Short-Term Solar Forecasting in Germany: Using Satellite Imagery and a Hybrid CNN-2-LSTM Approach

Leonhard Voß

Dissertation written under the supervision of

Professor Pedro Afonso Fernandes

Dissertation submitted in partial fulfillment of requirements for the

M.Sc. in Business Analytics at the Universidade Católica Portuguesa

31st of May 2024

Abstract

Short-Term Solar Forecasting in Germany:
Using Satellite Imagery and a Hybrid CNN-2-LSTM Approach

Leonhard Voß

This thesis presents a novel hybrid convolutional neural network to long short-term memory (CNN-2-LSTM) model for short-term solar forecasting using data from Southeast Germany. The model's performance is compared against a long short-term memory (LSTM) model and a persistence forecast. Satellite-derived cloud masks from the CLAAS-3 dataset by METEOSAT are combined with minute-by-minute global horizontal irradiance (GHI) data from the Fraunhofer Institute's PV-Live dataset. The models predict solar output over 15 minutes, 3 hours, and 6 hours with various input sequence lengths, resulting in 37 models tested.

A new performance metric is introduced, using balance energy prices to calculate the economic impact of the models, providing a practical perspective on financial implications. Results show that the CNN-2-LSTM model significantly outperforms benchmarks at the 15-minute horizon, demonstrating superior accuracy for very short-term predictions. However, its performance at the 3-hour horizon is comparable to the LSTM model, and it lags behind the LSTM model at the 6-hour horizon. These findings highlight the model's effectiveness for very short-term forecasting while emphasizing the need for optimization for specific temporal scales.

The research underscores the potential of hybrid deep learning approaches to enhance short-term solar forecasting accuracy. It offers a cost-effective alternative to more complex systems, valuable for solar energy businesses. The study encourages further exploration into optimizing deep learning models for different forecasting horizons, contributing to advancements in renewable energy management in line with sustainable development goals (SDG).

Keywords: Solar Forecasting; Machine Learning; Deep Learning; Convolutional Neural Networks; Long Short-Term Memory Networks.

Resumo

Short-Term Solar Forecasting in Germany:
Using Satellite Imagery and a Hybrid CNN-2-LSTM Approach

Leonhard Voß

Esta tese apresenta um modelo híbrido de Rede Neural Convolucional para Memória de Longo e Curto Prazo (CNN-2-LSTM) para previsão de energia solar a curto prazo, usando dados do sudeste da Alemanha. O desempenho do modelo é comparado com um modelo de Memória de Longo e Curto Prazo (LSTM) e uma previsão de persistência. Imagens de nuvens da base de dados CLAAS-3 da METEOSAT são combinadas com dados de Irradiância Horizontal Global (GHI) minuto a minuto do Instituto Fraunhofer. Os modelos prevêem a produção de energia solar em horizontes de 15 minutos, 3 horas e 6 horas, com 37 modelos testados.

A tese também introduz uma nova métrica de desempenho que usa preços de energia de balanço para calcular o impacto económico dos modelos. Os resultados mostram que o modelo CNN-2-LSTM supera significativamente as referências no horizonte de 15 minutos. No horizonte de três horas, seu desempenho é comparável ao modelo LSTM, e no horizonte de seis horas, fica atrás do modelo LSTM. Estes resultados destacam a eficácia do modelo para previsões de curto prazo e a necessidade de otimização para escalas temporais específicas.

A pesquisa sublinha o potencial das abordagens híbridas de aprendizagem profunda para melhorar a previsão de energia solar a curto prazo. O modelo oferece uma alternativa custo-eficaz a sistemas mais complexos, sendo uma ferramenta valiosa para empresas de energia solar. O estudo incentiva a otimização de modelos para diferentes horizontes, contribuindo para a gestão de energia renovável em linha com os objetivos de desenvolvimento sustentável (ODS).

Palavras chave: Previsão da produção de energia solar fotovoltaica; Aprendizagem automática (profunda); Redes neurais convolucionais; Redes de memória de curto e longo prazo.

Acknowledgements

My deepest gratitude goes to my advisor, Professor Pedro Afonso Fernandes, for his invaluable guidance and support throughout the course of this research. His expertise and insightful critiques have been instrumental in shaping this thesis, and his encouragement has continually inspired me to strive for more.

Special thanks are due to my brother-in-law, Jonas Voß, for sharing his extensive practical experience in the field of renewable energies, particularly on balance energy pricing. His insights were crucial in the development of the economic performance metric for my models, significantly enhancing the quality of this research.

I am also immensely grateful to my parents, Annette and Friedrich, for their role in shaping my critical thinking, creativity, and deep curiosity. Their guidance has been crucial to my growth, both academically and personally. I appreciate their constant support and encouragement. This thesis is dedicated to them.

Finally, my sincere thanks go to the friends who agreed to proofread this thesis. You only have yourself to blame.

Contents

List of Figures

List of Tables

List of Acronyms

ACF ..... Autocorrelation Function

AOI ..... Area of Interest

ARIMA ..... Autoregressive Integrated Moving Average

CNN ..... Convolutional Neural Network

GHI ..... Global Horizontal Irradiance

ISL ..... Input Sequence Length

LSTM ..... Long Short-Term Memory

MAE ..... Mean Absolute Error

MBEC ..... Mean Balance Energy Cost

MWh ..... Megawatt Hour

NWP ..... Numerical Weather Prediction

PACF ..... Partial Autocorrelation Function

reBAP ..... Uniform Imbalance Price

ReLU ..... Rectified Linear Unit

RNN ..... Recurrent Neural Network

RMSE ..... Root Mean Squared Error

SARIMA .... Seasonal Autoregressive Integrated Moving Average

SDG ..... Sustainable Development Goals

SWAG ..... Sliding Window Aggregation

TSL ..... Target Sequence Length

1 Introduction

The battle against climate change represents a major challenge for modern society. The energy sector, contributing the highest proportion of global greenhouse gas emissions (Lamb et al., 2021), is a prime target for emission reduction initiatives. Renewable energy, especially solar, is crucial for sustainable energy transformation. However, solar energy's variability due to the day-night cycle and weather dependency poses significant problems. These fluctuations make it difficult for electric grids to balance production and consumption, ensuring reliability. Fossil energy sources, like coal and natural gas, require time to ramp up production to compensate for solar fluctuations, complicating grid stability.

Solar energy providers are typically required to provide accurate forecasts of their energy production. Inaccurate forecasts can result in financial penalties, as providers must pay for balance energy to compensate for prediction errors. Thus, improving the forecasting accuracy of solar energy production is crucial for grid stability, environmental sustainability, and the economic viability of solar energy providers.

This thesis focuses on forecasting methods for solar irradiance over short-term horizons—15 minutes, 3 hours, and 6 hours. It investigates the potential of integrating satellite images of cloud cover, a primary driver of solar output variability, into traditional machine learning forecasting methods to enhance the accuracy of solar irradiance predictions. This is achieved through a hybrid deep learning approach: a convolutional neural network (CNN) captures the spatial features of cloud formations from satellite images, and a long short-term memory (LSTM) network captures the temporal dynamics of these images. A second LSTM captures the temporal dynamics of numerical solar irradiance inputs. Both branches form the proposed CNN-2-LSTM architecture, aiming to improve solar irradiance forecast precision.

The geographical focus is Germany, a global leader in solar power production (Kaliappan et al., 2019). Specifically, it uses 2022 data from Baden-Württemberg, the state with the highest solar irradiance in Germany (Parvanyan and Fox, 2019), making it ideal for solar operators. This thesis introduces a novel economic performance metric for model evaluation, estimating mean payments incurred for balance energy due to forecasting errors, emphasizing the business relevancy of accurate predictions.

This thesis aims to validate the effectiveness of the CNN-2-LSTM model against benchmark persistence models at different forecasting horizons and model configurations. By refining the understanding of solar forecasting, it enhances grid management and operational efficiency of renewable energy systems, improving the economic viability of solar energy and supporting a sustainable energy future in line with the Sustainable Development Goals (SDG) (United Nations Department of Economic and Social Affairs, 2023).

2 Related Work

Solar forecasting[1] has attracted an extensive amount of scholarly attention, resulting in a substantial body of research that covers a wide array of different approaches and forecasting methods. Several classifications of these methods have already been proposed in review literature (e.g., Inman et al. (2013), Diagne et al. (2013), Antonanzas et al. (2016), Yang et al. (2018), and Ahmed et al. (2020)). For the purpose of this thesis, widely used solar forecasting methods are clustered into the following classes: statistical, numerical weather prediction (NWP), and image-derived. Image-derived methods are further differentiated into sky-image-based methods and satellite-image-based methods. Each method has different characteristics in terms of applicability, time horizons (Figure 1), complexity, and costs. In the following sections, each of the method classes is briefly introduced, their main characteristics are described, and their scientific state of the art is reviewed.

Figure 1: Applicability of Solar Forecasting Methods Based on Forecasting Horizons

2.1 Statistical Solar Forecasting

Statistical solar forecasting refers to methods that are based on historical data and statistical models (Yang et al., 2018). The underlying data are often time series of key meteorological measurements such as solar irradiance, temperature, and humidity as well as historical data of solar power generation itself. Statistical analysis is then performed on these observations to uncover past patterns and extrapolate them into the future. Compared to the other solar forecasting methods presented further below, statistical solar forecasting is characterized by its relative low complexity, low computational requirements and its ease of implementation. In addition, the required input data is readily available at high temporal resolutions and low cost.

