-20}$$

$$Moderate: 5 \leq VI < 10 \quad \textbf{3-21}$$

$$High: 10 \leq VI \quad \textbf{3-22}$$

3.2 Model Architectures

3.2.1 Baseline Model

The baseline model used in this study is the persistence model, which takes the GHI at t=0 and predicts that the GHI does not change in the whole predicting interval. The formula is depicted in Equation 3-23.

$$GHI(t = 0, \dots, 21) = GHI^{measured}(t = 0) \quad \textbf{3-23}$$

The persistence model is used to compare the performance of the DL models and is known as a hard-to-beat model due to the fluctuating nature of the irradiance (Pothineni et al., 2019).

3.2.2 Convolutional Neural Network

Extracting meaningful information from the ASI images is achieved using convolutional blocks. The total convolutional network consists of 4 CNN blocks, where the filter size decreases, and number of filters increases. The last layer is a Flatten layer to reduce the 3d dimensionality to 1D. The flattened output from the CNN blocks is subsequently fed into 2 dense layers to allow the extraction of more complexity from the image.

The image of 3x64x64 is fed to the following CNN blocks and dense layers chronologically. In Figure 22 the convolutional neural network architecture is depicted including the intermediate output and input sizes of each layer.

0. The input layer is the image input with a size of 3x64x64.
1. Convolutional block with 20 filters of a size of 7x7 → 3x64x64 to 20x29x29.
2. Convolutional block with 30 filters of a size of 5x5 → 20x29x29 to 30x12x12.
3. Convolutional block with 40 filters of a size of 3x3 → 30x12x12 to 40x5x5.
4. Convolutional block with 50 filters of a size of 2x2 → 40x5x5 to 50x2x2.
5. Flatten layer → 50x2x2 to 1x200.
6. Dense Layer with 512 units → 1x200 to 1x512.
7. Dense Layer with 64 units → 1x512 to 1x64.

Hyperparameter tuning could potentially result in a marginal improvement, but is often not comparable to different deep learning model structures (Kong, 2017) and is therefore not conducted. This holds for all the deep learning models used in this research.

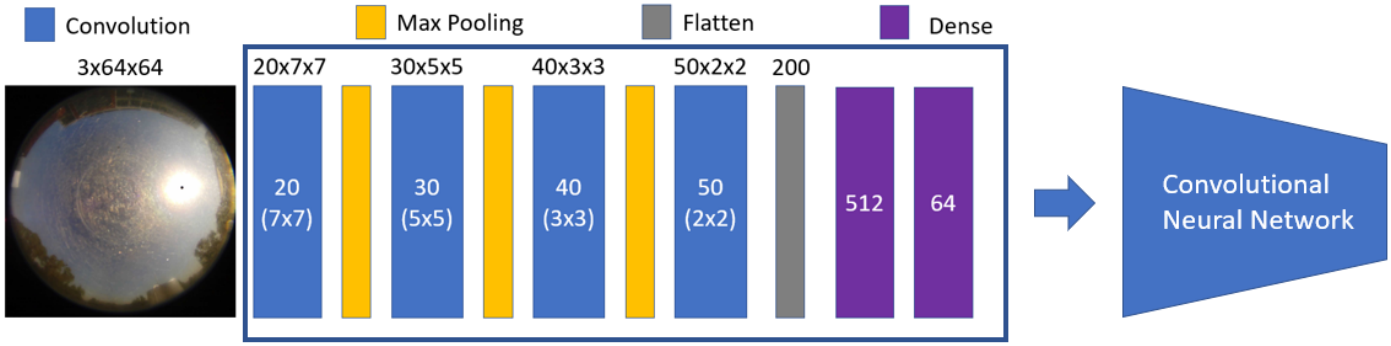


Figure 22: The convolutional neural network used for the deep learning model to extract image features is depicted.

3.2.3 Dense Neural Networks

Besides the images input, also 1-D inputs are used, which are best interpreted using dense neural networks. In this study the dense neural networks can be distinguished in two parts, namely the auxiliary data encoder and the prediction decoder. In this section those two neural networks will be discussed separately.

The auxiliary data encoder is used to extract important information from the auxiliary data consisting of the measured GHI, clear sky GHI and meteorological data. To achieve this, a dense neural network of 2 layers consisting of 18 units each is used, as depicted in Figure 23. The layers are chronologically depicted below.

0. The input layer is the auxiliary data vector with a size of 1x8.
1. Dense layer with 18 units → 1x8 to 1x18.
2. Dense layer with 18 units → 1x18 to 1x18.

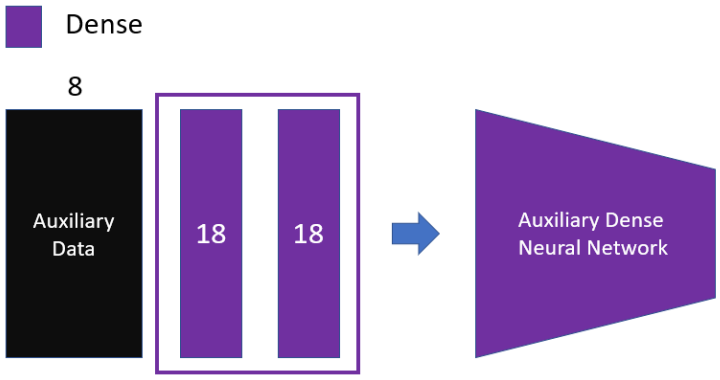


Figure 23: The auxiliary dense neural network is depicted.

The auxiliary data encoder output and convolutional neural network have the same dimensions (1-D) and are concatenated to a single vector. This vector is a representation of the image and auxiliary data combined and is used as input for the prediction decoder. The auxiliary data output is a vector of the size 1x18, and the output vector of the CNN blocks is 1x64, resulting in a vector of 1x82. This vector is used in the prediction decoder to map this vector representation of the image and auxiliary data to the GHI prediction.

In the prediction decoder the concatenated vector is used to make the final prediction. The prediction decoder consists of 2 dense layers that feed into 22 separate dense layers with a single unit that represent the prediction horizon of 0 to 21 minutes. The 2 dense layers have 64 and 32 units respectively, and the final 22 separate dense layers each consists of a single unit as is depicted in Figure 24. The layers are chronologically depicted below.

0. The input layer is the concatenated layer of 1x82.
1. Dense layer with 64 units → 1x82 to 1x64.
2. Dense layer with 32 units → 1x64 to 1x32.
3. The previous layer output is used in 22 separate dense layers with 1 unit → 1x32 → 1x1 (22x).

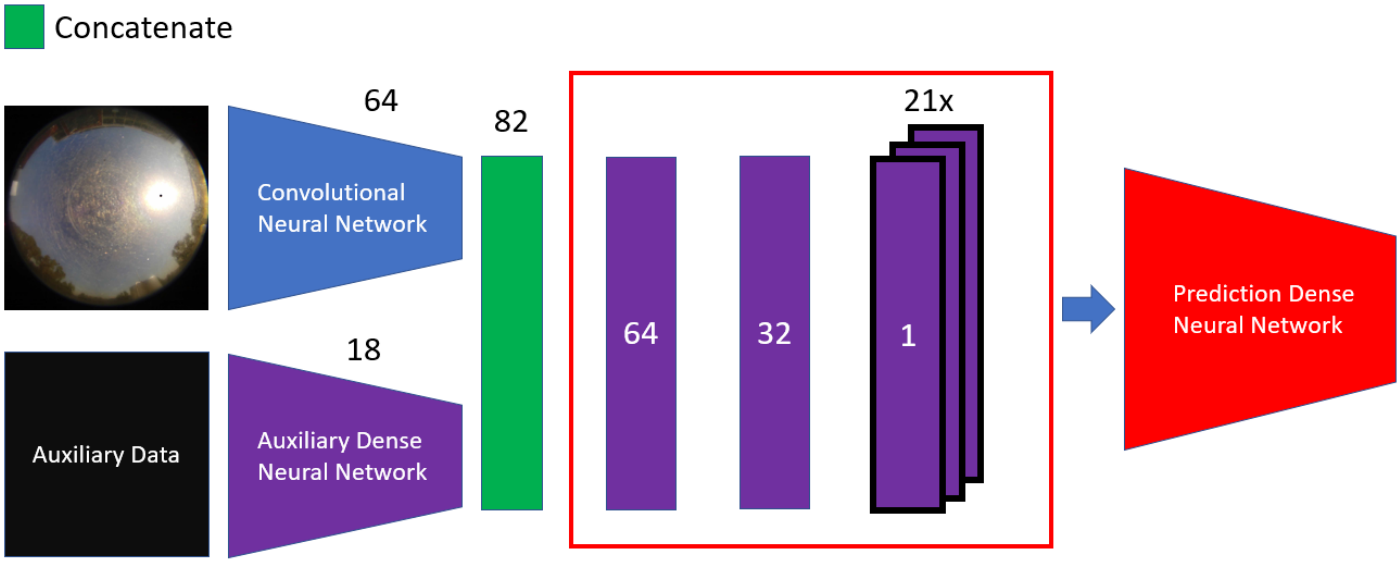


Figure 24: The architecture of the final prediction dense neural network architecture is illustrated.

An overall model, from image and auxiliary data input to the irradiance prediction can be simplified according to Figure 25. This model will be referred to as the 'CNN-Main' Model.

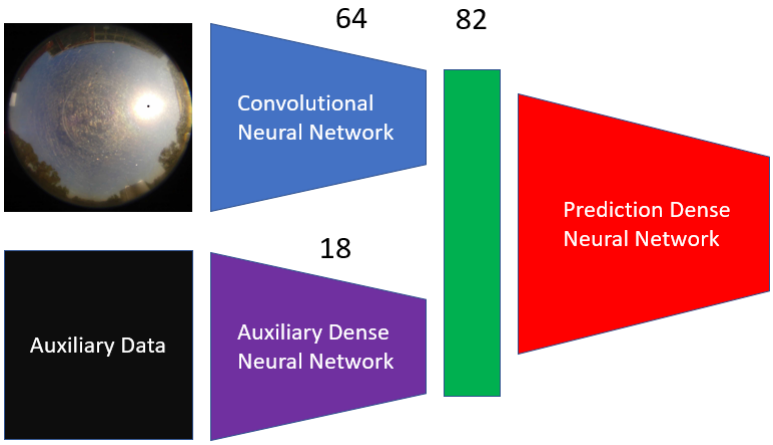


Figure 25: The overall architecture of the deep learning model to make irradiance prediction. This model architecture will be referred to as the 'CNN-Main' Model.

3.2.4 Recurrent Neural Networks

The architecture depicted in Figure 25 shows how a single point in time can be used to predict the irradiance. However, clouds and other variables are dynamic and change over time and to make an accurate prediction into the future, the dynamics of the input need to be considered. A sequence of past datapoints can be used to investigate the dynamics of the system. To handle sequence data, RNNs have proven to be efficient in extracting patterns (TensorFlow, 2021a). The used RNN model has a similar structure as the CNN-Main model prior to the RNN layer as depicted in Figure 23. Instead of taking a single data point, a sequence of the last 3 data points is used as input. Each input is separately encoded within the convolutional neural network and auxiliary dense neural network to create a vector of 82x1. After processing the three separate inputs, three vectors of 82x1 are obtained, forming a sequence representation of the input sequence data with a size of 3x82x1. Since the data is now in sequence form, the RNN can use this data as input as is chronologically depicted below and is illustrated in Figure 26. This model will be referred to as the 'RNN-Main' Model. The addition of the RNN layer to the RNN-Main model is the main difference with the CNN-Main model. Furthermore, does the RNN-Main model take a sequence of the last 3 timesteps as input, where the CNN-Main model only uses the last timestep as input.

0. The input layer is the sequence of concatenated layers of 3x82x1.
1. RNN layer with 100 units → 3x82 → 1x100.

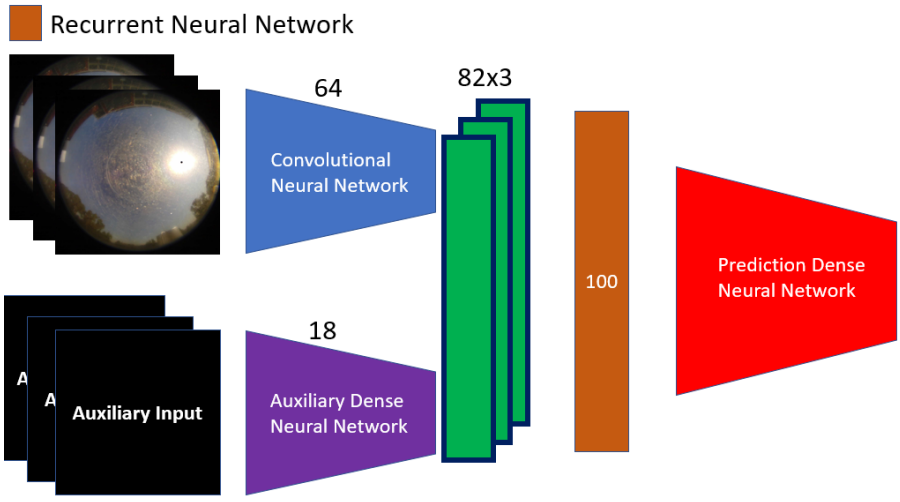


Figure 26: The DL RNN model using the blocks depicted in Figure 25 is illustrated. This model will be referred to as the 'RNN-Main' Model.

The RNN-Main and CNN-Main model are the main models for this study, but modifications to the input data and output data have been done to assess the sensitivities of the models. The number of trainable weights for each of the layers with the DL models is depicted in the Appendix.

3.3 Model Sensitivity Analysis

The sensitivity of the DL model is evaluated using multiple modifications to the input and output of the dataset. Subsequently, the model performances are compared to investigate the sensitivity to specific inputs and outputs. In Table 3, the variety of trained models is shown, including input, output, and sequence of data. All models starting with 'CNN' make use of the CNN model depicted in Section 3.2.2, and the models beginning with RNN make use of the model described in Section 3.2.4.

Table 3: The different trained models and the input variables including the input sequence, and the models output are depicted. The abbreviation 'Img' means ASI Image, $GHI^{measured}$ the measured global horizontal irradiance, $GHI^{clear\ sky}$ the global horizontal clear sky irradiance, and METEO the meteorological data input.

Model	Input	Sequence	Output
CNN-Main	Img, $GHI^{measured}$, $GHI^{clear\ sky}$, METEO	Input(t) : t=0	GHI($t = 0, \dots, 21$)
RNN-Main	Img, $GHI^{measured}$, $GHI^{clear\ sky}$, METEO	Input(t) : t=-2,…,0	GHI($t = 0, \dots, 21$)
CNN-Main (2014)	Img, $GHI^{measured}$, $GHI^{clear\ sky}$, METEO	Input(t) : t=0	GHI($t = 0, \dots, 21$)
CNN-Main (No Meteo)	Img, $GHI^{measured}$, $GHI^{clear\ sky}$	Input(t) : t=0	GHI($t = 0, \dots, 21$)
CNN-Main (No $GHI^{clear\ sky}$)	Img, $GHI^{measured}$, METEO	Input(t) : t=0	GHI($t = 0, \dots, 21$)
CNN-Main (No $GHI^{measured}$)	Img, $GHI^{clear\ sky}$, METEO	Input(t) : t=0	GHI($t = 0, \dots, 21$)
CNN-Main (No Img)	$GHI^{measured}$, $GHI^{clear\ sky}$, METEO	Input(t) : t=0	GHI($t = 0, \dots, 21$)
CNN-Main (Seq- $GHI^{measured}$)	Img, $GHI^{measured}$, $GHI^{clear\ sky}$, METEO	Input(t) : t=0 GHI(t=-9, …, 0)	GHI($t = 0, \dots, 21$)
CNN-Main-6min	Img, $GHI^{measured}$, $GHI^{clear\ sky}$, METEO	Input(t) : t=0	GHI($t = 0, \dots, 6$)
CNN-Main-11min	Img, $GHI^{measured}$, $GHI^{clear\ sky}$, METEO	Input(t) : t=0	GHI($t = 0, \dots, 11$)

The DL model architectures of the CNN-Main and RNN-Main are depicted in Section 3.2. The other DL models have similar structures as these models, but slight modifications are performed to match the input and output data. The modification made to the architectures are depicted below for each of the models.

- **CNN-Main(No Meteo):** has an auxiliary data input of 1x2 instead of 1x8 since the meteorological variables are excluded.
- **CNN-Main(No $GHI^{clear\ sky}$):** has auxiliary data input of 1x7 since Clear Sky GHI is not used as input.
- **CNN-Main(No $GHI^{measured}$):** has auxiliary data input of 1x7 since the measured GHI is not used as the input.
- **The CNN-Main(No Img):** excludes the convolutional neural networks and only uses the auxiliary data to predict the GHI.
- **CNN-Main (Seq-GHI):** Has auxiliary data input of 1x17 since the last 10 GHI sequence is used instead of only the last GHI.
- **CNN-Main-6min:** In the last layers of the prediction neural network, only seven dense layers with a single unit are used instead of 22 single unit layers to match the output label.
- **CNN-Main-11min:** In the last layers of the prediction neural network, only 12 dense layers with a single unit are used instead of 22 single unit layers to match the output label.
- **CNN-Main (2014):** Like the CNN-Main model but trained solely on the data of 2014.

All the models were trained on a combination of the data from 2014 & 2015 and tested on the year 2016 unless stated differently. Each DL model was trained three times with different seeds. Programs like Python make use of pseudo randomness and the seeds initialize the pseudo randomness. The three same seeds for each separate model are used to reduce stochasticity in model comparisons and make the models reproducible.

4 Results

4.1 CNN-Main and RNN-Main Forecasting Performance

The performance of the CNN-Main and RNN-Main models for the 21-minute ahead GHI forecast horizon is depicted in Table 4. It is visible that the CNN-Main model has the best performance, outperforming the persistence model with 24.8% on average over the whole forecast horizon of 0 to 21 minutes. The RNN-Main model also outperforms the persistence model, but slightly less than the CNN-Main model, namely with 19.4%. Despite using a sequence of 3 data points as input into the RNN-Main model, it does not outperform the CNN-Main model, which uses single data point as input. Since the CNN-Main model has the best performance, the emphasis will be on the CNN-Main model for the following sections.

Table 4: The performance of the CNN-Main & RNN-Main models relative to the persistence model are depicted.

Model	MAE(W/m^2)	RMSE(W/m^2)	nMAE	nRMSE	Forecast Skill
Persistence	42.01	82.89	9.6%	19.0%	-
CNN-Main	31.61	67.11	7.2%	15.4%	24.8%
RNN-Main	33.88	70.96	7.6%	16.2%	19.4%

In Figure 27, the average MAE for each timestep of the CNN-Main model and persistence model are depicted. It is visible that the persistence model slightly outperforms the CNN-main model in the initial three timesteps and after the three timesteps the CNN-Main model consistently outperforms the persistence model.

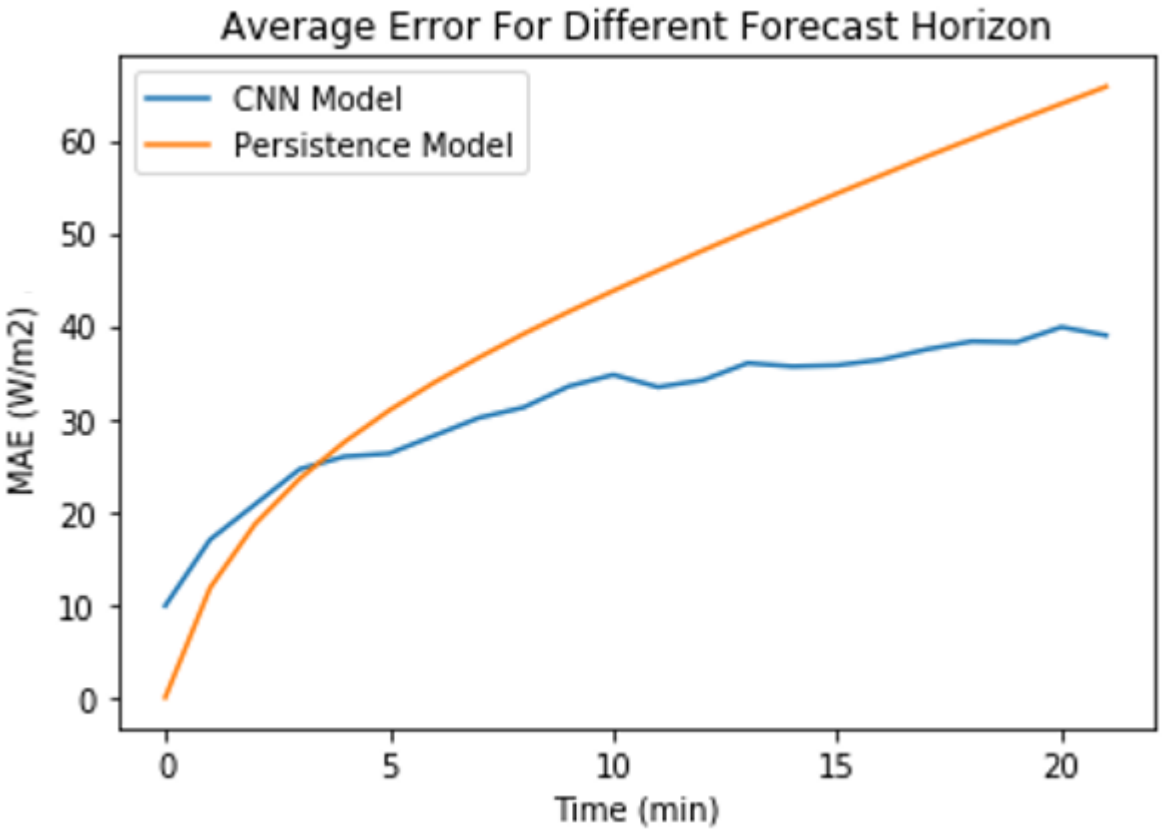


Figure 27: The Average MAE of the CNN-Main model and persistence model.

In Figure 28, a histogram of the average MAE occurrences for the whole forecast horizon is depicted, where each bar represents a step of 40 W/m^2. It is visible that the far majority of the average MAE occurrences is

within the 0-40 W/m^2 bar and that the error occurrence is decreasing rapidly in the higher W/m^2 bars. It is also visible that the average MAE occurrence of the CNN-Main model and persistence model have a somewhat similar distribution. Still, the persistence model has slightly more errors within the 40-80 W/m^2 segment, where the CNN-Main model has marginally more errors within the 0-40 W/m^2 segment. Since the error bars in Figure 28 are too small to interpret for larger errors, the total occurrence of errors is depicted in Table 5. It is visible that the largest occurring errors within the CNN-main model are in the range of 600-700 W/m^2, while the persistence model holds errors up to the range of 800-900 W/m^2. This shows that the CNN-Main model, besides having an overall improved performance, also significantly reduces the number of outliers of average MAE errors.

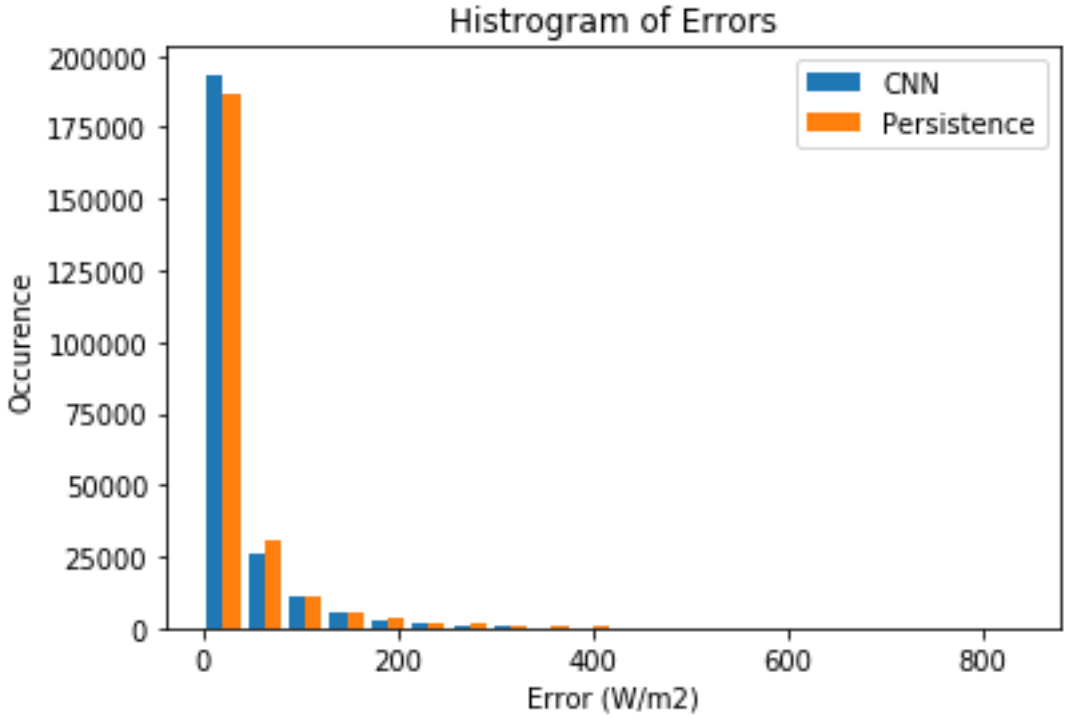


Figure 28: A histogram of the occurrences of the average MAE over the forecast horizon of 0 to 21 minutes of the CNN-Main and persistence models. Each bar represents a step of 40 W/m^2.

Table 5: The occurrences of the average MAE over the forecast horizon of 0 to 21 minutes of the CNN-Main and persistence models segmented in steps of 100 W/m^2

Error Segment	Persistence	CNN-main
0-100 W/m^2	221804	224064
100-200 W/m^2	13924	13916
200-300 W/m^2	4055	3154
300-400 W/m^2	1478	741
400-500 W/m^2	499	141
500-600 W/m^2	203	18
600-700 W/m^2	61	4
700-800 W/m^2	15	0
800-900 W/m^2	4	0

4.2 Deep Learning Model Performance Sensitivity to the Dataset

The sensitivity of the CNN-Main model to the used dataset is evaluated in two ways. First the sensitivity of the performance of the CNN-Main model is tested using datasets comprised of different years. A dataset comprised of a combination of data of 2014 and 2015 is compared with a dataset that is solely comprised of

data from 2014. Secondly the performance of the CNN-Main model with respect to different sizes of data within the dataset is evaluated.

4.2.1 Dataset Variability Sensitivity

Two similar CNN-Main models have been trained where one model has been trained on data of 2014 and the other model on a combination of the data of 2014 and 2015. Both models were trained on approximately the same amount of data points (±2%). In Table 6, the performances of the CNN-Main models trained on different datasets is depicted. From Table 6 follows that combining data from 2014 and 2015 performs 7.7% better compared to using a dataset that is solely comprised of data from 2014.

The increased performance can be explained with the fact that within deep learning the dataset must represent the situation that needs to be predicted. When observing weather patterns over a year, the same seasons can be observed, but every year differs. Some years have more cloudy days compared to other years. Ideally, the training dataset contains all possible situations that can occur. Unfortunately, this is not the case, since the possibilities of configurations of cloud positions, cloud movement trajectories, colors, and textures are beyond imagination. From Table 6, it follows that the year 2014 is a lesser representation for the year 2016 compared to a combination of the years 2014 and 2015.

Table 6: The influence of using a dataset comprised of solely 2014 compared to a dataset comprising the years 2014 and 2015 on the CNN-Main model performance is depicted.

Model	MAE(W/m^2)	nMAE	Relative Improvement
CNN-Main	31.61	7.2%	-
CNN-Main (2014)	34.03	7.8%	-7.7%

4.2.2 Dataset Size Sensitivity

Evaluation of the sensitivity of the CNN-Main model to the amount of training data is obtained by training the CNN-Main model for different sized datasets. The original dataset consists of 211,882 data points and is considered 100%. A percentage of the data is randomly selected from this dataset, which is subsequently used to train the CNN-Main models. The amount of data used, and the performances of the CNN-Main models trained on different sized datasets are depicted in Figure 29 and Table 7.

Figure 29 and Table 7 show that more data results in better model performance. Although, using only 50% of the data results in a minor performance decrease of 4.7%. At 20% data usage, the performance of the model starts to worsen considerably more and using less than 10% of the data, resulting in a significant performance decline. Within the field of DL, it is known that more data results in better model performance (Ng, 2020), which correlates with the findings in Figure 29 and Table 7.

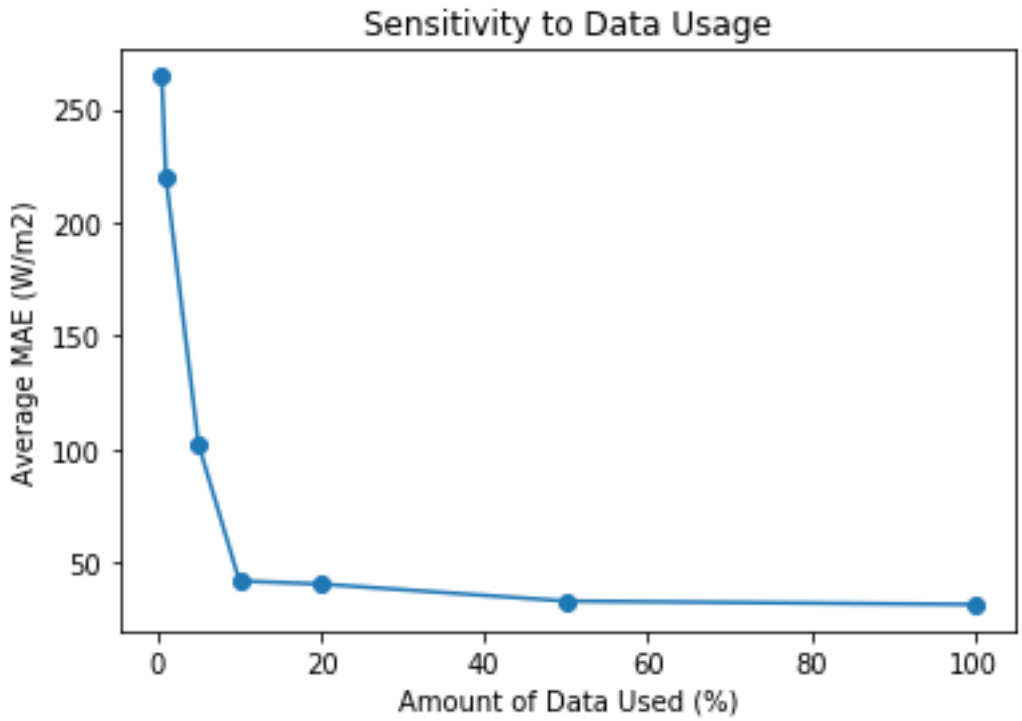


Figure 29: The influence of the amount of data on the performance of the CNN-Main model.

Table 7: The influence of the amount of data on the performance of the CNN-Main model.

Amount of datapoints	Amount of data	MAE(W/m^2)	Relative Improvement
211,882	100%	31.61	-
105,941	50%	33.09	-4.7%
42,376	20%	40.59	-28.4%
21,188	10%	41.95	-32.7
10,594	5%	102.18	-223.3%
2,118	1%	220.32	-597.0%
1,059	0.5%	264.54	-736.9%

4.3 Forecast Horizon Sensitivity

The performance sensitivity of the CNN-Main model to different forecast horizons is tested using similar models with similar datasets trained to predict different forecast horizons. The forecast horizons considered are:

- **CNN-Main:** 0 – 21 minutes
- **CNN-Main-11min:** 0 – 11 minutes
- **CNN-Main-6min:** 0 – 6 minutes

The performance of the CNN-Main model is assessed on the same forecast horizon as for the CNN-Main-11min and the CNN-Main-6min models. Also, the MAE improvement relative to the CNN-Main model has been calculated and is depicted in Table 8.

Table 8: The performance of the CNN-Main model on different forecast horizons is compared with models that were trained to forecast smaller forecast horizons.

Model	MAE(W/m^2)	nMAE	Forecast Skill
CNN-Main ; t(0,…,11)	27.36	6.3%	-
CNN-Main-11min ; t(0,…,11)	27.27	6.2%	0.5%
CNN-Main [0-6] ; t(0,…,6)	23.06	5.3%	-
CNN-Main-6min; t(0,…,6)	22.55	5.2%	2.2%

Table 8 depicted results are the average performances over the time horizon, and it is visible that the CNN-Main models trained on smaller forecast horizons result in better performance. When looking at the performance on each timestep, it is visible that smaller forecast horizons improve relative more on the initial timesteps and worse on the later timesteps, as is depicted in Figure 30 and Figure 31.