A variety of statistical methods are used to generate the forecasts. One of them being decomposition methods, which break the different time series datasets into their underlying components (e.g., trend, seasonality, cyclical patterns, and noise), forecast each one

[1]For simplicity and due to their close conceptual and technical relationship, both solar power generation forecasting and solar irradiance forecasting are referred to interchangeably as *solar forecasting* for short.

of them, and finally generate a prediction by combining all component forecasts.
Majumder et al. (2020) and Monjoly et al. (2017) both found that decomposition methods, particularly multidimensional empirical mode decomposition, can significantly improve the accuracy of solar power and radiation forecasting. Wang et al. (2018) and Zhang et al. (2018) further enhanced these methods by combining them with machine learning models, such as support vector machines and least-squares support vector machine, resulting in superior performance. Prasad et al. (2020) and Majumder et al. (2017) complemented these findings to weekly and short-term forecasting, respectively, with the latter achieving high accuracy using a hybrid empirical mode decomposition and extreme learning machines method. Malvoni et al. (2017) and Behera and Nayak (2020) explored the use of data preprocessing techniques and a three-stage approach, respectively, to further improve the accuracy of solar forecasting.
Similarly, a range of studies have explored the use of autoregressive integrated moving average (ARIMA) and seasonal ARIMA (SARIMA) models in forecasting solar energy generation. Atique et al. (2019) and Shadab et al. (2020) both successfully applied ARIMA and SARIMA models respectively to predict solar energy generation, with Shadab et al. (2020) achieving high forecast accuracy. However, Shoaga et al. (2022) found significant deviations in their SARIMA model for hydroelectric and solar production in Rwanda. Kushwaha and Pindoriya (2017) and Mukaram and Yusof (2017) further demonstrated the effectiveness of SARIMA models in very short-term solar generation forecasts and solar radiation forecasts, respectively. Atique et al. (2020) compared ARIMA and machine learning techniques, finding that support vector machines outperformed ARIMA in solar generation prediction.
Moreover, a number of studies deploy different more advanced machine learning techniques for statistical forecasting, achieving promising results. Shao et al. (2016) and Vennila et al. (2022) both developed multi-model blending approaches, with the latter emphasizing the benefits of a hybrid model. Datta et al. (2023) and Subramanian et al. (2023) focused on the use of machine learning techniques, with the former proposing an ensemble trees-based model and the latter achieving high accuracy using support vector machines, random forests, and gradient boosting. Jebli et al. (2020) and Shahid et al. (2020) discussed the potential of machine learning in this field, with the latter achieving improved accuracy using random forest and ridge regressor models. However, Zhao and Tian (2022) found that a traditional time series model outperformed a machine learning-based LSTM algorithm in predicting solar power generation.

2.2 Numerical Weather Prediction

NWP is concerned with formulating and solving mathematical equations that attempt to replicate the underlying physics of weather (Coiffier, 2011). Its application for solar

forecasting is intuitive: use the predicted values for relevant parameters (e.g., solar irradiance or cloud coverage) at a specific time and location to derive solar forecasts based on them. The underlying calculations of an NWP model are of such complexity that in most cases super-computing facilities are required to achieve timely outputs (Bauer et al., 2015). The associated high costs dictate that NWP-based forecasts can only be produced a few times a day and published by supranational weather services (typically at 6 hour intervals) (Zhang et al., 2022). This precludes their use for a solar forecasting time horizon of less than 6 hours and limits its applicability for intra-day solar forecasting.
A range of studies have explored the use of NWP methods for solar energy forecasting. Chen et al. (2017) and Aler et al. (2015) both proposed models that leverage NWP data, with Chen using gaussian process regression and convolutional networks, and Aler comparing the performance of support vector machines and gradient boosted regression. Martin et al. (2015) and Pelland et al. (2011) focused on feature selection and post-processing methods to improve the accuracy of solar energy forecasts. P. Mathiesen et al. (2013) evaluated the accuracy of NWP models for solar irradiance forecasting, finding that the global forecast system provided the best forecasts for the continental United States. Andrade and Bessa (2017) and Verbois et al. (2018) both demonstrated the potential of combining NWP data with advanced algorithms, with Andrade and Bessa (2017) achieving significant improvements in point and probabilistic forecast skill, and Verbois et al. (2018) significantly outperforming other models for day-ahead forecasts of solar irradiance.

Radiative transfer models, closely related to NWP, simulate the propagation and interaction of solar radiation within the Earth's atmosphere and surface (Matricardi et al., 2004). These models consider absorption, scattering, emission, reflection, and transmission of solar radiation, providing insights into how atmospheric conditions affect solar irradiance. When integrated into NWP systems, radiative transfer models enhance solar energy predictions by accounting for the impacts of clouds, aerosols, and other atmospheric constituents, thereby improving forecast reliability.
Numerous studies have investigated the application of radiative transfer models in solar forecasting. Hernandez-Travieso et al. (2014) and Mazorra-Aguiar and Díaz (2018) both used statistical methods, with the former achieving a mean average error of 0.04 kilowatts per hour. Dellino et al. (2015) proposed the use of transfer function models for energy production forecasting in a solar plant, while Hamilton et al. (2016) developed solar forecasting models using neural networks and model trees. Silva et al. (2015) applied data mining for short-term solar irradiance forecasting, and Paoli et al. (2009) used ad-hoc data preprocessing and neural networks for daily prediction of global solar radiation.

2.3 Image-Derived Solar Forecasting

The use of image data for solar energy forecasting has proven to be a promising approach that uses the wealth of information available from satellite and ground-based imaging systems. In recent years, researchers and practitioners have increasingly turned to these methods to improve the accuracy and granularity of solar irradiance predictions. By utilizing image processing techniques such as cloud motion vectors and CNNs, innovative predictive models have been developed to capture spatial and temporal variations in solar irradiance patterns. In the following, image-derived solar forecasting is reviewed, divided into sky- and satellite-based imagery.

2.3.1 Sky-Image-Based Methods

Sky-image-based solar forecasting is a scientific approach that uses sky imaging techniques to predict solar radiation. Sky images are photographs of the sky taken from ground-based cameras that capture real-time visual information for meteorological uses. By analyzing cloud cover, aerosol content and other atmospheric parameters captured in sky images, this prediction method aims to estimate the incoming solar radiation reaching the Earth's surface. By integrating advanced image processing algorithms and atmospheric models, it provides valuable insights into the (super-) short-term fluctuations in solar energy availability.

T. Schmidt (2017) and Lasanthika H. Dissawa et al. (2020) both developed models that use cloud motion tracking to predict solar irradiance, with the latter achieving mean bias error percentages of 11-12%. V. Jayadevan et al. (2012) and Al-lahham et al. (2020) (2020) focused on the use of machine learning, with the latter achieving competitive results with less computational complexity. Kong et al. (2020) and Wen et al. (2021) proposed hybrid and deep learning models, respectively, with the latter achieving a skill score of 17.7% in photovoltaic power ramp-rate control. Kamadinata et al. (2019) and Zhen et al. (2017) both used artificial neural networks, with the former achieving a minimum root mean square error (RMSE) of 143. These studies collectively demonstrate the potential of sky-image based solar energy forecasting, with various models and techniques showing promising results.

2.3.2 Satellite-Image-Based Methods

Satellite images are well suited for solar forecasting for a variety of reasons. With cloud coverage being the strongest and most relevant cause of solar irradiance fluctuations, it is intuitive to utilize satellite images since they yield an unobstructed view of the presence of clouds in the atmosphere. In addition, they offer a comparatively high resolution in both the spatial and temporal dimension and cover most of Earth's surface at once.

One of the first scientific contributions to solar energy forecasting using satellite images was made by Hiser and Senn (1980). They used imagery from geostationary satellites and achieve a general solar irradiance map at a 5 kilometer resolution. Yet, their approach is only applicable for overall seasonal estimates, rather than real-time forecasting. Based on this, Hammer et al. (1999) present a statistical method to forecast solar radiation for time scale from 30 minutes up to 2 hours and a practical application example for solar energy forecasting. They achieve this by detecting the motion of cloud structures visible in a series of satellite images and extrapolate the future cloud coverage based on this.
Marquez et al. (2013) propose a hybrid solar forecasting method that pioneers the use of artificial neural networks for satellite image analysis and is able to predict global horizontal irradiance at time horizons of 30 minutes up to 2 hours and achieve satisfactory results. Similarly, Dong et al. (2014) propose a hybrid model that uses exponential smoothing in combination with artificial neural networks. They found that their model outperforms traditional forecasting techniques when tested at an hourly forecast horizon.
Cros et al. (2014) employ a statistical phase correlation algorithm to obtain cloud motion vectors from subsequent geostationary satellite images. Their approach is found to have a 21% lower RMSE compared to benchmark persistence forecasting at a 4-hour prediction period. Jang et al. (2016) use support vector machines to predict atmospheric motion vectors for clouds. They trained their model on historical satellite images spanning 4 years and achieve satisfactory results compared to time-series models and artificial neural network approaches. Yu et al. (2020) present a solar power forecasting model that entails a cloud amount forecasting network that was trained on a set of historical satellite images. In addition, their model adopts convolutional self-attention that is capable of capturing historical features in the data. They show that cloud amount forecast network reduces the mean absolute percentage error by 22.5% when compared to a prediction model without. A similar approach is used by Si et al. (2020). They developed a hybrid method for predicting solar irradiance at time horizons of up to 4 hours. The model combines satellite imagery with general meteorological information. A modified CNN is used to derive cloud cover from the satellite images which is then enriched with the meteorological data to obtain a solar forecast. Their approach shows satisfactory results compared to benchmark models.
Prasad and Kay (2021) utilize near-real time satellite data and cloud motion vectors to achieve solar energy predictions at a 5-minutes ahead time horizon. Although their model outperforms persistence forecasting methods, it does not achieve the desired forecasting skill under live conditions. Cheng et al. (2022) developed a deep-learning model that is trained directly on satellite images for specific regions of interest. It is capable of delivering multi-step-ahead predictions. When compared to related studies, the model outperforms persistence forecasting models. It is found to be suitable for forecasting time horizons of up to three hours.

2.4 Identified Gap

NWP-based approaches, and image-derived methods. However, a notable gap exists in the operationalization of hybrid deep learning architectures, specifically the CNN-2-LSTM model, which integrates ground-based observations with satellite imagery for solar irradiance prediction. This model leverages CNNs for spatial feature extraction and LSTMs for sequence prediction, but its application across varying forecasting horizons remains underexplored.