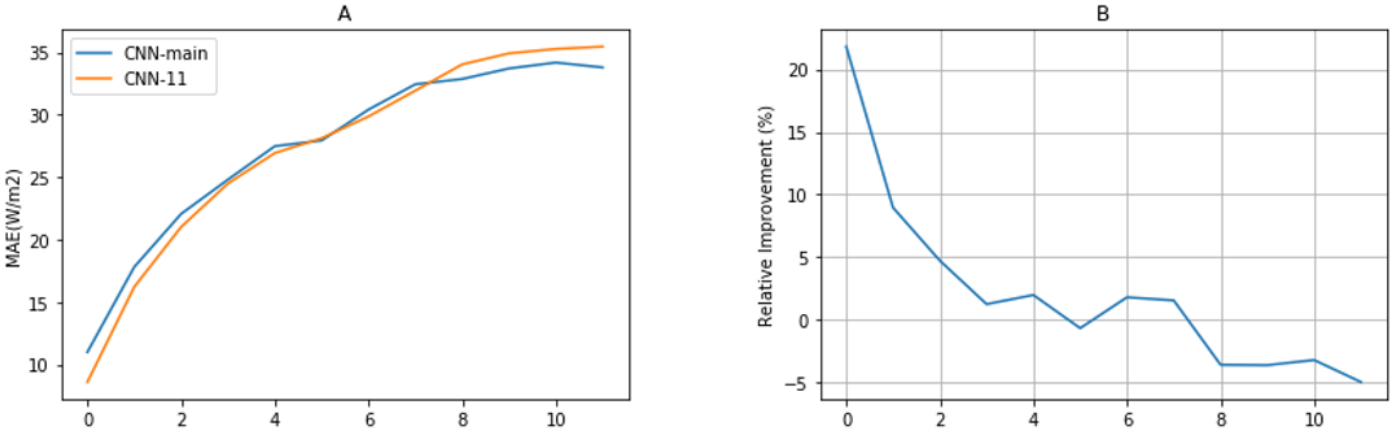


Figure 30: Shows on the left in Figure A the performance of the CNN-Main model on the first 11 timesteps and the performance of the CNN-11 model on the first 11. On the right in figure B, the relative performance of the CNN-11 model to the CNN-Main model per timestep is depicted.

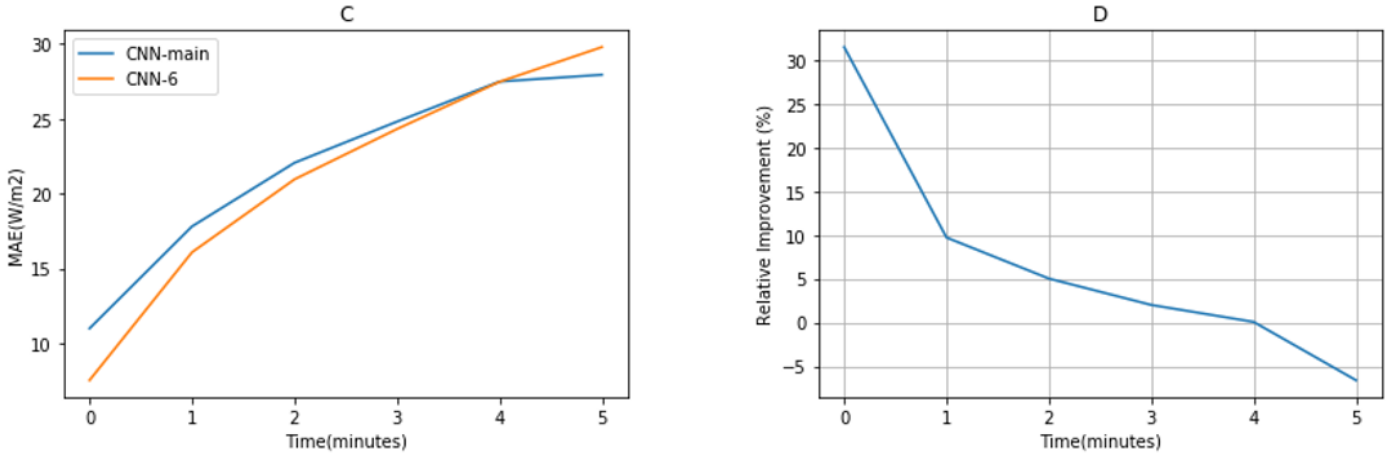


Figure 31: Shows on the left in Figure C the performance of the CNN-Main model on the first 6 timesteps and the performance of the CNN-6 model on the first 6 timesteps. On the right in figure D, the relative performance of the CNN-6 model to the CNN-Main model per timestep is depicted.

From Figure 30 and Figure 31 follows that when the forecast horizon is smaller, the overall performance increases slightly, with the most significant relative performance increase in the initial timesteps. The improved performance can be explained by looking at the deep learning algorithm MSE loss-function, which punishes more significant errors with a larger weight. Larger errors occur more frequently in later timesteps, and the CNN-Main model, therefore, prioritizes reducing errors for later timesteps compared to earlier timesteps. This explains why CNN-Main models with a shorter forecast horizon improve relatively more on initial timesteps.

4.4 Ablation Sensitivity

The sensitivity of the CNN-Main model to input variables is examined by doing an ablation study where single modalities of the multi-modal dataset were excluded (removing input variables before training the model). Similar structured models and datasets are used, except that a particular input variable is ablated (removed) from training the model. The results of the ablation study are depicted in Table 9.

From Table 9, it is visible that all inputs result in a performance improvement, except for the meteorological data input. Excluding meteorological data resulted in a marginal improvement of 1% in the model performance, showing that excluding the meteorological data is an improvement. Adding input variables into a model with no or minimal influence on what needs to be predicted causes unwanted complexity of the model, resulting possibly in worse performance. Besides the minimal influence of the meteorological data, the small improvement can also be explained due to stochasticity of DL models. Although similar seeds, datasets, and model structures are used; when the input is changed, the deep learning structure changes and results in a different training process and therefore a different model. Despite averaging the performances over three different models, stochasticity remains, and ideally, the models are trained more than three times and averaged to level out stochasticity to a greater extent.

It was expected that meteorological data would improve the performance, since meteorological data contains essential information about how the movement of clouds will develop, as shown in NWP algorithms. However, the CNN-Main model with the current structure and other data inputs cannot extract valuable information from meteorological variables to enhance the prediction performance of the CNN-main model. For the current DL model structure and data input, it is recommended to exclude meteorological data. However, it is not recommended to omit meteorological data from all future research. With the rapid development within deep learning, future, more complex models might interpret the meteorological data more effectively and utilize it to enhance performance (Schultz et al., 2021).

Table 9: The performance of the CNN-Main model compared to the CNN-Main models with the ablation of inputs is depicted.

Model	MAE(W/m^2)	nMAE	Forecast Skill
CNN-Main	31.61	7.2%	-
CNN-Main (No Image)	38.90	8.9%	-23%
CNN-Main (No Clear Sky)	36.82	8.4%	-16%
CNN-Main (No METEO)	31.15	7.1%	1%
CNN-Main (No Irradiance)	38.69	8.9%	-22%

4.5 Sky Condition Performance

Training, validation, and testing data is based on different years, and the occurrences of sky conditions differ for each year. For the years 2014, 2015 and 2016 the percentage of the occurrence of each sky condition is depicted in Figure 32. For each sky condition, three random images are depicted in Table 10. Examples of how the irradiance fluctuates during the different sky conditions is illustrated in Figure 33.

From Figure 32 it is visible that the years 2014, 2015 and 2016 are quite similar, with slight differences between occurrences of sky conditions of ± 1%. From Figure 32 it follows that within each year, roughly 50% of the measured irradiance intervals are classified as clear, and the other 50% consists of high, mild, moderate, and overcast conditions from high occurrence to low occurrence, respectively.

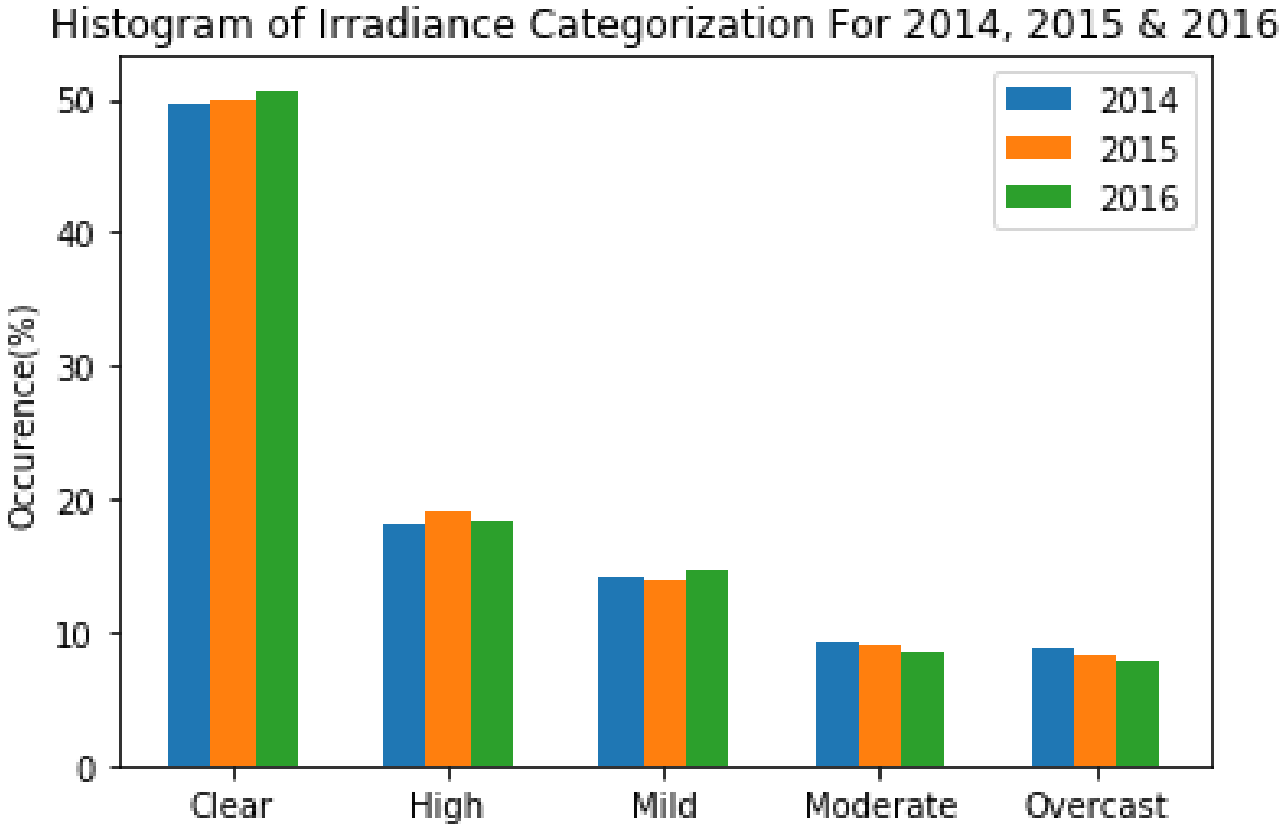
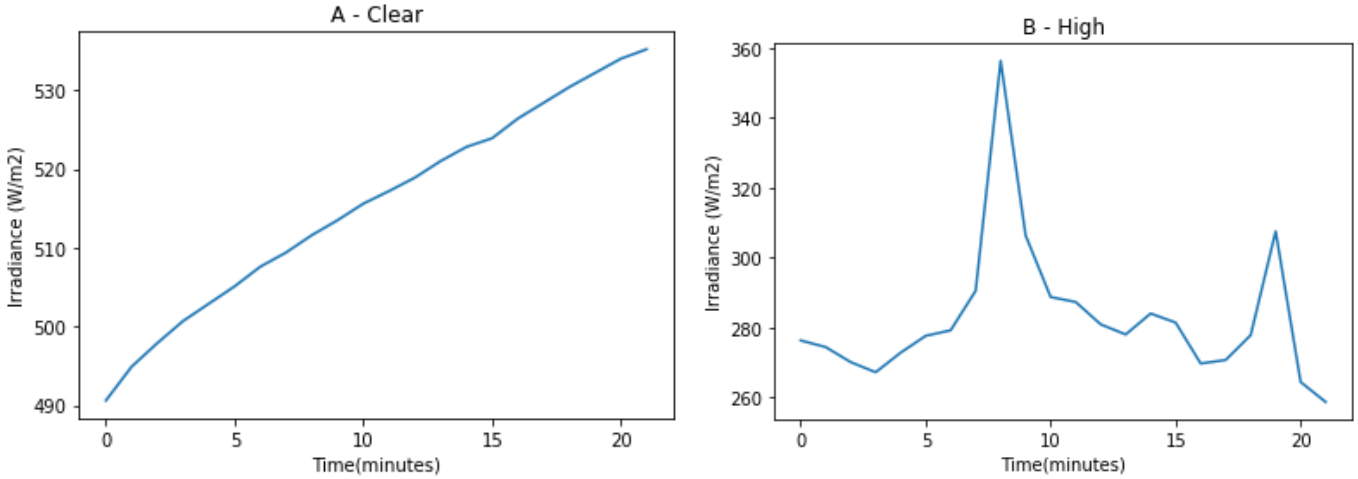


Figure 32: A histogram of the occurrence of the different sky conditions over the years 2014, 2015 and 2016.

To gain better insight into the model performance during the different occurring sky conditions separately, the performance of the CNN-Main model and persistence model have been tested separately for each sky condition. This technique will give an overview of how the DL model performs during different sky conditions. The obtained results are depicted in Table 11. It is visible that the CNN-Main model outperforms the persistence model, except for overcast conditions sky conditions. Overcast conditions result in constant low GHI measurements since the Sun is constantly obscured and is therefore well described using the persistence model. However, for the other sky conditions, the CNN-main model outperforms the persistence model, mainly during the clear and high classified sky conditions.



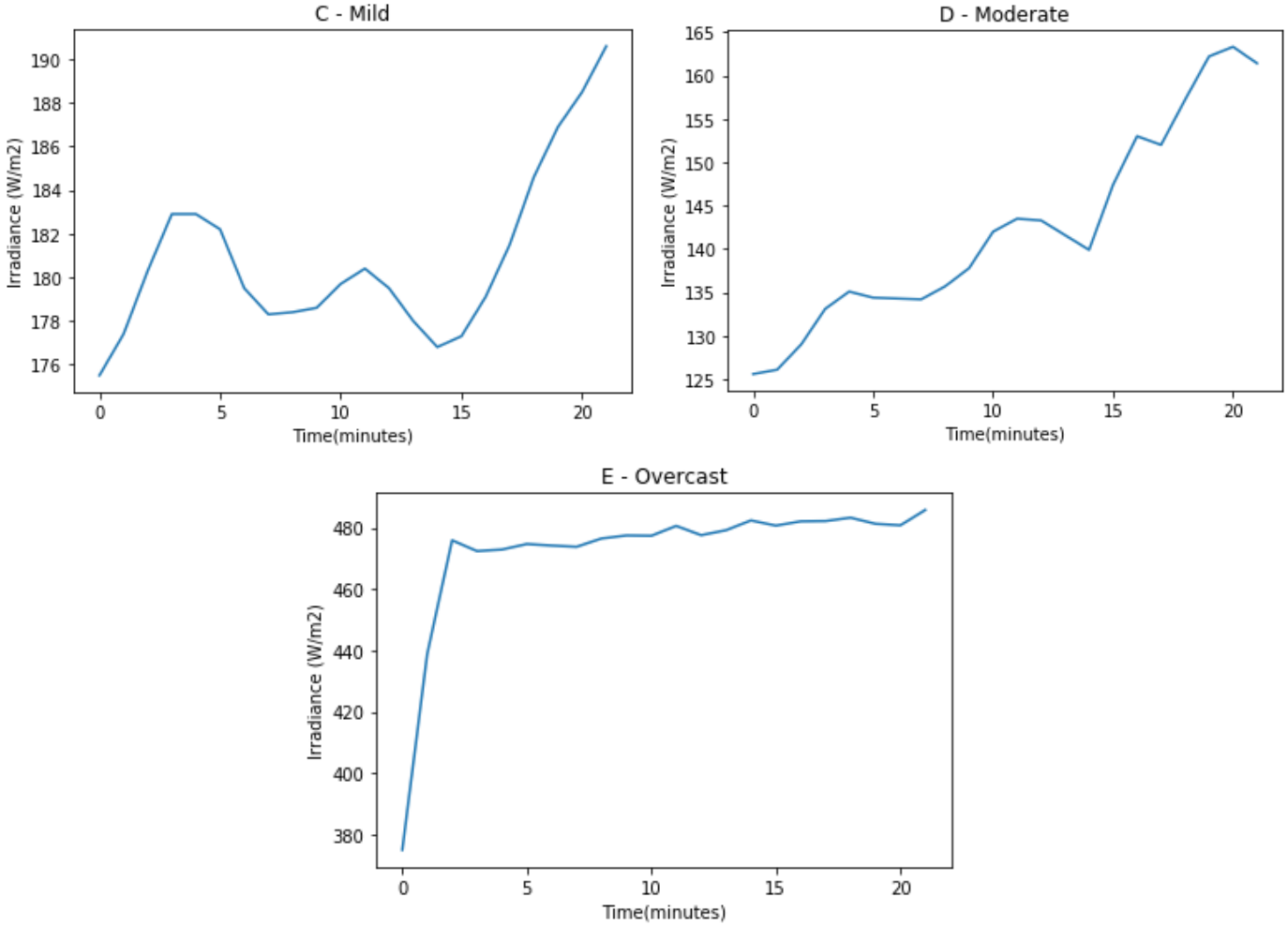


Figure 33: Examples of how the GHI fluctuates during different types of sky conditions. Figure A shows a clear sky day, Figure B shows a highly cloudy day, C a mildly cloudy day, D a moderately cloudy day and E an overcast day.

Table 10: Three random images taken from each set of different classified sky conditions are shown.

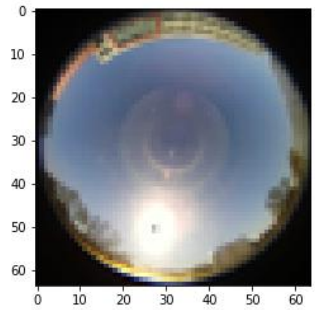
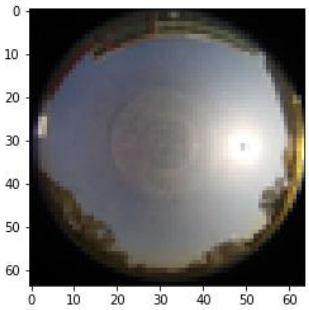
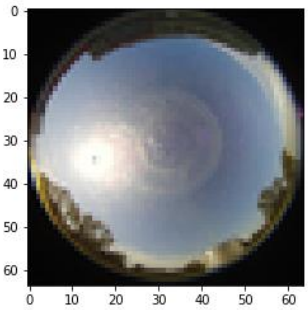
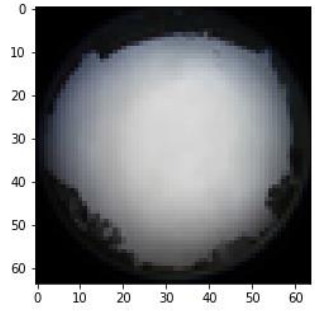
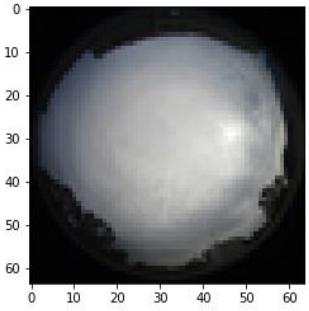
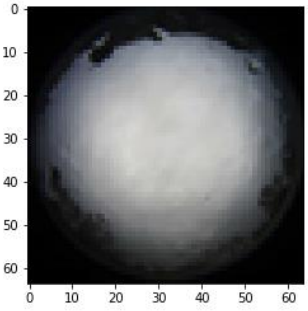
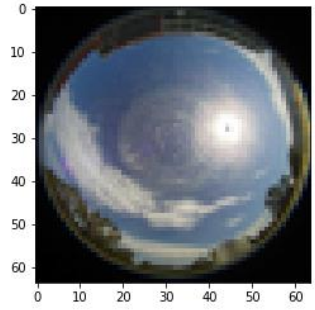
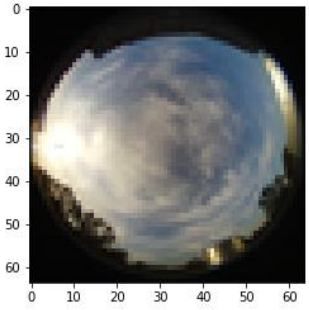
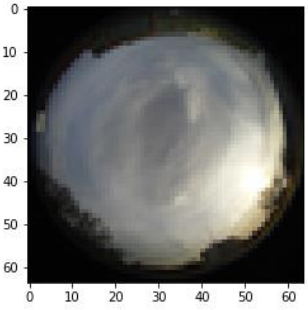
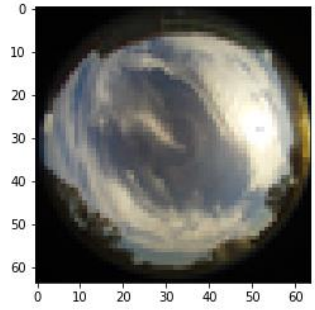
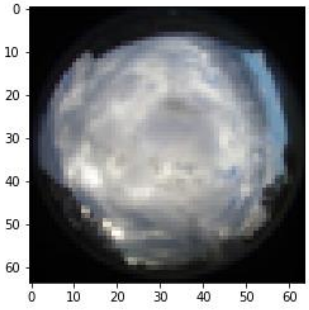
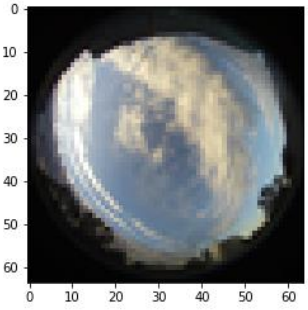
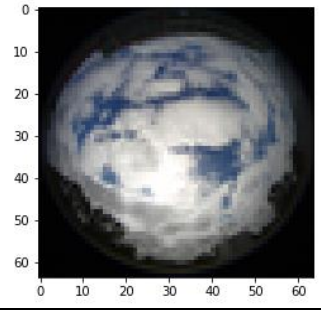
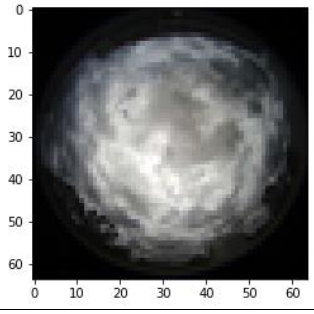
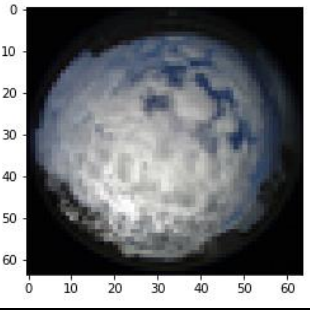
Sky Condition	Example 1	Example 2	Example 3
Clear			
Overcast			
Mild			
Moderate			
High			

Table 11: The performance of the CNN-Main model and Persistence model during different sky conditions is depicted.

Sky Conditions	CNN-Main		Persistence	
	MAE(W/m^2)	nMAE	MAE(W/m^2)	nMAE
Clear	9.8	2.0%	25.1	5.0%
Overcast	20.0	29.5%	13.5	19.9%
Mild	27.5	6.7%	30.4	7.5%
Moderate	45.2	10.3%	49.1	11.2%
High	94.0	17.9%	108.7	20.7%

From Table 11 the CNN-Main model is best applied during all sky conditions except during overcast conditions. Suppose the sky condition class can be predicted accurately before performing the irradiance prediction, a hybrid model could be used that uses the persistence model during expected overcast skies and the CNN-Main model during the other sky conditions. More elaboration about this subject will be given in Chapter 5.

4.6 Single GHI input compared to Sequence of GHI.

The CNN-Main (Seq- GHI^{measured}) model has been trained in the same way the CNN-Main model has been trained, but with a minor difference. Instead of giving the model the measured GHI at t=0, the model added the input of the last ten measured GHIs at the auxiliary data input. The results of the CNN-Main (Seq-GHI^{measured}) model and CNN-Main model are depicted in Table 12. From Table 12 follows that giving the GHI sequence as input results in a minor improvement, namely 1.8% relative to the CNN-Main model.

Table 12: The influence of using a sequence of past irradiances on the performance of the CNN-Main model.

Model	RMSE(W/m^2)	nRMSE	Forecast Skill
CNN-Main	31.61	7.2%	-
CNN-Main Irradiance Sequence	31.05	7.1%	1.8%

4.7 Model Behavior Examples

Figure 34, Figure 35, Figure 36 and Figure 37 show examples of the CNN-Main and persistence model behavior during cloudy sky conditions with the image on the left and the graph with measured and predicted irradiances on the right. The examples in Figure 34, Figure 35, and Figure 36 are randomly obtained from the sky conditions 'high,' and the image sample in Figure 37 is an example of the prediction with the highest average MAE of 2016 with an average MAE over 21-minute timesteps of 615.9 W/m^2.

From the random examples during cloudy conditions depicted in Figure 34, Figure 35, and Figure 36, the CNN-Main predictions are not oscillating while the measured GHI does. The CNN-Main predictions fit in between measurement oscillations rather than predicting the actual peaks and valleys. Two main factors that limit the model in predicting peaks and valleys are the deep learning model structure that will allow a certain complexity and the amount and quality of data. The current amount and quality of data points and model complexity are insufficient to predict peaks and valleys accurately. However, the overall performance shows promising results in predicting the irradiance and outperforming the persistence model during all-sky conditions, except overcast conditions. More information about improvements on the current model is given in Chapter 5.

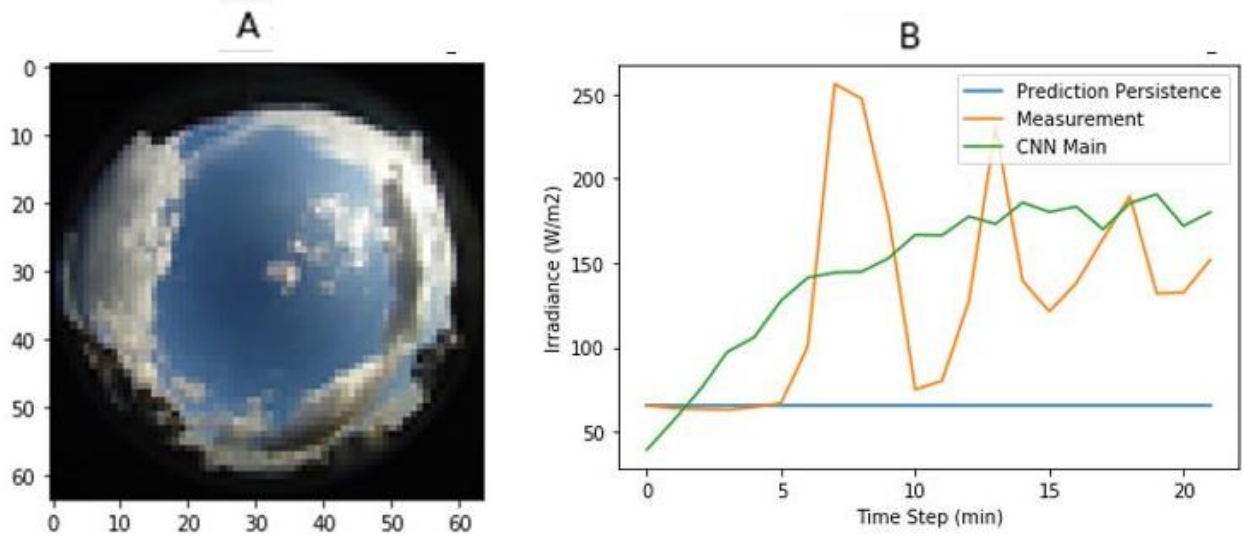


Figure 34: Example 1 of CNN-Main model behavior during cloudy sky conditions. On the left in Figure A, is the image shown that results in the predictions and measurements on the right in Figure B.

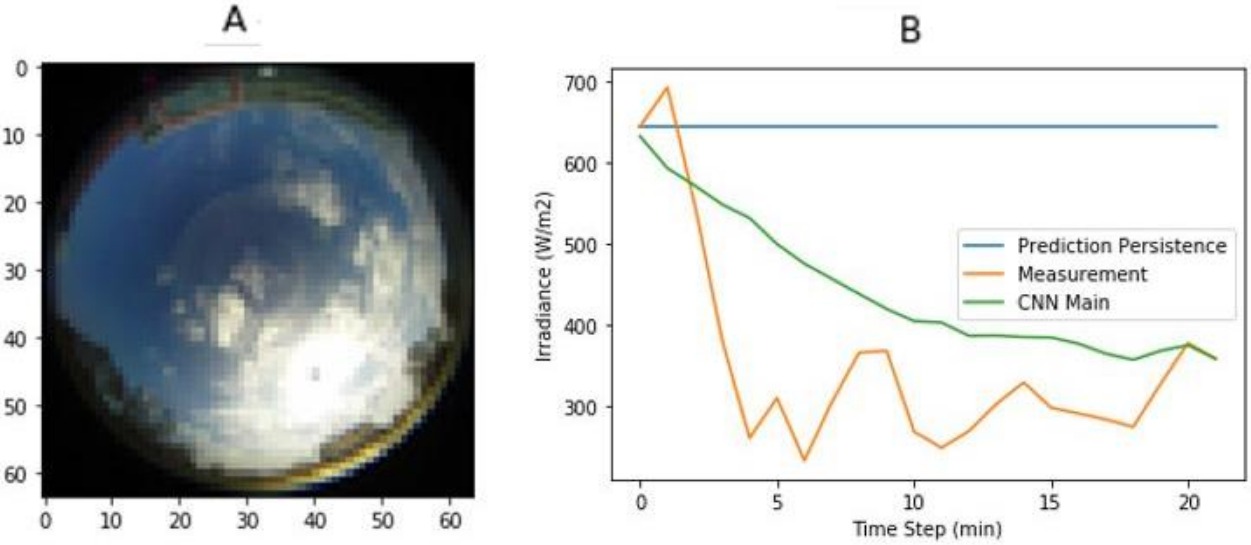


Figure 35: Example 2 of CNN-Main model behavior during cloudy sky conditions. On the left in Figure A, is the image shown that results in the predictions and measurements on the right in Figure B.

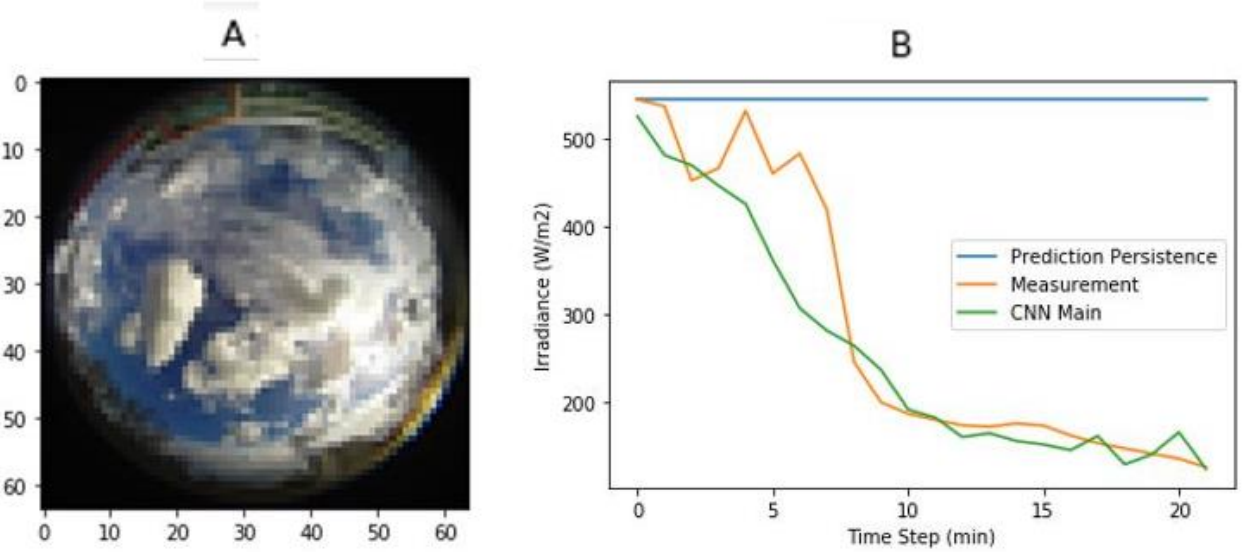


Figure 36: Example 3 of CNN-Main model behavior during cloudy sky conditions. On the left in Figure A, is the image shown that results in the predictions and measurements on the right in Figure B.

As already depicted in the histogram in Figure 28, the model has a good overall performance, but on rare occasions, high errors occur, as illustrated in Figure 37. Looking at the measured irradiance, it becomes clear that dark clouds obscuring the Sun can make the irradiance jump from 200W/m^2 to 1000W/m^2 in subsequent timesteps. According to the low measured irradiance measurements in the first 18 timesteps, the pyranometer is likely obscured by dark clouds. The CNN-Main model is unable to predict the obscuration of the Sun, resulting in a large overprediction of the irradiance.