Additionally, while extensive research exists on global solar forecasting, there is a lack of localized studies for regions like Baden-Württemberg. Local geographic and meteorological conditions can significantly influence solar irradiance patterns, necessitating region-specific models for accurate forecasting.

Current literature also tends to focus on very short-term (up to an hour) or long-term (day-ahead) predictions, with intermediate time horizons (e.g., 15 minutes, three hours, six hours) often overlooked. These time frames are crucial for grid management and energy trading.

This thesis addresses these gaps by implementing a CNN-2-LSTM model for solar irradiance forecasting, integrating solar observations with satellite imagery. It specifically targets Baden-Württemberg, tailoring the model to local conditions. By exploring multiple forecasting horizons, this study provides a comprehensive understanding of the model's performance across different time scales, offering valuable insights for energy systems management.

3 Data

The data for model training in this thesis combines two distinct datasets: satellite image data and solar irradiance observations. The following sections describe both datasets, detail the preprocessing steps for each, and illustrate the data merging, splitting, and sequencing processes.

3.1 CLAAS-3 Cloud Property Dataset

The satellite imagery for the solar forecasting model of this thesis is obtained from the CLAAS-3 dataset (Meirink et al., 2022) created by the European Organisation for the Exploitation of Meteorological Satellites. It is derived from measurements captured by a combined visible and infrared light sensor onboard METEOSAT satellites.

The dataset consists of multiple subsets, which are created by processing raw satellite data in different ways to derive detailed information about cloud properties for specific scientific uses. These subsets include imagery for cloud type, cloud top temperature, cloud top height, cloud optical thickness, and cloud masks amongst others. In this thesis, only the latter subset is used. The cloud masks are generated by applying an algorithm to raw satellite data to predict cloud presence for each pixel in the image. The output, illustrated in Figure 2, is a grayscale image where pixels containing clouds are assigned a value of zero (black), and pixels without clouds are set to one (white).

Figure 2: Unedited Example Cloud Mask Images from the CLAAS-3 Dataset

The cloud mask subset of the CLAAS-3 datasets spans back to 2004, is regularly updated, and has a temporal resolution of 15 minutes. For this thesis, only data from 2022 is used, yielding a total of 35,040 images. Furthermore, the images in the dataset geographically cover half of Earth's surface (latitude: -81.30° S to 81.30° N, longitude: -81.25° W to 81.25° E) at a spatial resolution of 3 kilometers. For this thesis, the area of interest (AOI) is the federal state of Baden-Württemberg. Therefore, the raw images are

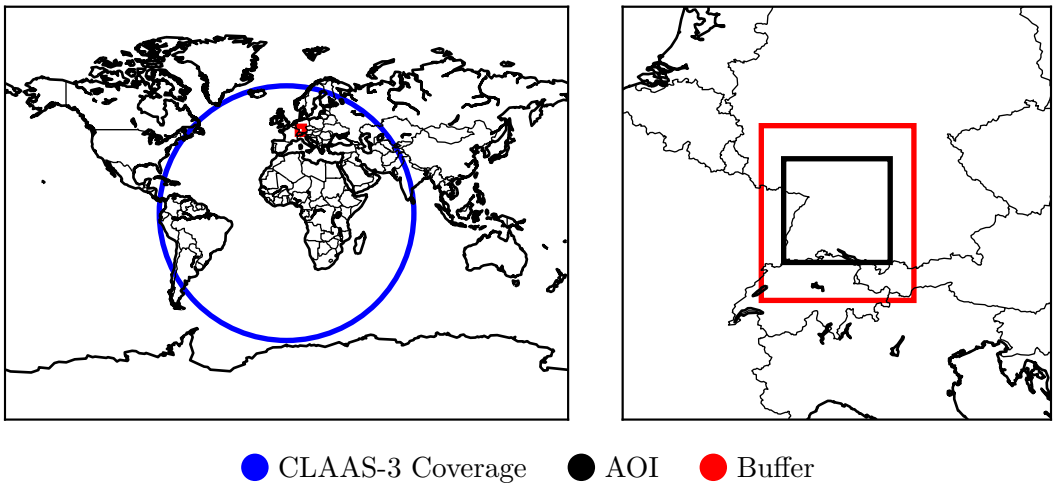

Figure 3: Illustration of CLAAS-3 Coverage, AOI, and Buffer Extents

cropped to the extreme points of the AOI, adding a 100 kilometer buffer in each direction (latitude: 46.68° N to 50.69° N, longitude: 6.89° E to 11.37° E) (Figure 3). The buffer is introduced under the assumption that the proposed model can extract useful information from the presence of clouds outside of the AOI. More specifically, the premise is that cloud cover does not appear spontaneously, but enters from outside the AOI.

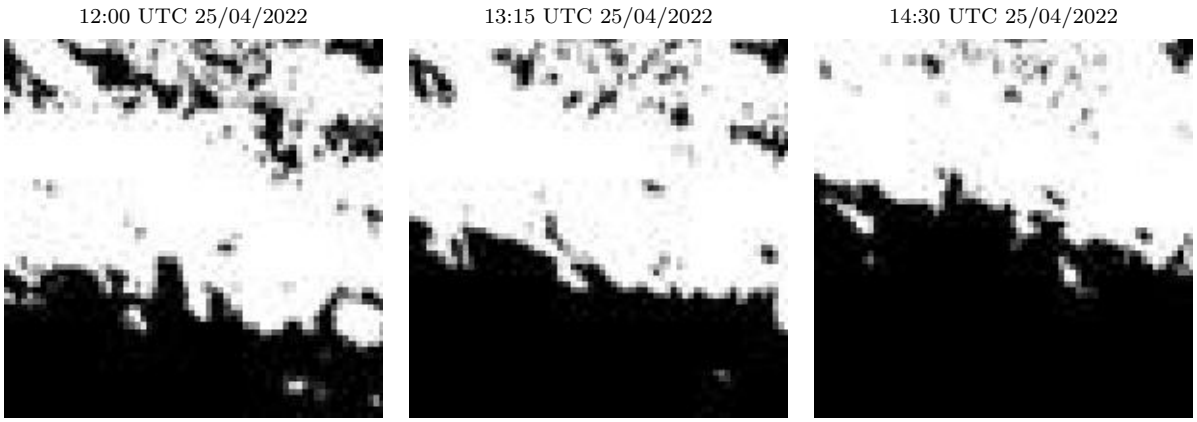

Figure 4: Cropped, Resized, and Normalized Example Images of AOI and Buffer

The resulting images have a resolution of 272 by 369 pixels with a single grayscale channel. Due to computational constraints and efficiency they were resized to a final resolution of 64 by 64 pixels and normalized (by dividing each pixel value by 255) prior to model training. As observable in Figure 4, the pixels in the processed images are no longer solely black or white but have also have shades of gray. This effect is due to rescaling and the associated image interpolation and retains some visual information lost to rescaling.

3.2 PV-Live Solar Irradiance Dataset

The numerical solar irradiance data for the proposed model stems from the PV-Live dataset created by the Fraunhofer Institute for Solar Energy Systems (Dittmann et al., 2024). It contains minute-by-minute measurements of global horizontal irradiance (GHI)[2] from 40 points primarily spread across the federal state of Baden-Württemberg in southern Germany. The data set currently covers the period from September 2020 to August 2023. Only the 2022 values are used in this thesis.

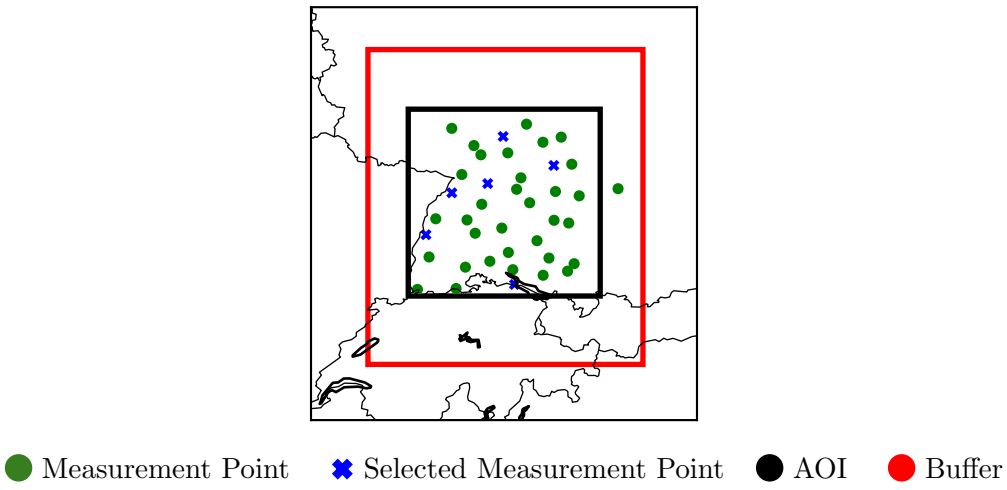

Figure 5: Measurement Points of PV-Live Dataset

Furthermore, due to scope and computational restraints, a selection of six out of 40 measurement points was made. During the selection process, the main focus was on identifying the measurement points with as few missing values as possible while ensuring satisfactory geographical coverage over the AOI.