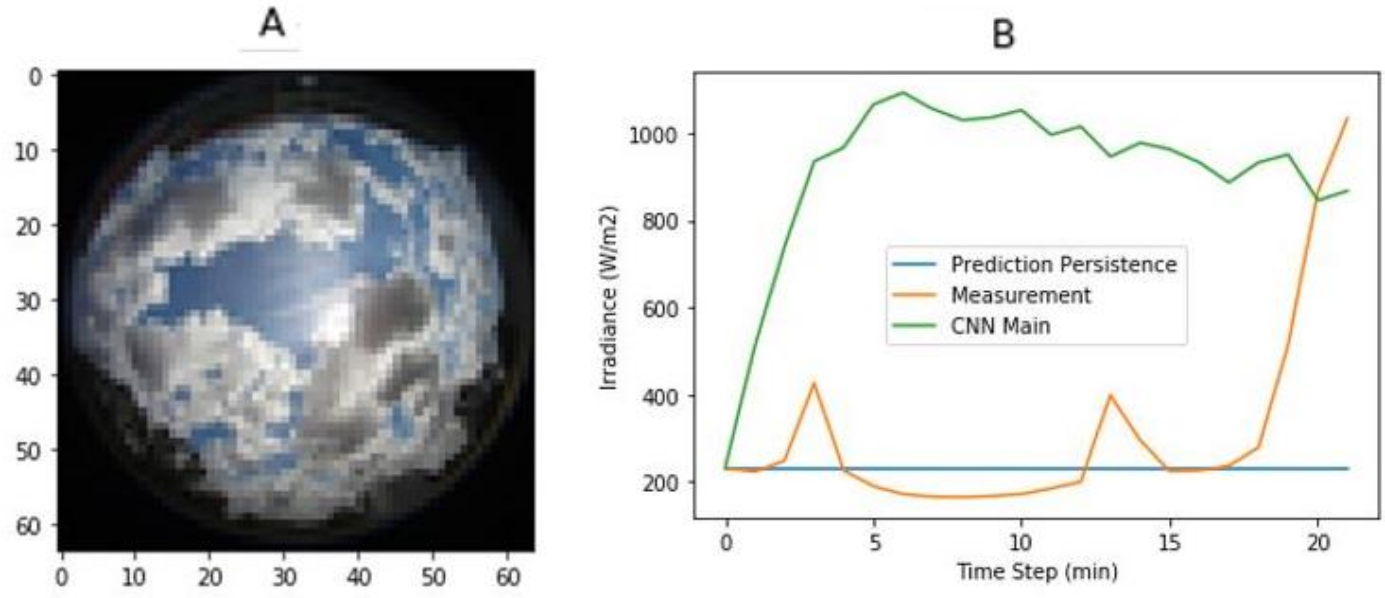


Figure 37: Example of the worst performance of the CNN-Main model behavior during the year 2016.

To obtain a better insight into the model behaviour of the CNN-Main model an example is elaborated on. The CNN-Main predictions, and measurements on a 20-minute horizon have been numbered in Figure 38. Each number corresponds with the image of which some are depicted in Figure 39 and all images are illustrated in the Appendix. It is visible that the CNN-Main model predicts that the GHI gradually increases, while the GHI stays rather constant. Important to note is that the CNN-Main model only uses the image and auxiliary data at timestep 0. Similar behavior is observed with the RNN-Main model and the CNN-Main(Seq- GHI^{measured}) model. This shows that using a sequence of data as input as is done in this study does not resolve this problem.

When looking closer at image 0 within Figure 39, it is visible that the Sun is initially obscured. The obscuration persists until timestep 14, resulting in low irradiance measurements as is visible within Figure 38. At timestep 14 the clouds in front of the Sun move away, resulting in a rapid increase of the irradiance. Lower irradiance is measured at timesteps 17 and 18, while in the corresponding images the Sun seems unobscured. This can be explained since the used irradiance measurements are the average irradiance values over one minute and within a single minute the Sun can be temporally obscured and unobscured. On top of that is it visible that the clouds are in proximity of the Sun at timesteps 17 and 18, which make it more likely that the Sun was partially obscured during these minutes.

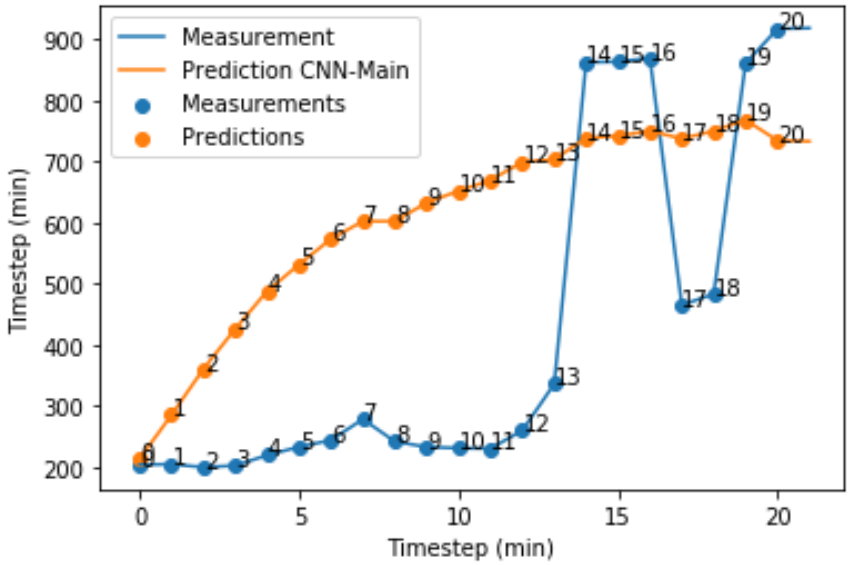


Figure 38: Example of the CNN-main model behavior during the year 2016 on which it performs poorly. Each measured point is depicted with a number that correspond with an image.

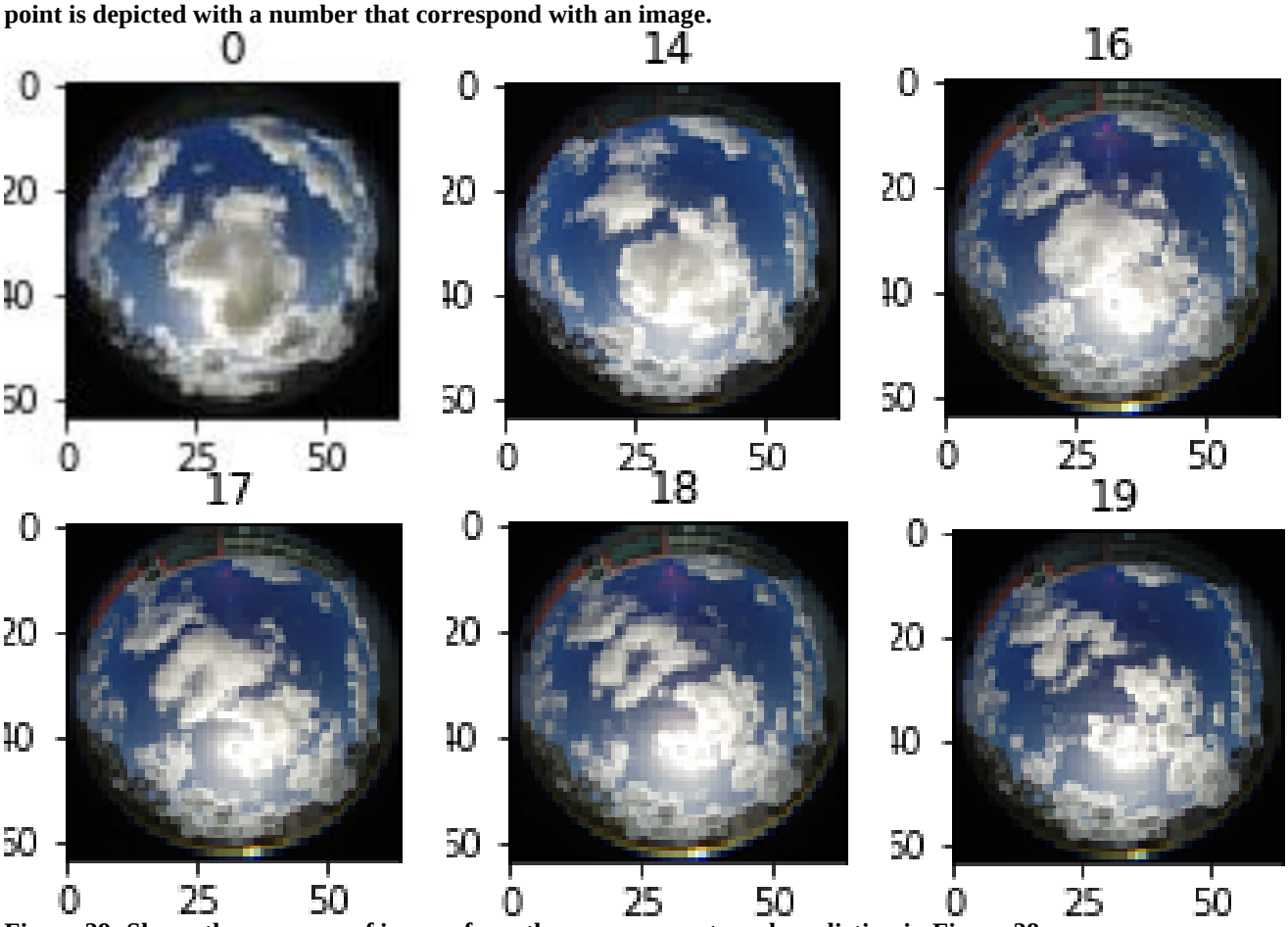


Figure 39: Shows the sequence of images from the measurements and prediction in Figure 38.

This behavior shows again that the CNN-Main model does not predict the individual spikes and valleys but aims to predicts the mean of the spikes. An evident approach is to give a sequence of image inputs; however, the same problem arises with the RNN-Main model, which takes three images as inputs. Besides, in the papers of (Paletta et al., 2021) and (Kong et al., 2020), similar behavior is observed, taking a sequence of 5 images. Following this model behavior, the CNN-Main model cannot accurately predict the irradiance spikes and peaks with current model inputs and DL architectures. Although, the DL model outperforms the persistence model more research into this field is required to increase performance and is informed about in Chapter 5.

5 Discussion

5.1 Study Comparison

This study shows that DL is a promising approach for interpreting All-Sky images and auxiliary data to forecast short-term GHI between 0 to 21 minutes. In previously studies (Paletta et al., 2021)(Kong et al., 2020) it is also shown that DL is a relevant approach for GHI forecasting using sky images. In this study a DL approach with similarities to (Paletta et al., 2021) and (Kong et al., 2020) is applied to a new significantly larger dataset, which shows that the deep learning models can extract important information from sky images. Differences and similarities in the studies are depicted in Table 13 and Table 14. In all studies is shown that the DL models occasionally miss important weather events, resulting in large prediction errors, which is also observed within the DL model of this study.

The studies are compared based on the performance relative to the persistence models. The best performance is obtained with the DL model in (Paletta et al., 2021). The model performance of this study and the model in (Kong et al., 2020) perform almost equally well. Important to note is that comparing performance of GHI forecasting models from different studies holds inaccuracies due to factors like geographical location, pre-processing, forecast horizon, input variables, the dataset and more. Several of these factors will be discussed and elaborated on the influence of these factors on the model performance.

Table 13: The comparison between the study of (Kong et al., 2020) **and this study is depicted.**

Element	(Kong et al., 2020)	This Study
Prediction label	PV-output (2kW)	GHI
Location	Hong Kong (city)	California, Folsom (city)
Data Timeframe	3 Months	3 years
Sample frequency	30-seconds	1-minute
Inputs per sample	Image & PV Power Output	Image, $GHI^{clear\ sky}$, $GHI^{measured}$, Meteorological data
Input sequence	5 data points	1 or 3 data points
Forecast horizon	4-20 minutes	0-21 minutes
Performance relative persistence (%)	27.61%	27.60% (Based on 4-20 minute horizon)

Table 14: The comparison between the study of (Paletta et al., 2021) **and this study is depicted.**

Element	(Paletta et al., 2021)	This Study
Prediction label	GHI	GHI
Location	France, Paris (city)	California, Folsom (city)
Data Timeframe	3 Years	3 years
Sample frequency	2-minutes	1-minute
Inputs per sample	Long Exposed Image, Short Exposed Image, $GHI^{measured}$, Sun position coordinates	Image, $GHI^{clear\ sky}$, $GHI^{measured}$, Meteorological data
Input sequence	5 data points	1 or 3 data points
Forecast horizon	Point forecasts (2, 6, 10, 20 minutes)	0-21 minutes
Performance relative persistence (%)	52.01% (at minute 20)	40.02% at the 20^{th} minute

Geographical Location:
When the performance of DL models is compared on different locations problems can occur regarding the occurrences of weather conditions. The dry, temperate climate within Folsom, California results in a dataset with a lot of clear days, while the mild continental climate in France, Paris result in a dataset with more cloudy days. When training a DL model on the dataset of California, Folsom the model will more likely perform better on clear days. This happens, because the DL model will learn from the dataset that it is more important to perform well during clear sky conditions, since these conditions are more represented within the dataset. On the other hand, the DL model trained on the dataset of France, Paris will most likely perform better during cloudy conditions.

The performance of the DL model in this study is tested on a dataset with approximately (±1%) similar relative occurrences of sky conditions as on which it is trained. Most likely this is also the case within the studies of (Paletta et al., 2021) and (Kong et al., 2020), however it is not specifically stated. The performance of the persistence model is also different for different sky conditions as is shown in this study in Table 11. The performance of the persistence model during overcast conditions is for example much better compared to during partially cloudy conditions. This study also showed that the relative performance of the DL model relative to the persistence model differed per sky condition. The relative sky condition occurrences within a dataset thus result in different relative performances of the DL models to the persistence models. Comparing the different DL models relative to their performance to the persistence model is therefore flawed. Potential solutions to this problem are hybrid models and pre-classification algorithms on which is elaborated in Section 5.2.

Pre-Processing
Within DL normalization is commonly done, due to the variety of input data. The range of the temperatures is between 10 and 40, while the range of the GHI goes from 0 to 1400. By normalizing the data, all data has the same scale, which results that the learning algorithm converges the weights faster to the optimal configuration (Agarwal, 2014). It is expected that without normalization the performance of the model will be similar or slightly worse, while taking more time to train.

The wind velocity and direction were converted to wind vectors. This has been done since the angle of the wind direction was in degrees from 0° to 360°. The direction in degrees are bad model inputs since the values of 0° and 360° are far apart but should be close together since they denote almost the same wind direction. The distribution of wind vectors is therefore easier to interpret compared to direction in degrees (TensorFlow, 2021b). However, in the ablation study is shown that the meteorological data has minimal influence on the DL model performance, and since the wind vectors are part of the meteorological data it is expected that this pre-processing step most likely has minimal influence on the DL model performance.

Pre-processing prepares the data to train the DL models. The different studies use different pre-processing steps and are too comprehensive to discuss all in this section. Two main pre-processing differences will be depicted. First, in the study of (Paletta et al., 2021) the image downscaling is done to 128x128x1. The image does not use color channels but is converted to grayscale to save memory. Within the study of (Kong et al., 2020) downscaling is not mentioned. This study downscales the images to 64x64x3. In the study of (Sun, 2018) is shown that higher sky image resolutions result in better prediction performance, and also that color images outperform grayscale images. However, due to computer power and memory constraint studies often downscale image resolution and/or convert images to grayscale format. In this study a lower image resolution is used. Ideally, high resolution color images are used to obtain high performance if training time and computer memory are not considered.

Hardware Limitations
Using large amounts of data and training DL models are ideally done with powerful computers. Especially using a NVIDIA GPU can speed up the learning process, namely with 5 to 6.7 times compared to using a CPU, due to the parallel computing power of GPUs (Nvidia Corporation, 2015). Other GPUs are not compatible

with TensorFlow and therefore only certain NVIDIA GPUs are appropriate to use. Within the studies of (Paletta et al., 2021) and (Kong et al., 2020) a computer with NVIDIA GPU is used, while in this study no GPU is used. For future research is recommended to make use of a NVIDIA GPU to significantly increase the training times.

During this study there was the possibility to use a powerful computer at the TU Delft, which would speed up the training process and allow for higher resolution images. However, during this study the sensitivities of the DL model were assessed by experimenting with the DL model architectures resulting in frequent modifications. Due to the limited availability of this powerful computer, it was decided not to use it.

Forecast Horizon

For short-term energy trading, electricity can be traded 5 minutes before delivery using contracts of 15-minutes, 30-minutes and per hour (Epexspot, 2021). In this study a contract is considered of 15-minutes that has to be traded 5 minutes before delivery. Therefore it is important to know what the irradiance does 20-minutes in advance. Acquiring the input data for the DL model and performing the predictions take time and therefore predicting 20-minutes into to future comes short. Ideally 21 minutes are predicted to compensate for lost time during data acquisition and making the prediction, which is why a 21-minute forecast horizon is considered.

As depicted in the study comparison in Table 13 and Table 14, different forecast horizons are used within the studies. It is shown that using different forecast horizons result in different performances of DL models and that smaller forecast horizons result in better performance in Table 8. Comparing studies with models trained on different forecast horizons is therefore not optimal, and ideally comparison of models is done by training a model using a similar forecast horizon to subsequently compare the performance. However, the influence of forecast horizon on the performance of DL models is investigated showing that shorter forecast horizon slightly improves the overall DL model performance, especially in the initial time steps.