ID	Name	Lat. N	Long. E	NaN	Mean GHI	Max GHI
06	Mahlberg	48.280	7.787	1.257%	155.54	1,358
25	Eberbach	49.465	8.987	1.491%	144.01	1,377
28	Pforzheim	48.899	8.746	1.513%	150.26	1,319
30	Konstanz	47.674	9.163	2.165%	158.26	1,377
35	Schwäbisch Hall	49.117	9.774	2.545%	149.71	1,385
36	Baden-Baden	48.787	8.189	1.529%	152.46	1,379

Table 1: Summary Statistics for Selected Points in 2022 without Preprocessing

As a first step, missing values recorded during nighttime were set to zero, assuming no solar irradiance. All other missing data points were imputed with the mean of the available

[2]GHI represents the total amount of sunlight reaching a horizontal surface on Earth, expressed in watts per square meter. This measurement includes both direct sunlight and diffuse horizontal irradiance. GHI is typically recorded using devices known as pyranometers.

measurements at the corresponding time. Subsequently, the data was aggregated into 15-minute intervals by calculating the mean of every 15 consecutive measurements. The date and time of each observation was transformed into categorical variables with a separate column for month, day of month, hour of day, and minute of hour. Finally, the sum of all six measurement points per time step was generated and saved as its own column (*ghisum*) in the data. This sum will serve as the main target variable for the proposed model.

After the initial data cleaning and preliminary preprocessing, exploratory data analysis was performed on the numerical data to obtain a better understanding of the data at hand and guide model development.

Variable	**n**	**Mean**	**SD**	**Min.**	**25%**	**Med.**	**75%**	**Max.**
ghisum	35,040	900.65	1391.48	0.00	20.86	20.86	1366.42	5878.52
month	35,040	6.53	3.45	1.00	4.00	7.00	10.00	12.00
day	35,040	15.72	8.80	1.00	8.00	16.00	23.00	31.00
hour	35,040	11.50	6.92	0.00	5.75	11.50	17.25	23.00
minute	35,040	22.50	16.77	0.00	11.25	22.50	33.75	45.00

Table 2: Summary Statistics for Numerical Data after Initial Cleaning

A prominent characteristic of the *ghisum* variable, apparent from both the summary statistics (Table 2) and the histogram (Figure 6), is its pronounced left skew. This skewness primarily arises because a significant portion of the observations are zero, attributable to the lack of solar irradiance during nighttime hours. Specifically, 16,274 or 46.66% of all observations register as zero. Additionally, the range of *ghisum* is notably broad, with the maximum value reaching 5,878.52 and a standard deviation of 1,391.48. While the marked left skewness is not problematic, as the proposed model does not require normally distributed data, the extensive range of the data should and will be addressed using data scaling techniques (see Section 3.3).

Upon examining the monthly averages of *ghisum* (Figure 7), a clear seasonal pattern is discernible. Solar irradiance is heavily correlated with meteorological seasons, a correlation that is well-documented and expected due to variations in solar angle and daylight duration. The data shows a gradual increase in the monthly average GHI starting in January, culminating in a peak in July. Following this zenith, there is a steady decline in irradiance, leading to the annual minimum in December.

The daily averages of *ghisum* not only reinforce the seasonality previously observed in the data but also highlight the significant variability of GHI. This variability is evident from the substantial fluctuations in GHI values even between consecutive days. Additionally, the mean daily GHI profiles per month (Figure 8) reveal another seasonal aspect of the time series data. Not only is there significant variance in the mean daily peaks of solar irradiance across different months (with the highest peaks occurring in July and

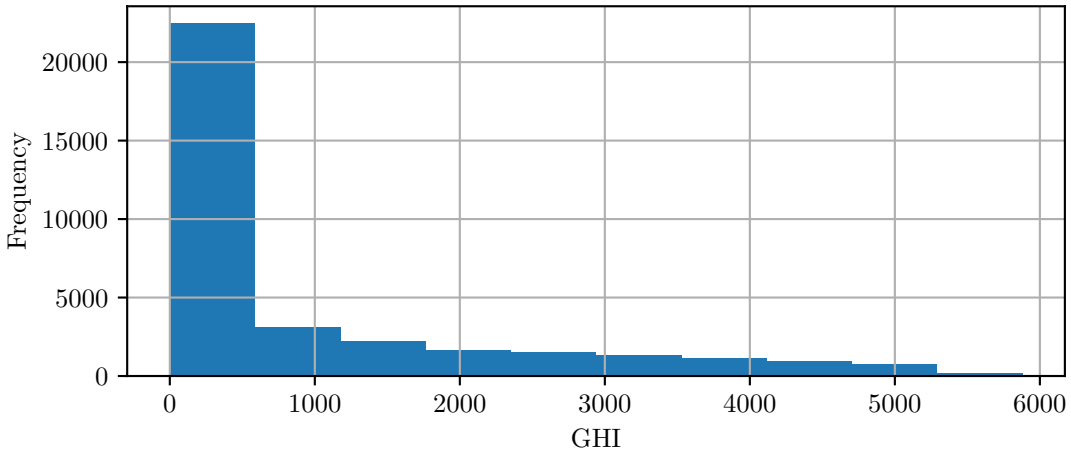

Figure 6: Distribution of *ghisum*

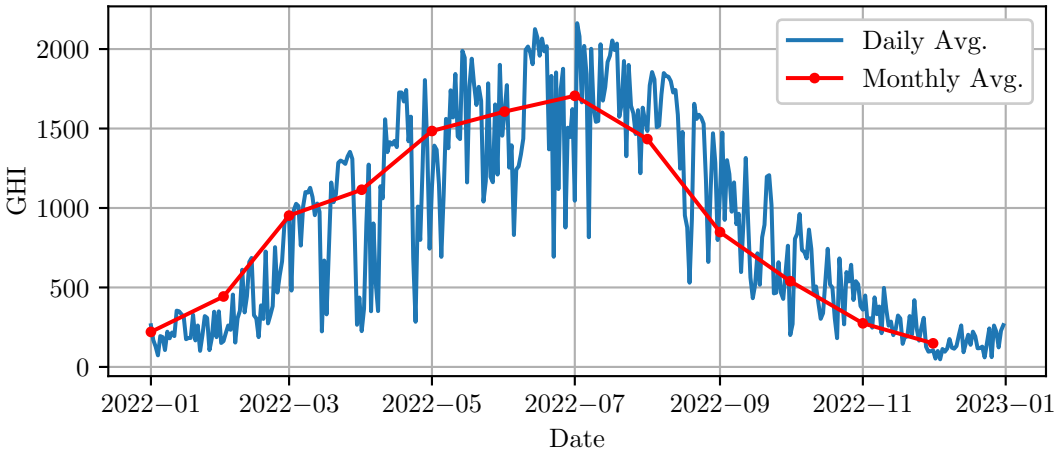

Figure 7: Daily and Monthly Average of *ghisum*

the lowest in December), but the duration of solar irradiance each day also varies considerably. In December, the average duration of daily solar irradiance is approximately 8 hours, whereas in June, it extends to nearly 16 hours[3]. Interestingly, the irradiance peaks during the winter months tend to be smoother, while those in the summer months display more variability, reflecting the more dynamic atmospheric conditions (e.g., cloud cover) experienced during warmer periods.

The autocorrelation function (ACF) plot demonstrates a pronounced sinusoidal pattern, oscillating between positive and negative values. This pattern diminishes gradually over time before repeating, indicative of a strong seasonal influence within the data, recurring approximately every 96 lags (or about 24 hours). The amplitude of the autocorrelations is highest at lag zero, which is expected since a dataset is perfectly correlated with itself at zero lag. The gradual decrease in correlation over time highlights the last-

[3]Please note that the time stamps for the data are formatted in UTC+0, whereas the local time zone in Germany is UTC+2.

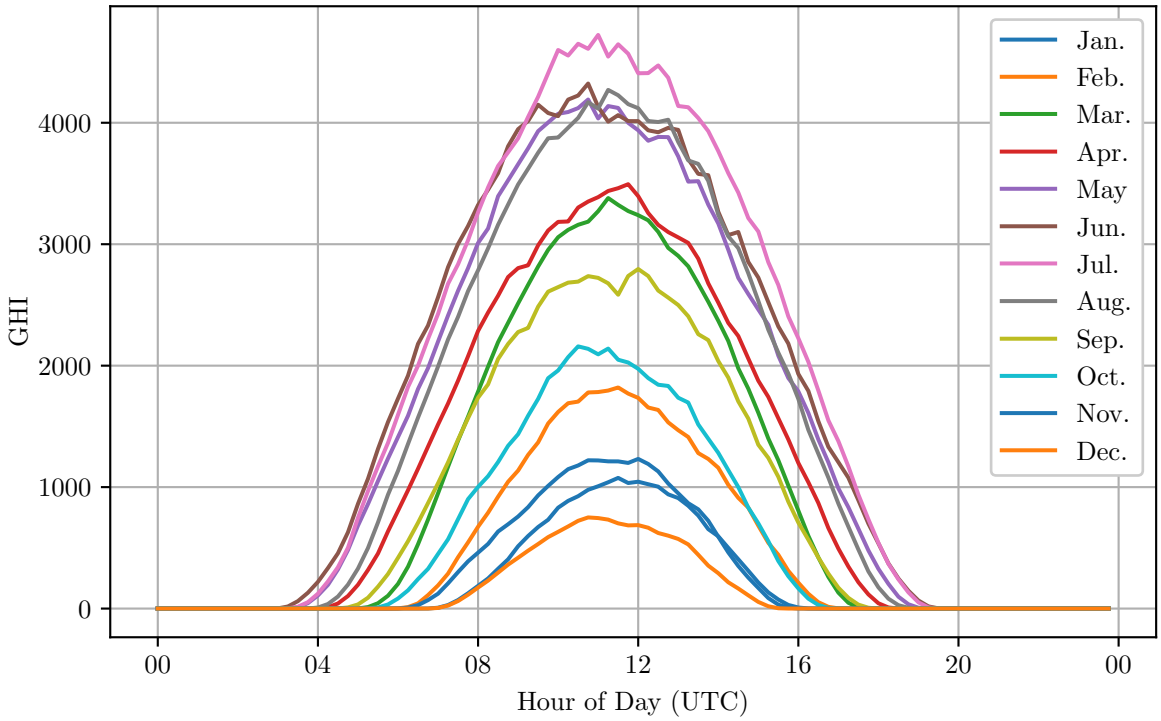

Figure 8: Mean Day *ghisum* Profiles per Month

ing impact of past values over several intervals. Conversely, the partial autocorrelation function (PACF) plot reveals a significant spike at the first lag, which then quickly reduces to near-zero or insignificant values for subsequent lags. This pattern suggests that each observation in the series predominantly depends on its immediate predecessor, with minimal influence from further past values.

To summarize, the exploratory data analysis conducted on the PV-Live dataset has provided comprehensive insights into the temporal dynamics and characteristics of solar irradiance as measured across selected points in Baden-Württemberg, Germany. The summary statistics and visual analyses reveal a pronounced left skew in the GHI data, primarily due to a significant number of zero observations during nighttime, coupled with a high variability in GHI values. Seasonal patterns are evident with monthly and hourly periodicities. Solar irradiance is peaking in July and reaching its lowest in December, correlating strongly with meteorological seasons due to variations in solar angles and daylight hours. Additionally, the daily GHI profiles per month illustrate both the variance in daily peaks and the duration of solar irradiance, which varies significantly across the year. The ACF and PACF plots further underscore the seasonal influence and the autocorrelation structure within the dataset, providing valuable guidance for model design and forecasting efforts. This exploratory data analysis has laid the groundwork for more detailed modeling and has helped identify the key features of the dataset that will influence subsequent analyses and predictions.

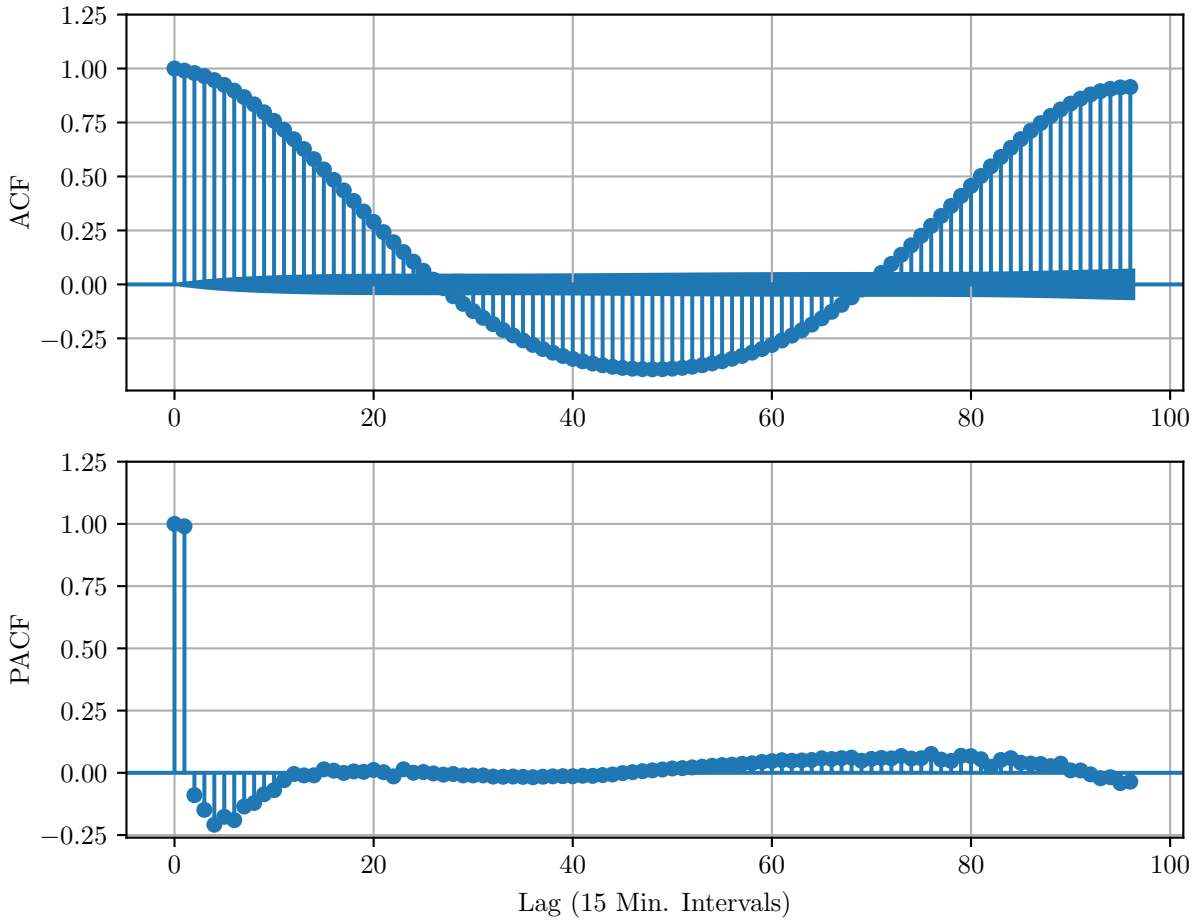

Figure 9: ACF and PACF for *ghisum*

3.3 Data Preparation for Model Training

To enable effective model training, specific modifications must be performed on the aforementioned datasets. These modifications include several additional data preprocessing steps. Namely data scaling, data splitting, and data sequencing. All of which are outlined in this section.

To begin, data scaling was implemented for all numerical input features[4]. Data scaling can significantly enhance model performance by ensuring that all input features are treated equally by the model, preventing any single feature from dominating due to its larger numerical magnitude. Additionally, it improves model robustness to outliers by normalizing values within a specified range, leading to more accurate and reliable predictions (Ahsan et al., 2021).

Specifically, all numerical features were scaled using a min-max scaler, defined as:

$$X_{scaled} = \frac{X - Min(X)}{Max(X) - Min(X)} \tag{1}$$

where:

X = Numerical input variable
$Min(X)$ = Minimum value of X
$Max(X)$ = Maximum value of X

Min-max scaling is a technique that rescales data to a fixed range, typically [0, 1]. In this method, the smallest value in the original dataset is mapped to 0, and the largest value is mapped to 1, with all other values scaled proportionally within this range. This technique is straightforward to apply and highly interpretable, even without reversing the scaling. One of the main reasons for using min-max scaling in this context is that the GHI data does not contain significant outliers, which would necessitate more complex techniques. Additionally, min-max scaling preserves a key characteristic of the data: the high number of zero values. In contrast, a robust scaling approach might produce negative values during model training, which are nonsensical in the context of solar forecasting. Moreover, the proposed architecture includes a technique to ensure that the predicted output values are non-negative. This could not be realized with a scaling method that yields negative values. Thus, min-max scaling is an appropriate and effective choice for this application.

After data scaling, the processed satellite imagery from the CLAAS-3 dataset was seamlessly merged with the processed numerical data from the PV-Live dataset. This combination was achieved by aligning both datasets along the time axis, ensuring that each set of satellite images corresponded precisely with its numerical data counterpart at each time point.

[4]Note that the image data was already normalized (i.e., scaled) during creation of the image dataset (see Section 3.1)

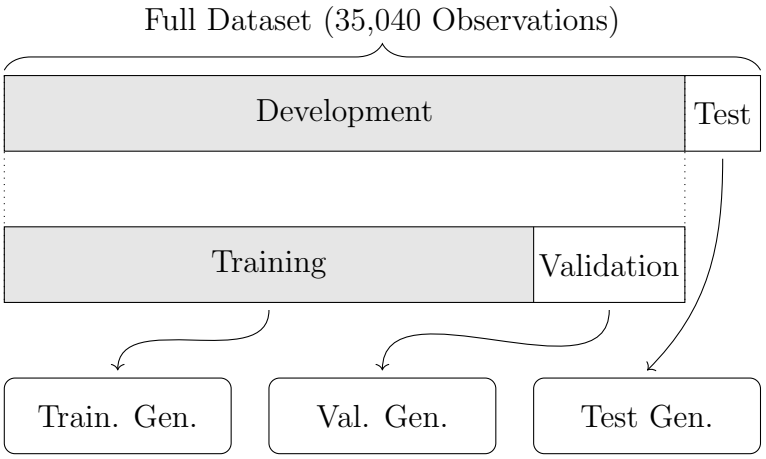

Figure 10: Illustration of Data Splitting

Subsequently, the data was split into a development set and a test set, with the development set comprising 90% of the data (31,526 data points) and the test set comprising the remaining 10% (3,504 data points). The test set is reserved exclusively for performance evaluation after model training, providing a reliable estimate of the model's effectiveness on unseen data. The development set was further divided into a training set (70% of the original data) and a validation set (20% of the original data), ensuring a robust training process. The model is trained on the training set, and its performance is continuously monitored and fine-tuned based on results from the validation set, allowing for iterative improvements and optimization during the training phase. This approach ensures that the model generalizes well and avoids overfitting, leading to better performance on the test set and potentially in a real world environment.

The different data subsets were fed to the model using separate data generators to efficiently handle large datasets and preventing data leakage by streaming the training, validation, and test sets in manageable batches (instead of loading them at once), reducing memory usage.