It was expected that shorter forecast horizons would increase the performance. All the input data (image and auxiliary data) is converted within the DL model to a 1-D vector representation from which the GHI prediction is made. If the prediction horizon is smaller, the single vector created within the DL model could be more specialized for the smaller forecast horizons and therefore perform better. Within (Paletta et al., 2021) the forecast horizon is even reduced to a single point in the future, which allows an even better 1-D representation of the single point in future. However, making a prediction 20 minutes into the future would then require 20 separate trained models, which significantly increase training time. However, once the models are trained, a single prediction is performed in less than a second assuming that similar DL architecture complexity is used as in this study.

Input Variables

In this study a single image or a sequence of 3 images is used as input, while within the studies of (Paletta et al., 2021) and (Kong et al., 2020) a sequence of 5 images is used as input. It is shown that using a sequence of images as input results in better irradiance prediction performance in the study of (Pothineni et al., 2019). However, in this study the CNN-Main model using a single datapoint outperformed the RNN-Main model, using a sequence of 3 datapoints, thus showing that it is better to use a single input compared to a sequence of 3 as input. An explanation for this is that the RNN-Main model and CNN-Main model have different DL architectures, since the RNN-Main model makes use of an RNN layer. Potentially, when the CNN-Main model is using a sequence of 3 images as input, without a RNN layer, that it outperforms the CNN-Main model using a single datapoint as input. However, this has not been investigated in this study. In the study of (Pothineni et al., 2019) similar DL architectures are used for this comparison, and showed that a sequence did perform better than a single point. In future research it is therefore recommended to use a sequence as input.

Using sequences as input to a DL model has a drawback concerning memory. As shown in this study and is confirmed within the studies of (Ng, 2020) and (Paletta et al., 2021), more data result in better performance.

Ideally large datasets with large sequences of high-resolution images are used as input, but computer memory limits the possibilities. When looking at the memory usage of the dataset used, the far majority is occupied by images (~99%). If instead of a single datapoint a sequence of 5 datapoints is used as input, each datapoint within the dataset occupies almost 5 times as much memory, limiting the amount of datapoints that can be used considerably due to computer memory limits. The computer memory limits can be solved by using the *ImageDataGenerator* from Keras (Keras, 2021). The *ImageDataGenerator* allows to load a batch of the dataset, train the model on that batch, remove that batch from memory, and load a new batch to start again. This way, internal memory will not be a constraint (as long as the amount of data in the batch fits in memory) and larger image sequences, higher resolutions and large datasets can be used. However, using the *ImageDataGenerator* significantly increases training time of the models. Due to the added complexity and increased training time the *ImageDataGenerator* from Keras is not used in this study.

Within this study the clear sky GHI is used as input and shows improved performance. Within (Paletta et al., 2021) the Sun position coordinates are used and in (Kong et al., 2020) neither the clear sky GHI or Sun position coordinates are used. In this study is shown that the addition of the clear sky GHI is an improvement in performance. The influence of the Sun position coordinates used to the model performance is unknown. It is expected that the clear sky GHI obtained through the Perez & Ineichen model is a better addition to the model compared to the Sun position coordinates, since the clear sky GHI entails more information about the interaction with the GHI and atmosphere. A sensitivity study about which input is better (the clear sky GHI or solar position coordinates) should show which input, or a combination of inputs would be better.

Dataset

This study and the study of (Paletta et al., 2021) use random samples of multiple years for training the DL models. However, within the study of (Kong et al., 2020) the clear sky days, completely overcast days and rainy days were omitted from the dataset. The dataset selectivity within (Kong et al., 2020) results that comparing studies becomes more flawed, due to the sensitivity to sky conditions as explained above. Besides the sky conditions present in the dataset, the amount of data within each dataset is also different for each study. In Table 15 the amount of data for training, validation and testing is shown.

Table 15: The amount of data for training, validation and testing for the compared studies is depicted.

Dataset	(Kong et al., 2020)	(Paletta et al., 2021)	This Study
Training	18,643	35,000	169,505
Validation	7,989	10,000	42,376
Testing	4,640	10,000	242,038

Within DL, more data results in better model performance (Ng, 2020). This is also proven within this study looking at Figure 29 and in the study of (Paletta et al., 2021). Besides the amount of data, this study also showed that the DL model performance increases when the dataset comprises multiple years compared to a single year as is depicted in Table 6. The paper of (Paletta et al., 2021) also conducted a study to the quantity of data and compared a dataset of a single year to a dataset of two years for training. The study of (Paletta et al., 2021) found increased performance and concluded that the reason is solely due to the larger dataset. However, in this study is shown that the increased performance is due to larger data quantities and the variation of the dataset.

This and preceding studies show that the usage of more data results in better DL model performances. However, despite using significantly more data in this study compared to the study of (Paletta et al., 2021), the DL model of (Paletta et al., 2021) showed increased performance relative to the persistence model. It can be argued that this is due different forecast horizons or sky conditions occurrences in the dataset, but most likely is a result of input data quality. The higher image resolution and large sequence input in the study of (Paletta et al., 2021) seem to result in a more significant performance increase compared to using a significantly larger dataset with lower image resolution and single input in time as is done in this study. Most likely this study can be improved more effectively by increasing the input data quality, compared to using more

datapoints. Ideally a large dataset consisting of sequence inputs with high resolution images is used, and improvements to the DL model of this study are discussed in Section 5.2. Besides the amount and quality of data, the DL architectures used were different on which will be elaborated in the next section.

Deep Learning Architecture
In (Paletta et al., 2021) multiple deep learning architectures are described for irradiance forecasting, in which is shown that the LSTM-CNN structure outperforms the CNN-structure. However, in (Kong et al., 2020) is stated that CNN models outperform the LSTM-CNN models. In this study a simplified layer of the LSTM, namely the RNN, is used and shows that the CNN-Main model outperforms the RNN-Main model. The architectures of the different studies mainly differed at the CNN part of the models. The study of (Paletta et al., 2021) makes use of the residual neural networks (ResNet) within the CNN layers (He et al., 2016). ResNet have proven to improve the performance of DL models using images. The addition of the ResNet within the DL model of (Paletta et al., 2021) could be an explanation that the CNN-LSTM model outperforms the CNN model. In this study ResNet is not used, due to time constraints.

Hyperparameters
Hyperparameters determine the DL model design, like the number of layers, amount of weights in the layers, the loss function and more (Nabi, 2019). Hyperparameter tuning is done by trial and error where the model is trained and evaluated after each change, which makes hyperparameter tuning a time-consuming process. In this study a computer without NVIDIA GPU is used, resulting in long training times. Therefore, the used hyperparameters in this study are not tuned, but based on the previous study of (Paletta et al., 2021). One of the important hyperparameters used in this study is the MSE loss function on which will be elaborated.

Ideally the DL model is learning in such a way that it is able to predict the irradiance during all types of sky conditions. To achieve this, the DL model minimizes the MSE by adapting internal weights to improve the performance on the training dataset. When looking at the irradiance, the highest values are measured during clear sky conditions and lower values during cloudy conditions. A relative prediction error of 10% during clear sky conditions results therefore in a larger absolute error compared to cloudy sky conditions. As is stated in Section 3.1.6, the MSE penalizes larger errors more than smaller errors. This makes the DL model penalize errors during clear sky conditions more than during cloudy sky conditions. On top of that is shown that the used training dataset consists of many clear sky condition datapoints, which make the model prioritize these conditions even more. A solution is to develop a new loss function that incorporates a way of punishing poor predictions during partly cloudy sky conditions more than during clear sky conditions, to improve the performance during these conditions. Another approach is to make models for separate sky conditions, on which will be elaborated in Section 5.2. A combination of using a new loss function and making DL models for separate sky conditions could also be a solution to the problem stated above.

5.2 Future Research

Within Section 5.1 the implications of the different design decisions within DL models for irradiance forecasting are stated, based on a study comparison. Using the knowledge obtained during this study about the design decision implications, combined with results from the ablation study, sensitivity analysis and influence during different weather conditions, DL models for irradiance forecasting ideally makes use of the following:

- A large dataset.
- A sequence of 5 or more high resolution color images as input.
- A sequence of multiple past measured irradiances.
- A ResNet within the CNN
- The clear sky irradiance as input
- No meteorological data as input

To make a new DL model using above mentioned points and avoid computer memory problems, the *ImageDataGenerator* of Keras should be used. Using existing proven DL architectures as depicted the study of (Paletta et al., 2021) is recommend. The *ImageDataGenerator* will result in long training times, so ideally

a powerful computer with NVIDIA GPU is used. This study and the studies of (Paletta et al., 2021) and (Kong et al., 2020) still miss important weather events resulting in occasional large irradiance errors. The current inputs in combination with the current DL architectures are not sufficient to omit those large errors, however using above mentioned improvements might resolve this. Besides, extra inputs that give more information about clouds obscuring the Sun could be the key to solve this problem. Some possibilities for extra inputs are given below.

- At the TU Delft a new All-Sky Imager is installed accompanied with software that allows the extraction of the cloud movement in pixels per second. The cloud movement can be used as extra input into the DL models to obtain a better view of the cloud dynamics.
- Weather stations like KNMI calculate different variables like cloud height, cloud composition and more variables using satellite images. These images are taken with a 15-minute frequency, thus might not be ideal for 1-minute frequency forecasting. However, if those variables are rather constant within 15-minutes this might not be a problem. Using those inputs as extra auxiliary data in DL models might improve the performance. The satellite images could also directly be used within the DL model, using an extra CNN.

Sky Condition Sensitivity

The study comparison shows that comparing DL models is problematic due to differences in sky condition occurrences at different geographical locations. To compare different datasets and DL models fairly, a new method could be developed. If different datasets are classified as in this study, the performance of the models could be evaluated on each sky condition separately. Different methods for sky condition classifications are available, based on measured irradiances values and clear sky values (Hartmann, 2020), based on sky images (Heinle et al., 2010) and DL models using sky images (Nie et al., 2020). Comparing different datasets and DL models for different sky conditions using similar sky condition classification methods and irradiance normalization will result in more accurate comparisons.

Future research to create a benchmark method to compare the forecasting performance of DL models trained on different datasets is recommended. The benchmark method should at least take the following into account.

- Evaluate the performance of the model during different sky conditions. The sky conditions should be classified in a similar way, so the sky conditions between datasets correlates.
- Evaluate the performance for different quantities of data.
- Normalization of irradiance, to compare locations with higher average irradiance to locations with lower average irradiances.
- Evaluate the performance of the model on a similar forecast horizon.

Sky condition classification can also be used as a pre-processing step. A dataset could be split into sky classes, which can subsequently be used to train different models. Depending on the amount of sky condition classes, different models can be trained specialized for a certain sky condition class. These models will likely show improved performance since the models will not be generalized for all sky conditions but specialized for a single sky condition. However, prior to creating different models it is first important to be able to predict the type of sky condition that will occur accurately. Using multiple models for a single prediction is called a hybrid model. Within solar forecasting it is common to make use of hybrid models (Haupt et al., 2017).

New Methods

Many new DL methods for interpreting images are being developed. Promising methods are video prediction algorithms (Le Guen & Thome, 2020) that predict future images based on a sequence of past images. Another promising method is solving NWP fast and efficiently using DL on the short term (Z. Li et al., 2020). There are also different DL architectures for interpreting sequence data to make predictions, like Transformers, which show promising results with video prediction (Vaswani et al., 2017). The field of short-term irradiance forecasting using sky images and DL is relatively novel compared to other short-term irradiance forecasting studies looking at the dates of publications of most papers regarding this subject. Therefore, it is expected that in the coming years new methods will be developed

Multiple Datasets

If a DL model using sky images for short-term irradiance forecasting will be used for a location, data must be collected to train the DL model. This means that after installation, several months of data must be collected before the DL model can be trained and implemented. Ideally, a DL model would be developed that works for different geographical locations. This would allow the installation of a ASI and directly use the sky images to make predictions. This can be achieved by looking into different datasets consisting of all sky images and irradiance measurements. Recently three datasets recorded in Switzerland consisting of all sky images and measured irradiance with a frequency of 10-seconds consisting of 1.57 million samples each is made publicly available (Evangelos Ntavelis, 2021). Using multiple datasets from different geographical locations make it possible to train a DL model on a geographical location and look how well such a model performs on a different geographical location. Currently the two publicly available datasets of (Evangelos Ntavelis, 2021) and (Carreira Pedro et al., 2019) could be used for this research. On top of that is currently data acquired with the recently installed ASI at the TU Delft, which could also be used to extend the number of used datasets.

6 Conclusion

Predicting the irradiance is critical when integrating PV to maintain grid stability. The main irradiance prediction error is due to clouds obscuring the Sun, which can be detected using All-Sky Imagers. Image processing techniques are inadequate to interpret sky images, missing certain cloud types and movements. However, since large datasets are publicly available, deep learning is a strong choice. Deep learning has already proven to be effective in extracting important information from images and its applicability within irradiance forecasting using all sky images. In this study a new short-term 21-minute ahead deep learning irradiance forecasting model is trained and tested on a considerably bigger dataset compared to preceding studies. This study extensively investigates the sensitivity of the deep learning model, with respect to important factors. This study demonstrates that deep learning using sky images and auxiliary data for irradiance forecasting is effective and outperforms the persistence model with 24.8%. It is shown that besides using more data, also the variability of data from different years results in better performance. In addition, it is shown that the clear sky irradiance is an improvement for the model performance and that meteorological data has a negative impact on the performance. Furthermore, the model performance is evaluated during different sky conditions and shows that the deep learning models outperform the persistence model, except during completely overcast conditions. Despite outperforming the persistence model, the deep learning model suffers from occasional large prediction errors during partially cloudy sky conditions and tend to predict the trend of the irradiance fluctuations rather than the actual fluctuations. New methods like video prediction algorithms or using pixel movement maps as extra input possibly results in models that will not miss Sun obscuration and predict actual irradiance fluctuations instead of the trend of the fluctuations. Short-term prediction of irradiance using DL will facilitate the control of solar power plants, thus reducing the problems associated to intermittency of renewable energy. This will cause an even better acceptance of PV energy, facilitating its penetration on power systems and helping in the energy transition towards an emission free electric system.

Bibliography

Agarwal, S. (2014). Data mining: Data mining concepts and techniques. In *Proceedings - 2013 International Conference on Machine Intelligence Research and Advancement, ICMIRA 2013*. https://doi.org/10.1109/ICMIRA.2013.45

Al-Lahham, A., Theeb, O., Elalem, K., Alshawi, T. A., & Alshebeili, S. A. (2020). Sky imager-based forecast of solar irradiance using machine learning. *Electronics (Switzerland)*, *9*(10), 1–14. https://doi.org/10.3390/electronics9101700

Antonanzas-Torres, F. (2019). Clear sky solar irradiance models: A review of seventy models. *Renewable and Sustainable Energy Reviews*.

Bhande, A. (2018). *What is underfitting and overfitting in machine learning*. https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76

Brownlee, J. (2018). *Difference Between a Batch and an Epoch in a Neural Network*. https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/

Brownlee, J. (2019). *How to use Learning Curves to Diagnose Machine Learning Model Performance*. https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/

Brownlee, J. (2021). *Difference between backpropagation and stochastic gradient descent*. https://machinelearningmastery.com/difference-between-backpropagation-and-stochastic-gradient-descent/

Campbell Scientific. (n.d.). *RSR2-KIT - Rotating Shadowband Radiometer*. 1–3. https://www.campbellsci.com.au/rsr2-kit

Carreira Pedro, H., Larson, D., & Coimbra, C. (2019). *A comprehensive dataset for the accelerated development and benchmarking of solar forecasting methods*. https://doi.org/10.5281/ZENODO.2826939

Chauvin, R., Nou, J., Thil, S., & Grieu, S. (2016). Cloud motion estimation using a sky imager. *AIP Conference Proceedings*, *1734*(May 2016). https://doi.org/10.1063/1.4949235

Chow, C. W., Belongie, S., & Kleissl, J. (2015). Cloud motion and stability estimation for intra-hour solar forecasting. *Solar Energy*, *115*, 645–655. https://doi.org/10.1016/j.solener.2015.03.030

Chow, C. W., Urquhart, B., Lave, M., Dominguez, A., Kleissl, J., Shields, J., & Washom, B. (2011a). Intra-hour forecasting with a total sky imager at the UC San Diego solar energy testbed. *Solar Energy*, *85*(11), 2881–2893. https://doi.org/10.1016/j.solener.2011.08.025

Chow, C. W., Urquhart, B., Lave, M., Dominguez, A., Kleissl, J., Shields, J., & Washom, B. (2011b). Intra-hour forecasting with a total sky imager at the UC San Diego solar energy testbed. *Solar Energy*, *85*(11), 2881–2893. https://doi.org/10.1016/j.solener.2011.08.025

De Luca, G. (2019). *Bias in Neural Networks*. https://www.baeldung.com/cs/neural-networks-bias

Donges, N. (2021). *A guide to RNN: Understand recurrent networks*. https://builtin.com/data-science/recurrent-neural-networks-and-lstm

Epexspot. (2021). *Basics of the Power Market*. https://www.epexspot.com/en/basicspowermarket

Evangelos Ntavelis, J. R. (2021). *SkyCam: A dataset of sky images and their irradiance values*. https://github.com/vglsd/SkyCam

F. Holmgren, W., W. Hansen, C., & A. Mikofski, M. (2018). Pvlib Python: a Python Package for Modeling Solar Energy Systems. *Journal of Open Source Software*, *3*(29), 884. https://doi.org/10.21105/joss.00884

Goodfellow, I. (2016). Deep learning. In *Nature* (Vol. 29, Issue 7553). MIT Press.