LSTMs, as used in the proposed model, typically require data to be fed in sequences or windows of the original time series. To achieve this, a sliding window approach was used (in conjunction with the data generators), where a fixed-size window slides over the data set, capturing subsets of data points at each position (Figure 11). This method sequentially feeds these subsets, or windows, into the model. The window length is determined by the predefined lengths of the input and target sequences. For instance, with data recorded at 15-minute intervals, a forecasting horizon of 3 hours would result in a target sequence length (TSL) of 12. The input sequence length (ISL), must be chosen empirically based on the data and its characteristics (see ACF and PCF plots in Section 3.2). In this thesis, various ISLs spanning from 1 (=15 min.) to 96 (=24 hrs.) were tested and their performances compared to determine the optimal length. The order of the windows themselves (note: not their contents) were shuffled within the training and

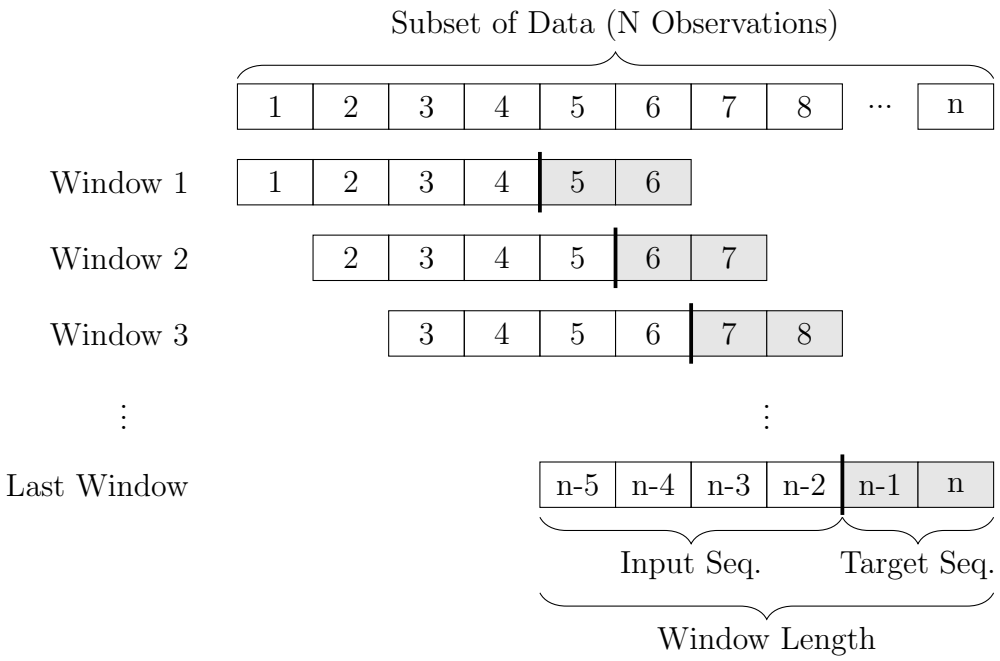

Figure 11: Illustration of Sliding Window Data Sequencing

validation generator before being sent to the model. This approach keeps the temporal dependencies within of the datapoints within a window, while increasing the robustness of the model and preventing overfitting. For the test generator, the windows were not shuffled but fed in a strictly sequential manner to ensure conditions close to a real world environment.

The sliding window approach with overlapping sequences is advantageous because it increases the training data while preserving time-dependent features. However, this method poses a unique challenge: overlapping predictions for the same time step when the target sequence length is longer than one. For instance, as illustrated in Figure 11, target step six appears in the first two windows, leading the model to produce two different predictions for the same target step. To address this, various sliding window aggregation (SWAG) algorithms have been developed for this purpose (Tangwongsan et al., 2022). These algorithms, such as simple averaging, weighted averaging, and advanced ensemble methods, merge overlapping predictions. Complex SWAG techniques have the potential to enhance prediction accuracy by considering factors like confidence level, historical performance, and context of the predictions. Yet, for the sake of interpretability, the proposed model uses a SWAG algorithm that essentially produces a simple average of the overlaps without weighting the different predictions.

4 Proposed Model

In the following sections, the benchmark models are defined, the technical backgrounds of LSTMs and CNNs are illustrated, the proposed CNN-2-LSTM architecture is described, and the model training process is detailed.

4.1 Benchmark Persistence Model

Persistence models serve as a fundamental benchmark for performance evaluation in various predictive fields. Despite their simplicity, these models are extensively utilized in meteorology and solar forecasting (e.g., Iheanetu (2022), Pandžić and Capuder (2023)). The core principle of persistence models is straightforward: they assume that the current or past values will remain unchanged and thus will be the same in the future. This approach, while basic, provides a valuable point of reference for comparing the accuracy of more complex forecasting methods. Persistence models often utilize different look back periods, such as the most recent value, the values from the same time yesterday, or the values from the same time one year ago. A look back period refers to the specific past time interval from which data is used to make predictions for the future. The general concept of the benchmark persistence models used can be formalized as:

$$\hat{GHI}_{t+s} = GHI_{t-l+s} \tag{2}$$

where:

GHI_t = Observed GHI at time step t
$\hat{GHI}_{t+s}$ = Predicted GHI for time step $t+s$
t = Time step
l = Look back period
s = Forecasting horizon

In this thesis, the persistence forecasts are based on the actual GHI observations from the previous day, which corresponds to 24 hours or 96 15-minute intervals earlier. Consequently, the thesis establishes benchmark models for three distinct forecasting horizons, resulting in three separate persistence models. Their performance will serve as a baseline during model evaluation and they are defined as follows:

$$\text{15-Min. Horizon:} \quad \hat{GHI}_{t+s} = GHI_{t-96+s} \quad for \quad s = 1 \tag{3}$$

$$\text{3-Hr. Horizon:} \quad \hat{GHI}_{t+s} = GHI_{t-96+s} \quad for \quad s = 1, 2, \ldots, 12 \tag{4}$$

$$\text{6-Hr. Horizon:} \quad \hat{GHI}_{t+s} = GHI_{t-96+s} \quad for \quad s = 1, 2, \ldots, 24 \tag{5}$$

4.2 Technical Foundations of the Proposed Model

Understanding the technical foundations of LSTMs and CNNs is essential for comprehending their integration and performance implications in the CNN-2-LSTM model. Therefore, the following subsections provide a concise introduction to the technical aspects of both techniques.

4.2.1 Technical Foundations of LSTM Networks

In traditional feed-forward neural network designs, connections only go forward from input to output. On the contrary, recurrent neural networks (RNN) allow connections that feed in the opposite direction, either from one node to another or from one node to itself (Figure 12). This is especially useful for sequences of data (e.g., time series data), since information about the previous time step can be fed back into the network to be used in the current time step. This essentially enables the RNN to retain persistent information over multiple time steps.

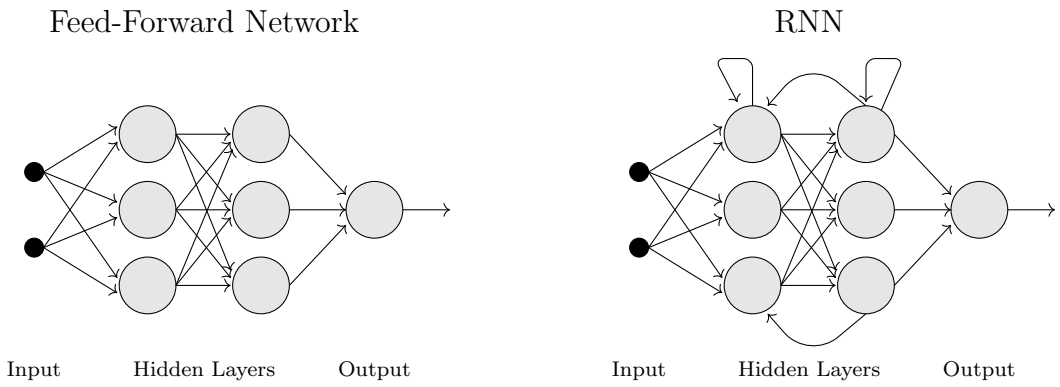

Figure 12: Simplified Conceptual Comparison of Feed-Forward Networks to RNNs

Yet, RNNs struggle to maintain information over long sequences due to issues connected to vanishing and exploding gradients. Long Short-Term Memory (LSTM) networks are a specialized type of RNN that is designed to specifically address these issues and to learn long-term dependencies in sequence prediction problems. They were first introduced by Hochreiter and Schmidhuber in 1997.

The architecture of an LSTM cell, illustrated in Figure 13, includes several key components designed to regulate the flow of information effectively. Central to the LSTM's operation is the *cell state* (C), which acts as a kind of memory that carries information across different time steps. The information within the cell state is regulated by structures known as *gates*, which decide what information to forget, add, and output at each time step. Gates are neural network layers themselves, which means that their exact parameters are learnt during model training.

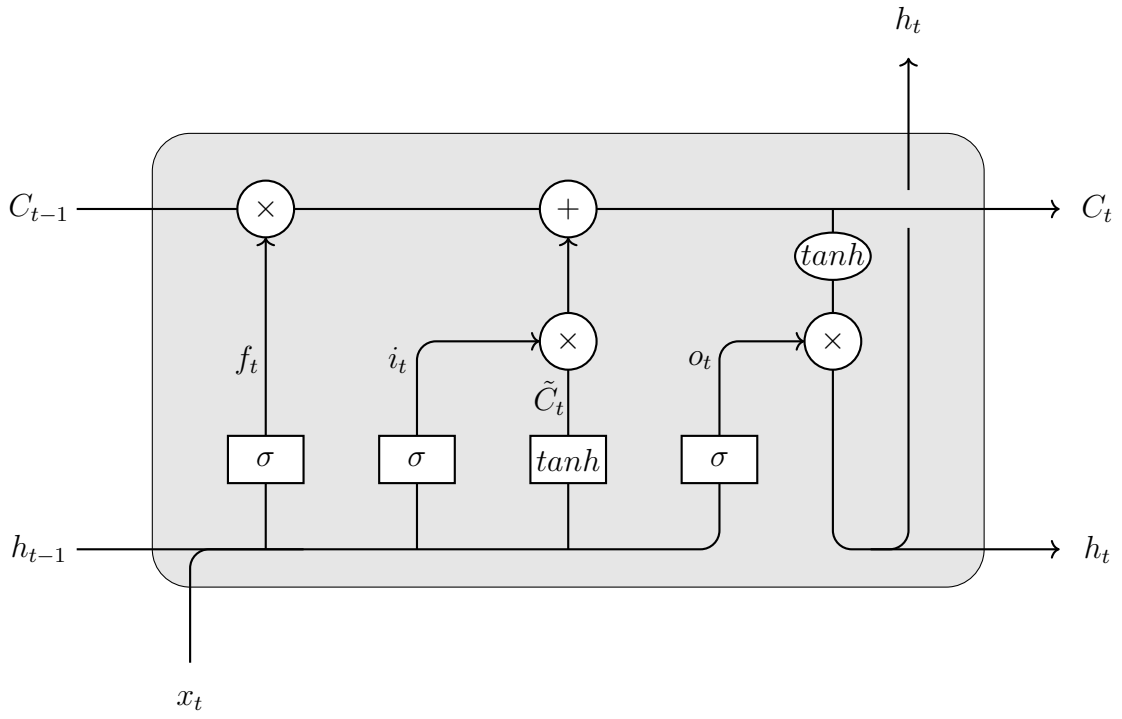

Figure 13: LSTM Cell Schematic

The *forget gate* (f_t) decides what portion of the old cell state (C_{t-1}) should be carried forward. It achieves this by using the output from the previous time step (h_{t-1}), the input from the current time step (x_t) and a sigmoid activation function (σ) which yields a result between zero and one. A zero would erase the entire cell state, whereas a one would keep all of the information of the cellstate. The *input gate* (i_t) controls the extent to which new information ($\tilde{C}_t$) should be written to the cell state. It also uses a sigmoid activation function to scale the importance of the new information. The *candidate cell state* ($\tilde{C}_t$) is a vector of new candidate values, created by applying the tanh function to the input data and previous hidden state, which could potentially be added to the cell state.