Hartmann, B. (2020). Comparing various solar irradiance categorization methods – A critique on robustness. *Renewable Energy*, *154*, 661–671. https://doi.org/10.1016/j.renene.2020.03.055

Haupt, S. E., Jiménez, P. A., Lee, J. A., & Kosović, B. (2017). Principles of meteorology and numerical weather prediction. *Renewable Energy Forecasting: From Models to Applications*, 3–28. https://doi.org/10.1016/B978-0-08-100504-0.00001-9

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-Decem*, 770–778. https://doi.org/10.1109/CVPR.2016.90

Heinle, A., Macke, A., & Srivastav, A. (2010). Automatic cloud classification of whole sky images. *Atmospheric Measurement Techniques*, *3*(3), 557–567. https://doi.org/10.5194/amt-3-557-2010

Heliosat. (2021). *Solar Radiation And Meteorological Data Service*. http://www.soda-pro.com/home;jsessionid=82B4BC191D1647428686BE0B5BF51ECD

Howe, B. (2012). *Understanding the Bias Variance Tradeoff*. https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229

Ineichen, P., & Perez, R. (2002). A new airmass independent formulation for the Linke turbidity coefficient. *Solar Energy*.

Kazantzidis, A., Tzoumanikas, P., Blanc, P., Massip, P., Wilbert, S., & Ramirez-Santigosa, L. (2017). Short-term forecasting based on all-sky cameras. In *Renewable Energy Forecasting: From Models to Applications* (Issue Ccd). Elsevier Ltd. https://doi.org/10.1016/B978-0-08-100504-0.00005-6

Keras. (2021). *Keras: A Deep Learning API in Python running on top of the Machine Learning platform TensorFlow*. https://keras.io/api/

KNMI. (2021). *KNMI HARMONIE40 Single Level en Boundary Layer Fields Nederland*. https://www.knmidata.nl/data-services/knmi-producten-overzicht/atmosfeer-modeldata/data-product-1

Kong, W. (2017). Effect of automatic hyperparameter tuning for residential load forecasting via deep learning. *IEEE*, 1–6.

Kong, W., Jia, Y., Dong, Z. Y., Meng, K., & Chai, S. (2020). Hybrid approaches based on deep whole-sky-image learning to photovoltaic generation forecasting. *Applied Energy*, *280*(November), 115875. https://doi.org/10.1016/j.apenergy.2020.115875

Kreuter, A. (2009). *All-sky Imaging: a simple versatile system for atmospheric research*. 1091–1097.

Lanham, M. (2021). *Gradient Descent Explained*.

Le Guen, V., & Thome, N. (2020). A deep physical model for solar irradiance forecasting with fisheye images. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, *2020-June*, 2685–2688. https://doi.org/10.1109/CVPRW50498.2020.00323

Li, M., Chu, Y., Pedro, H. T. C., & Coimbra, C. F. M. (2016). Quantitative evaluation of the impact of cloud transmittance and cloud velocity on the accuracy of short-term DNI forecasts. *Renewable Energy*, *86*, 1362–1371. https://doi.org/10.1016/j.renene.2015.09.058

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2020). *Fourier Neural Operator for Parametric Partial Differential Equations*. *2016*, 1–16. http://arxiv.org/abs/2010.08895

Long, C. N., Sabburg, J. M., Calbó, J., & Pagès, D. (2006). Retrieving cloud characteristics from ground-based daytime color all-sky images. *Journal of Atmospheric and Oceanic Technology*, *23*(5), 633–652. https://doi.org/10.1175/JTECH1875.1

Ludwig, J. (2007). *Image Convolution, slides*. 1–8. http://web.pdx.edu/~jduh/courses/Archive/geog481w07/Students/Ludwig_ImageConvolution.pdf

Luiz, E. W., Martins, F. R., Costa, R. S., & Pereira, E. B. (2018). Comparison of methodologies for cloud cover estimation in Brazil - A case study. *Energy for Sustainable Development*, *43*, 15–22. https://doi.org/10.1016/j.esd.2017.12.001

Marquez, R., & Coimbra, C. F. M. (2013). Intra-hour DNI forecasting based on cloud tracking image analysis. *Solar Energy*, *91*, 327–336. https://doi.org/10.1016/j.solener.2012.09.018

Marquez, R., Pedro, H. T. C., & Coimbra, C. F. M. (2013). Hybrid solar forecasting method uses satellite imaging and ground telemetry as inputs to ANNs. *Solar Energy*, *92*, 176–188. https://doi.org/10.1016/j.solener.2013.02.023

Mayank, M. (2020). *Convolutional Neural Networks Explained*. Towarddatascience. https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

Mitchell, T. (1997). *Machine Learning*. http://www.cs.cmu.edu/~tom/mlbook.html

Mölders, N., & Kramm, G. (2014). *Lectures in Meteorology*. http://link.springer.com/10.1007/978-3-319-02144-7

Nabi, J. (2019). *Hyper-parameter Tuning Techniques in Deep Learning*. https://towardsdatascience.com/hyper-parameter-tuning-techniques-in-deep-learning-4dad592c63c8

NASA. (2017). *Earth Atmospheric Layers*. NASA. https://www.nasa.gov/mission_pages/sunearth/science/atmosphere-layers2.html

Neuhoff, K., Batlle, C., Brunekreeft, G., Konstantinidis, C. V., Nabe, C., Oggioni, G., Rodilla, P., Schwenen, S., Siewierski, T., & Strbac, G. (2015). Flexible Short-Term Power Trading: Gathering Experience in EU Countries. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.2639088

Ng, A. (2020). *Improving Deep Neural Networks: Hyperparameter Tuning, Regularization and Optimization*.

Nie, Y., Sun, Y., Chen, Y., Orsini, R., & Brandtb, A. (2020). PV power output prediction from sky images using convolutional neural network: The comparison of sky-condition-specific sub-models and an end-to-end model. *Journal of Renewable and Sustainable Energy*, *12*(4). https://doi.org/10.1063/5.0014016

Nvidia Corporation. (2015). Whitepaper: GPU-Based Deep Learning Inference: A Performance and Power Analysis. *Nvidia*, *November*, 1–12. https://www.nvidia.com/content/tegra/embedded-systems/pdf/jetson_tx1_whitepaper.pdf

Paletta, Q., Arbod, G., & Lasenby, J. (2021). *Benchmarking of Deep Learning Irradiance Forecasting Models from Sky Images -- an in-depth Analysis*. http://arxiv.org/abs/2102.00721

Panadda, K. (2019). *CNN: Flattening*. https://medium.com/@PK_KwanG/cnn-step-2-flattening-50ee0af42e3e

Pothineni, D., Oswald, M. R., Poland, J., & Pollefeys, M. (2019). KloudNet: Deep Learning for Sky Image Analysis and Irradiance Forecasting. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11269 LNCS*, 535–551. https://doi.org/10.1007/978-3-030-12939-2_37

Rapp-Arrarás, Í., & Domingo-Santos, J. M. (2011). Functional forms for approximating the relative optical air mass. *Journal of Geophysical Research Atmospheres*, *116*(24), 1–13. https://doi.org/10.1029/2011JD016706

Reno, M. J., Hansen, C. W., & Stein, J. S. (2012). Global Horizontal Irradiance Clear Sky Models: Implementation and Analysis. *SANDIA REPORT SAND2012-2389 Unlimited Release Printed March 2012*, *March*, 1–66.

Schultz, M. G., Betancourt, C., Gong, B., Kleinert, F., Langguth, M., Leufen, L. H., Mozaffari, A., & Stadtler, S. (2021). Can deep learning beat numerical weather prediction? *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *379*(2194). https://doi.org/10.1098/rsta.2020.0097

Smets, A., Jager, K., Isabella, O., Van Swaaij, R., & Zeman, M. (2016). Solar energy. In *UIT Cambridge Ltd* (Vol. 38, Issue 148). https://doi.org/10.1126/science.148.3673.1127

Spellman, F. R. (2013). The handbook of meteorology. In *Choice Reviews Online* (Vol. 50, Issue 10). https://doi.org/10.5860/choice.50-5607

Sun, Y. G. S. A. B. (2018). Solar PV output prediction from video streams using convolutional neural networks. *Energy & Environmental Science*, *7*.

TensorFlow. (2021a). *Recurrent Neural Networks (RNN) with Keras*. https://www.tensorflow.org/guide/keras/rnn

TensorFlow. (2021b). *Time Series Forecasting*. https://www.tensorflow.org/tutorials/structured_data/time_series

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *2017-Decem*(Nips), 5999–6009.

Yang, D., Alessandrini, S., Antonanzas, J., Antonanzas-Torres, F., Badescu, V., Beyer, H. G., Blaga, R., Boland, J., Bright, J. M., Coimbra, C. F. M., David, M., Frimane, Â., Gueymard, C. A., Hong, T., Kay, M. J., Killinger, S., Kleissl, J., Lauret, P., Lorenz, E., … Zhang, J. (2020). Verification of deterministic

solar forecasts. *Solar Energy*, *210*(November 2019), 20–37. https://doi.org/10.1016/j.solener.2020.04.019

Yang, D., Kleissl, J., Gueymard, C. A., Pedro, H. T. C., & Coimbra, C. F. M. (2018). History and trends in solar irradiance and PV power forecasting: A preliminary assessment and review using text mining. *Solar Energy*, *168*(February), 60–101. https://doi.org/10.1016/j.solener.2017.11.023

Appendix

Additional Cloud Parameters

Table 16: Several characteristic parameters of different clouds and fog. T_c, T_p, w, CR_r, CR_y, LWC and D stand respectively for the cloud lifetime, air parcel resident time in the cloud, mean average velocity, long-wave radiative cooling, wet-adiabatic cooling, liquid/ice water content, vertical cloud depth. Unknown stands for the parameters that are unknown. (Mölders & Kramm, 2014)

Cloud Type	T_c(h)	T_p(s)	w(ms^{-1})	CR_r(Kh^{-1})	CR_y(Kh^{-1})	LWC(gm^{-3})	D(m)
Fog	2-6	10^4	0.01	1-4	0.2	0.05-0.2	100
Stratus/ Strato-cumulus	6-12	10^5	0.1	2	2	0.02-0.25	1,000
Cumulus	0.17-0.5	0.17	3	4	50	0.3-1	1,500
Cumulus congestus	0.34 – 0.75	500	10	4	Unknown	0.5-2.5	5,000
Cumulo-nimbus	>0.75h	400	30	4	Unknown	1.5-4.5	12,000
Altocumulus lenticularis	>10	1,200	15	2	18	0.2	18,000

Deep Learning Terms

Within deep learning nomenclature is used, which is also used in this study. In this section some important nomenclature of deep learning is explained.

Epochs & Batches: Epoch is a parameter the defines how many times the whole training dataset goes through the deep learning model to learn weights. A single epoch results that all samples in the training dataset update the internal model weights one time. When a DL model is trained all the inputs can be handled separately, or in groups of data called batches. If all the datapoints are used separately to train the DL model, then the internal weights are updated after each datapoint resulting in long training times. To reduce training time, DL models often use batches of data to train the internal weights. A batch is a group of datapoints, and a for-loop is used that iterates over each datapoint to make predictions. At the end of each batch the predictions of each datapoint are compared with the expected values and an error is calculated. The calculated error is used to update the internal weights within the DL model. (Brownlee, 2018)

Learning Curves: When a model is trained on a training dataset, the performance of the model increases after each epoch. The performance of the model on the training dataset is evaluated after each epoch. Besides looking at how well the model performs on the training dataset; the performance is often also evaluated for a **validation dataset**. The validation dataset is a part of the training dataset that is not used for training, to look how well the DL model performs on data outside of the training dataset, during the training process. This is done to avoid **overfitting**. Overfitting occurs when the DL model weights are adapted too well for the training dataset but do perform poorly on data outside of the dataset. Besides overfitting, the model can also be **underfitting,** which happens when the weights are not adapted enough to perform well on the training dataset. In Figure 40 an example of underfitting and overfitting is given. When the DL model starts to perform better on the training dataset, compared to the validation dataset it is likely that the model starts to overfit to the data. The learning curves shows the performance of the model on the validation dataset and training dataset after each epoch. Two examples of learning curves obtained in this study are given in Figure 41. (Brownlee, 2019)

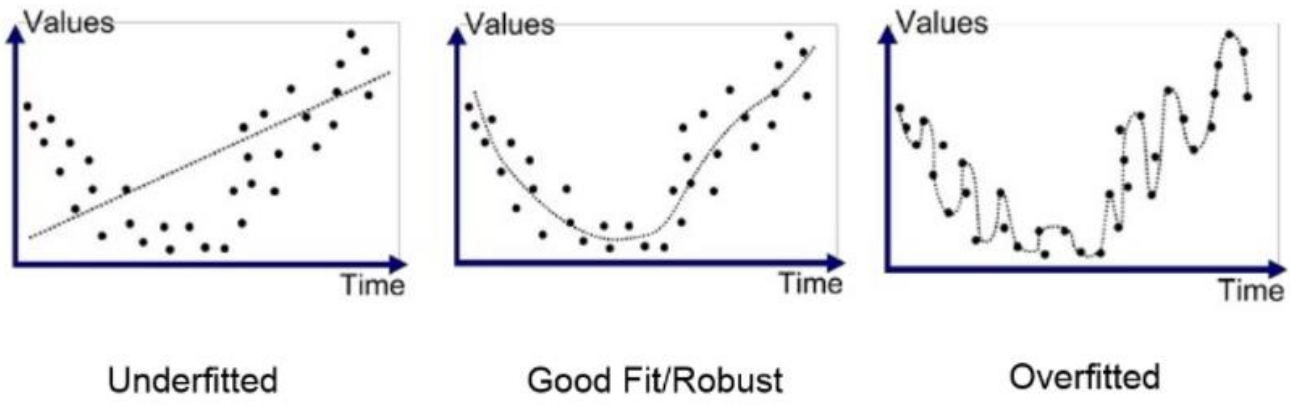


Figure 40: This Figure shows what happens when a model underfits or overfits on data. Obtained from (Bhande, 2018)

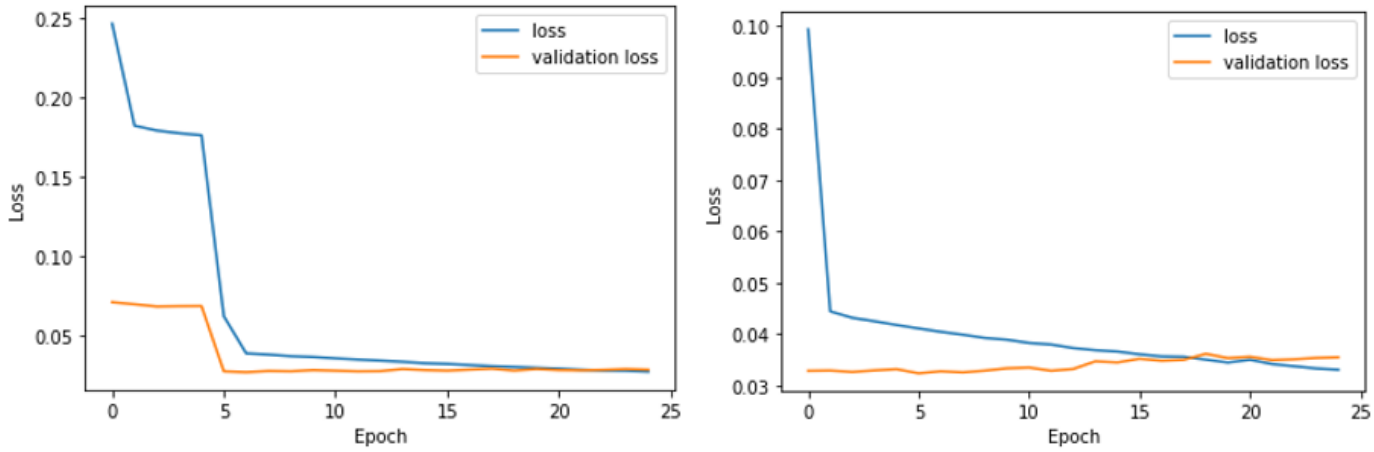


Figure 41: Two examples of learning curves.