The cell state (C_t) is updated by combining the forget gate's filtered old cell state and the scaled candidate cell state. Mathematically, this is represented as:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{6}$$

The *output gate* (o_t) determines the output of the LSTM cell. It uses the sigmoid activation function to decide which parts of the cell state should be output. The final output (h_t) is then obtained by applying the tanh function to the updated cell state and multiplying by the output gate's result:

$$h_t = o_t \cdot \tanh(C_t) \tag{7}$$

The gates and their interactions enable the LSTM to effectively manage the flow of

information and retain long-term dependencies, which are essential for tasks requiring the modeling of long term sequences. This makes them an ideal choice for the solar forecasting task at hand.

4.2.2 Technical Foundations of CNNs

CNNs have emerged as a pivotal technology in the field of deep learning, particularly for tasks involving image and spatial data. Introduced by Yann LeCun and his collaborators in the late 1980s and early 1990s, CNNs have revolutionized pattern recognition and computer vision through their unique architecture that leverages the spatial hierarchies in data (Lecun et al., 1998).

The fundamental innovation of CNNs lies in their convolutional layers, which consist of a set of learnable filters applied to input data to produce feature maps. These filters enable CNNs to capture local spatial relationships effectively, making them highly suited for image processing tasks (Krizhevsky et al., 2017). Unlike traditional fully connected neural networks, where each neuron is connected to every neuron in the previous layer, CNNs utilize a sparse connectivity pattern, significantly reducing the number of parameters and computational complexity. This approach not only enhances computational efficiency but also improves the model's ability to generalize from the training data.

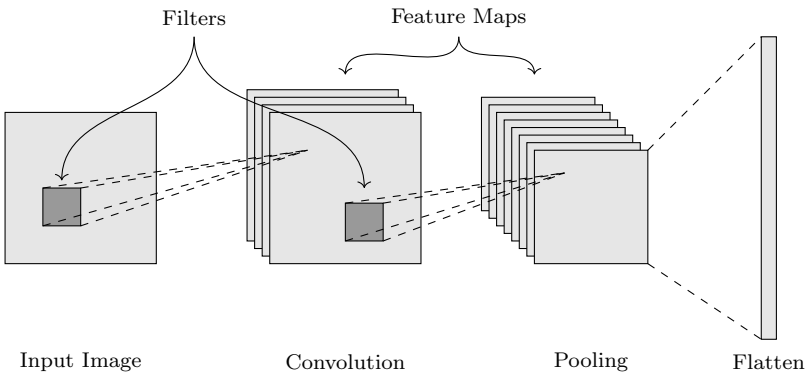

Figure 14: Simplified CNN Schematic

CNNs typically consist of two key components: convolutional layers and pooling layers. The convolutional layers apply filters to the input image, detecting features such as edges, textures, and shapes. These features are then downsampled by pooling layers, which reduce the spatial dimensions of the feature maps, thereby decreasing computational load and helping to achieve spatial invariance (Zeiler and Fergus, 2013). At the end, the feature maps are usually flattened to turn them into a one-dimensional vector. This vector can then be fed to other types of neural layers (e.g., dense layers, LSTMs) for further use.

One of the key strengths of CNNs is their ability to perform end-to-end learning, where the network automatically learns the optimal features directly from raw data, guided by the task-specific loss function. This capability eliminates the need for manual feature

extraction, a significant advantage over traditional machine learning methods (Rawat and Wang, 2017).

In conclusion, the ability of CNNs to learn hierarchical feature representations from raw visual data makes them the ideal choice for processing the satellite images in the context of solar forecasting.

4.3 Proposed CNN-2-LSTM Architecture

As an overview, the proposed model architecture comprises two main branches: an LSTM component for numerical data and a CNN-LSTM component for image data (Figure 15). The LSTM branch captures temporal dependencies in numerical sequences, while the CNN-LSTM branch extracts and processes spatial features from image sequences. The outputs from both branches are concatenated and passed through fully connected layers to generate the final forecast, effectively integrating spatial and temporal information for robust solar irradiance predictions.

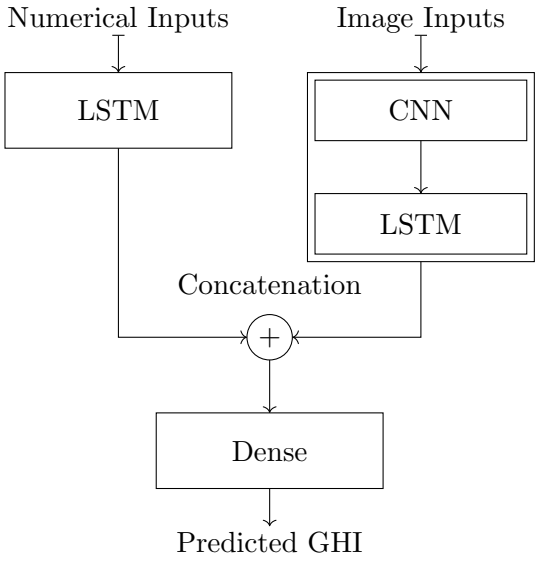

Figure 15: High-Level Model Architecture Overview

4.3.1 LSTM Component of the Model for Numerical Inputs

The LSTM branch of the model is designed to handle numerical input data and is structured to capture and process temporal dependencies in the input sequences. The input sequences consist of GHI measurements at time t along with variables representing the month of the year, day of the month, hour of the day, and minute of the hour at that point in time. The length of the sequences is dependent on the specific model configuration (see Section 4.4).

The input data first passes through an input layer, which accommodates sequences with these five features at each time step of the input sequences. This data then enters the first LSTM layer, which contains 32 cells and returns sequences, thereby transforming each time step into a 32-dimensional vector while maintaining the temporal structure of the sequence.

The output from the first LSTM layer is subsequently fed into a second LSTM layer, also consisting of 32 cells and configured to return sequences, further refining the temporal information. A time distributed dropout layer with a dropout rate of 0.2 is then applied to the output, which helps to prevent overfitting by randomly setting a fraction of the input units to zero during training. The time distributed wrapper ensures that the dropout is applied to each time step independently within the sequence, preserving the temporal structure of the data and maintaining consistency across the sequence. This encapsulation is necessary to ensure that the dropout does not disrupt the temporal dependencies captured by the LSTM layers, allowing the model to effectively learn and generalize from the sequential data.

The dropout-regularized sequences are then processed by a third LSTM layer containing 64 cells, which also returns sequences. This layer captures more complex temporal patterns by transforming each time step into a 64-dimensional vector. The final LSTM layer, with 64 cells, processes the sequence output from the third LSTM layer but is configured to return only the last output in the sequence, resulting in a single 64-dimensional vector that encapsulates the temporal information from the entire input sequence.

Finally, this output vector is passed through a batch normalization layer, which normalizes the data by adjusting and scaling the activations. This step helps to stabilize and accelerate the training process.

Layer Type	**Output Shape**	**Description**
Input Layer	(ISL, 5)	Sequenced, five features
LSTM Layer 1	(ISL, 32)	32 cells, returns sequences
LSTM Layer 2	(ISL, 32)	32 cells, returns sequences
Distributed Dropout	(ISL, 32)	Rate = 0.2, applied per time step
LSTM Layer 3	(ISL, 64)	64 cells, returns sequences
LSTM Layer 4	(64)	64 cells, returns vector
Batch Normalization	(64)	Normalizes the output vector

Note: ISL = Input Sequence Length

Table 3: Summary of Layers and Settings for the Numerical LSTM Branch

4.3.2 CNN-LSTM Component for Image Inputs

The CNN-LSTM branch of the model is designed to process satellite image input data and is structured to capture spatial features from the images as well as temporal dependencies

across the sequence of images. The input consists of sequences of 64 by 64 pixel grayscale satellite images, where each image corresponds to a specific time step.

The input data first passes through an input layer that accommodates sequences of these 64 by 64 pixel images. Each image in the sequence is then processed independently by convolutional layers. Similarly to the numerical LSTM branch, the layers are wrapped in a time distributed layer, ensuring that the same convolutional operations are applied at each time step.