Bias & Variance: The bias is the difference of the average prediction and the labelled value. A high bias means that the model is oversimplified resulting on high errors within training and test data. The variance shows the variability of predictions for a datapoint and gives more information about the spread of the data. High variance means that the model is paying a lot of attention to the training data, but not on new data. This results in models with a good performance on training data, but less on testing data. Bias and variance are correlated with overfitting and underfitting as is depicted in Figure 42. (Howe, 2012)

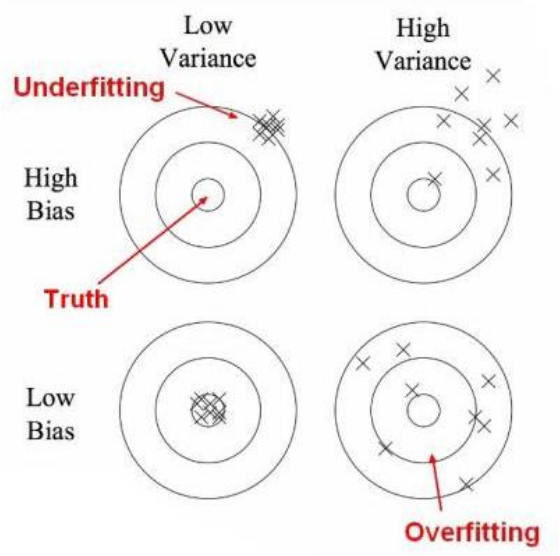


Figure 42: The correlation between variance, bias, underfitting and overfitting is shown. Obtained from (Howe, 2012)**.**

Measurement Instruments

Sky Camera

The All-Sky Imager is collocated with the pyranometers. It captures color images in an 8-bit Red-Green-Blue (RGB) spectrum with 256 color levels with a resolution of 1536 x 1536 pixels at a 1-minute interval. More information about the used All-Sky Imager is absent, but an output example is shown in **Error! Reference s ource not found.**. (Carreira Pedro et al., 2019)

Figure 43: A single output image of the all-sky image. Obtained from (Carreira Pedro et al., 2019)**.**

Pyranometers

Measurements of the GHI and DNI are acquired with a second-generation rotating shadowband radiometer (RSR) from Augustyn, Inc depicted in **Error! Reference source not found.** (Campbell Scientific, n.d.). The R SR contains a shadowband head unit accompanied by two Licor LI-200SZ pyranometers. One of the pyranometers measures the GHI, and the pyranometer with shadowband measures the DHI. Subsequently, the DNI is calculated using the solar zenith angle. The RSR is compared with reference instrumentation over 12 months and showed -1.2% to 1.0% uncertainties for the GHI measurements. The GHI and DHI measurements were recorded with 1-minute average rates. The unit of the irradiance measurements is in W/m^2. (Carreira Pedro et al., 2019)

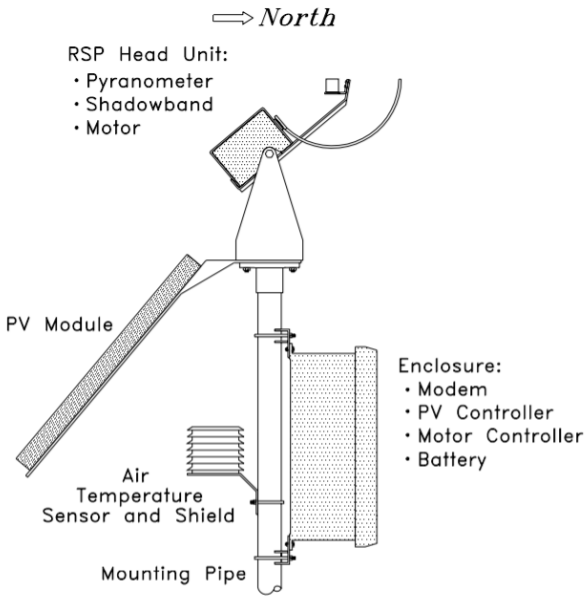


Figure 44: An image of the RSR-2 including components. Obtained from (Campbell Scientific, n.d.)**.**

Weather Transmitter

The meteorological data is recorded using a Vaisala WXT520. The recorded variables and the units are depicted in Table 17. The meteorological variables were also recorded with 1-minute average rates. (Carreira Pedro et al., 2019)

Table 17: Depicts the variables measured with the Vaisala WXT520 and the corresponding units.

Variable	Information	Unit
Pressure	Surface Pressure	Pa
Ambient Temperature	Surface Temperature	K
Relative Humidity	2 m above ground	%
Wind Speed	10 m above ground	ms^{-1}
Wind Direction	10 m above ground	°
Precipitation	Total Precipitation	Kg/m^2

Normalized Meteorological Variables.

In Figure 45, the normalized meteorological data are depicted for 900 minutes. The sudden temperature drops at the 300th and 850th indicate the beginning and end of a day.

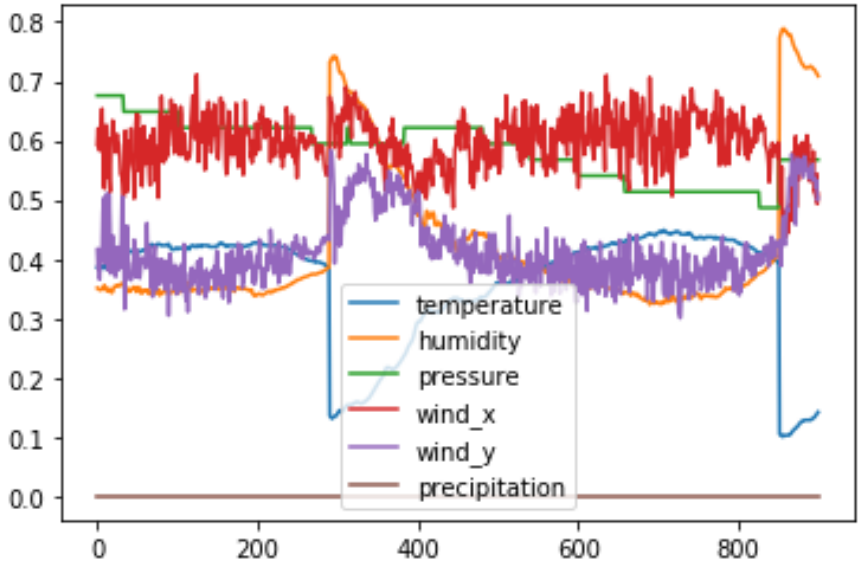


Figure 45: The normalized meteorological data is visualized for 900 minutes.

Trainable Weights

The different DL models consists of several layers, where each layer consists of weights. In Table 18 is depicted how much weights are trained for each layer. The number of trainable weights are determined by the input shape and the amount of neurons or kernels within the layer.

Table 18: This Table shows the trainable weights for each of the different DL layers.

Layer	Output Shape	Trainable Parameters
Convolutional Network		
Input	64,64,3	0
Rescalling	64,64,3	0
Convolution	58,58,20	2,960
Pooling	29,29,20	0
Convolution	25,25,30	15,030
Pooling	12,12,30	0
Convolution	10,10,40	10,840
Pooling	5,5,40	0
Convolution	4,4,50	8,050
Pooling	2,2,50	0
Flatten	200	0
Dense	512	102,912
Dense	64	32,832
Auxiliary Data Encoder		
Input	8	0
Dense	18	162
Dense	18	342
Concatenation		
Concatenate	82	0
Recurrent Network		
RNN	100	18,300
Prediction Decoder (Convolutional Network)		
Dense	64	5,312
Dense	32	2,080
Dense (22x)	1	33
Prediction Decoder (Recurrent Network)		
Dense	64	6,464
Dense	32	2,080
Dense (22x)	1	33

Addition to Example

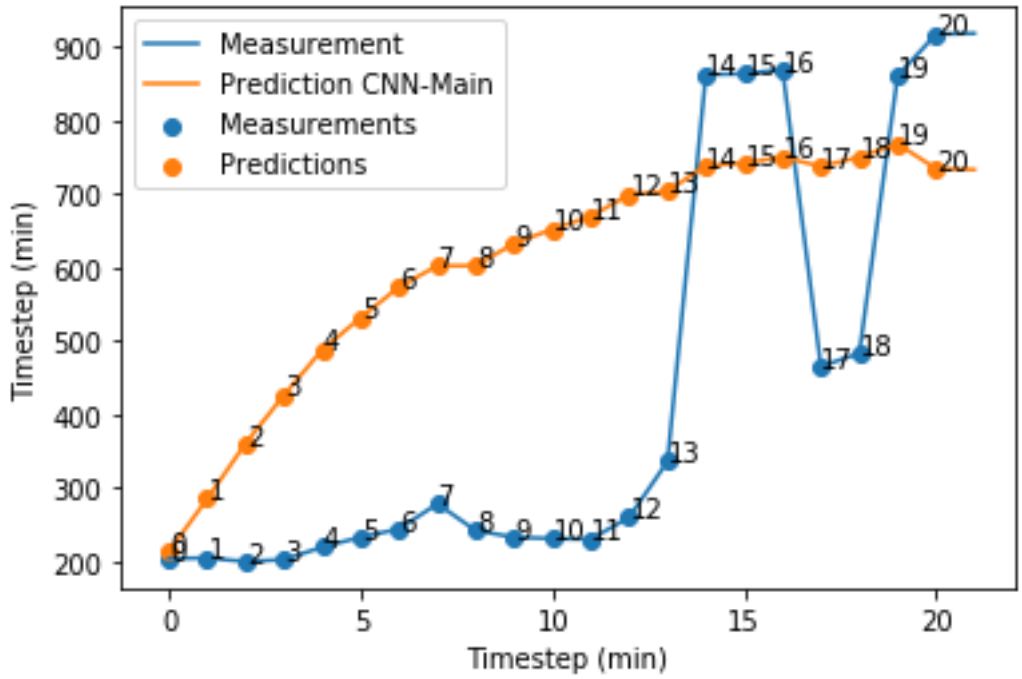
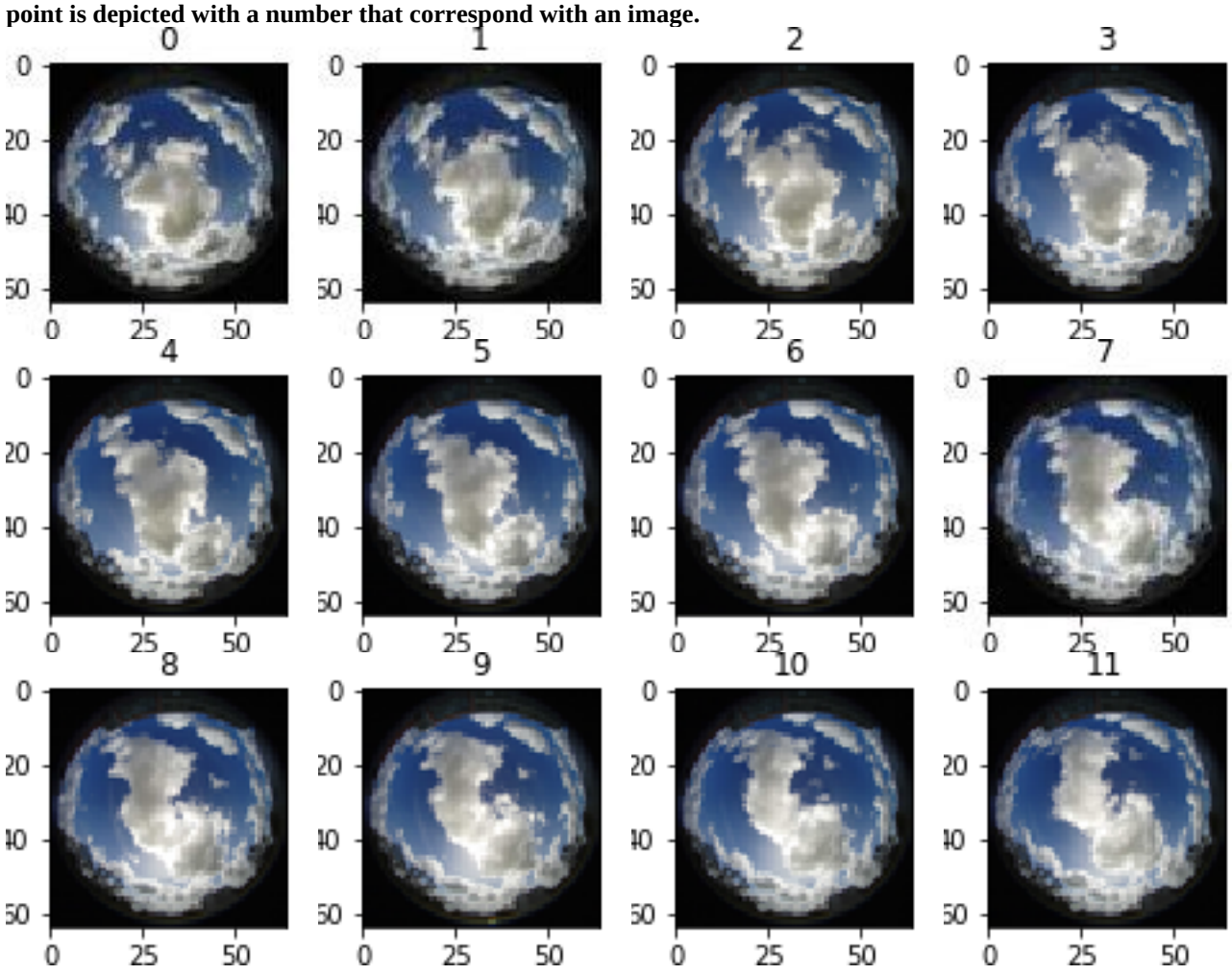


Figure 46: Example of the CNN-main model behavior during the year 2016 on which it performs poorly. Each measured point is depicted with a number that correspond with an image.



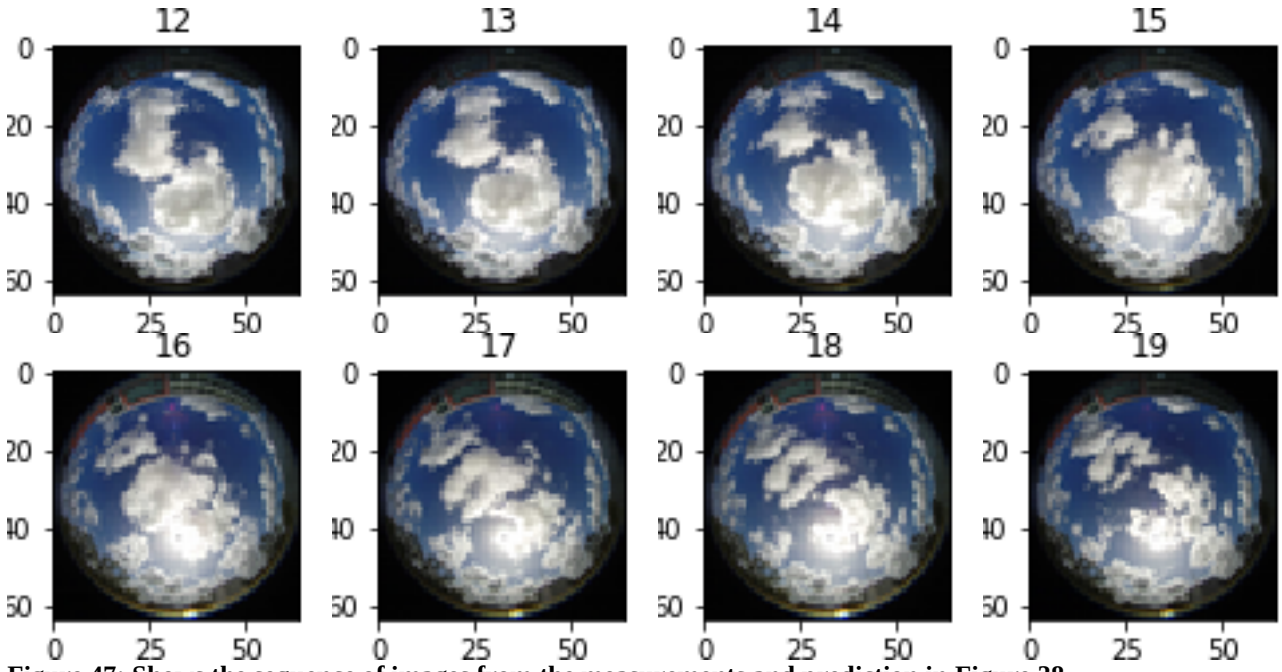


Figure 47: Shows the sequence of images from the measurements and prediction in Figure 38.