Layer Type	**Output Shape**	**Description**
Input Layer	(ISL, 64, 64, 1)	Sequenced, 64x64 pixel images
Distributed Conv2D	(ISL, 32, 62, 62)	32 filters, (3, 3), ReLU
Distributed MaxPooling2D	(ISL, 32, 31, 31)	Pool size (2, 2)
Distributed Conv2D	(ISL, 16, 29, 29)	16 filters, (3, 3), ReLU
Distributed MaxPooling2D	(ISL, 16, 14, 14)	Pool size (2, 2)
Distributed Flatten	(ISL, 3136)	Flattened feature maps
LSTM Layer 1	(ISL, 32)	32 cells, returns sequences
LSTM Layer 2	(ISL, 32)	32 cells, returns sequences
Distributed Dropout	(ISL, 32)	Rate = 0.2, applied per time step
LSTM Layer 3	(ISL, 64)	64 cells, returns sequences
LSTM Layer 4	(64)	64 cells, returns vector
Batch Normalization	(64)	Normalizes the output vector

Note: ISL = Input Sequence Length

Table 4: Summary of Layers and Settings for the CNN-LSTM Branch

The first convolutional layer uses 32 filters of size (3, 3) and applies a rectified linear unit (ReLU) activation function. ReLU is a popular activation function in deep learning that introduces non-linearity into the model by outputting the input directly if it is positive, otherwise, it outputs zero. This non-linearity allows the model to learn complex patterns in the data while also helping to mitigate the vanishing gradient problem, thus enabling faster and more effective training. The spatial features captured by this layer are then downsampled using a time distributed max pooling layer. This is followed by a second convolutional layer with 16 filters of the same size, again followed by a time distributed max pooling layer to further extract and downsample spatial features.

After the convolutional layers, the output is flattened using a time distributed flattening layer, preparing the data for sequence modeling. The flattened features are then fed into two LSTM layers, each containing 32 cells and configured to return sequences. These layers capture the temporal dynamics of the image sequence. A time distributed dropout layer with a dropout rate of 0.2 is applied to the output of the second LSTM layer to prevent overfitting.

The final LSTM layer, with 64 cells, processes the sequence output but returns only the last output in the sequence, resulting in a single 64-dimensional vector that encap-

sulates the temporal information from the entire image sequence. This output vector is then normalized using a batch normalization layer to stabilize and accelerate the training process.

4.3.3 Combination of Subcomponents and Output

The final part of the model involves concatenating the outputs from the LSTM branch (for numerical data) and the CNN-LSTM branch (for image data), followed by processing through fully connected dense layers to produce the final forecast.

The concatenation step integrates the learned features from both numerical and image data sources. The normalized output vectors from the LSTM branch and the CNN-LSTM branch are concatenated into a single vector. This combined vector encapsulates both the temporal dynamics captured by the LSTM layers and the spatial features processed by the CNN layers, providing a comprehensive representation of the input data.

The concatenated vector is then passed through a series of dense layers to refine and map the integrated features to the desired output space. The first dense layer has 16 neurons and applies a linear activation function to transform the combined features. This is followed by a second dense layer with 8 neurons, also using a linear activation function, further refining the feature representation.

Finally, the processed data is passed to the output dense layer, which uses a ReLU activation function to produce the final forecast. In this layer the ReLU activation ensure an output that is positive (negative values would be nonsensical in the context of solar forecasting). The number of neurons in this layer corresponds to the target sequence length, which represents the forecasting horizon.

Layer Type	**Output Shape**	**Notes**
Concatenation	(128)	Combines LSTM and CNN-LSTM outputs
Dense Layer 1	(16)	16 neurons, linear activation
Dense Layer 2	(8)	8 neurons, linear activation
Output Layer	(TSL)	Neurons equal forecast horizon, ReLU activation

Note: TSL = Target Sequence Length / Forecasting Horizon

Table 5: Summary of Layers and Settings for the Combination, Dense Layers, and Output

To benchmark the performance and determine the impact of the satellite image branch, a pure LSTM model was also trained. This model only utilizes the Numerical LSTM and Output parts of the full model. By comparing the performance of this pure LSTM model with the full CNN-LSTM model, it is possible to assess whether the satellite image branch contributed positively to the overall performance or if most of the performance was achieved by the numerical LSTM component alone.

This setup effectively leverages the strengths of both the LSTM and CNN-LSTM branches, providing a robust framework for producing accurate solar irradiance forecasts.

4.4 Model Training

A total of 36 models was trained, considering the inclusion or exclusion of the image branch, three different forecasting horizons (15-minutes, 3 hours, and 6 hours), and six different input sequence lengths (1, 4, 8, 16, 48, and 96).

The models were trained using TensorFlow and Keras on Google Colab, utilizing an L4 Tensor Core GPU. To accommodate for the size of the data, data generators were implemented (see Section 3.3), which efficiently managed memory by loading data in batches only as needed during the training process. For all models, a batch size of 32 was chosen to balance the training speed against the available memory resources effectively.

As a loss function for model training and performance evaluation, RMSE was chosen. Due to its quadratic nature, it penalizes large deviations exponentially more, which is desirable for solar forecasting. As a complementary loss function for performance evaluation, the mean absolute error (MAE) was used as well. They are defined as:

$$RMSE = \sqrt{\frac{\sum(GHI_i - \hat{GHI}_i)^2}{n}} \tag{8}$$

and

$$MAE = \frac{\sum |GHI_i - \hat{GHI}_i|}{n} \tag{9}$$

where:

GHI_t = Observed solar irradiance at time step t
$\hat{GHI}_t$ = Predicted solar irradiance at time step t
n = Number of observations

All models were initialized with random weights and trained for up to 50 epochs. An early stopping callback monitored validation loss, halting training if RMSE didn't improve for 10 consecutive epochs, and restoring the best weights. Most training runs ended between 20 and 35 epochs (Figure 16).

The *Adam* optimizer (Kingma and Ba, 2014) was used for efficient weight updates with an adaptive learning rate mechanism. Additionally, the learning rate was reduced by a factor of 0.2 if the validation loss plateaued for 5 epochs, preventing stagnation and ensuring efficient convergence. The minimum learning rate was set to 0.000001.

Figure 16 shows the training and validation losses for the best CNN-2-LSTM configurations for each forecasting horizon. Training losses were higher than validation losses, explained by the easier-to-forecast validation subset, which contains GHI data from one season, compared to the more variable training data. Additionally, dropout regularization increased training loss but not validation loss, promoting robust feature learning and preventing overfitting.

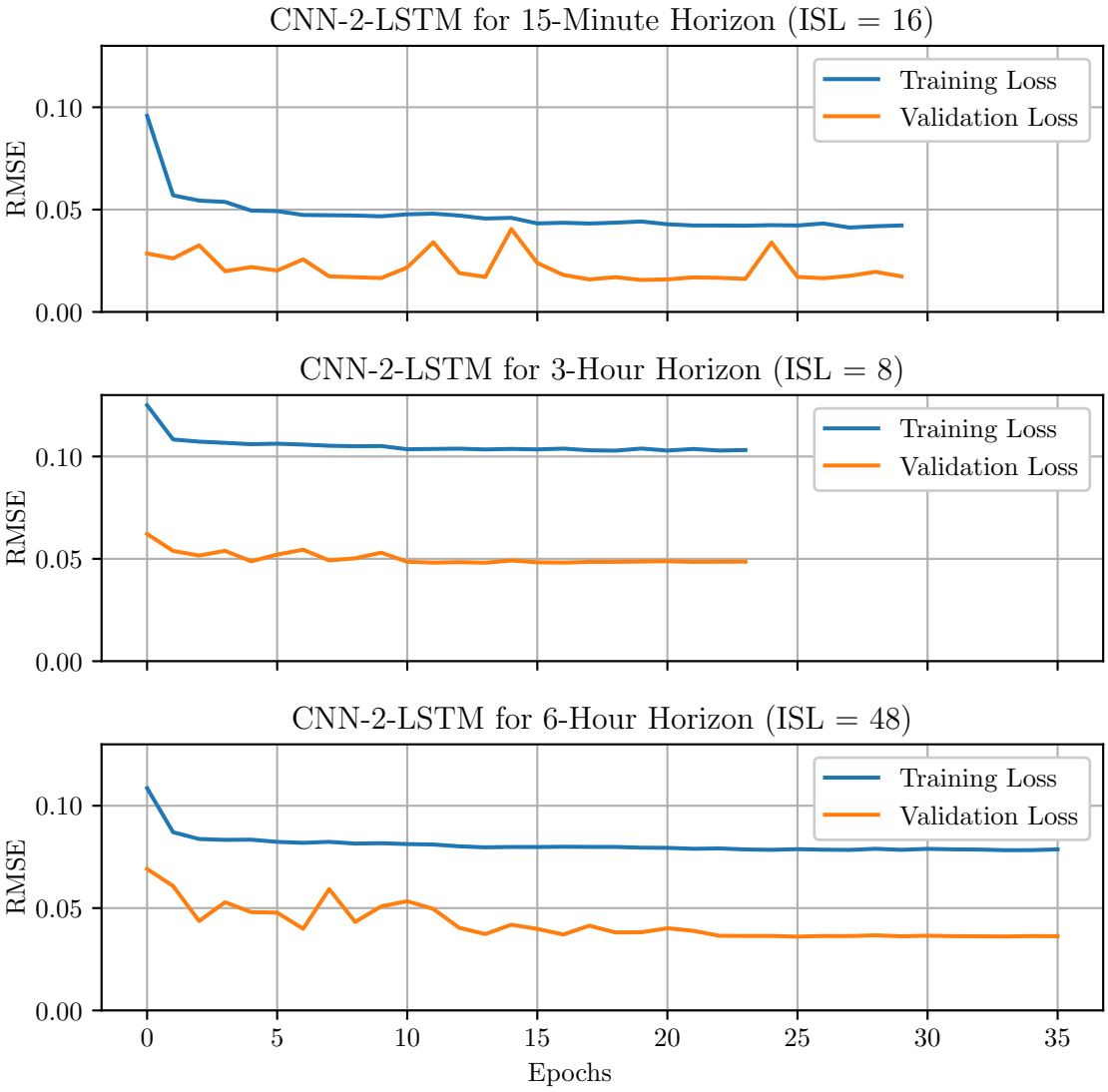

Figure 16: Training and Validation Losses for CNN-2-LSTM at Different Horizons

5 Model Performance and Evaluation

This section details the outcomes of models trained with varying configurations to assess different forecasting horizons and input sequence lengths. Performance is evaluated using RMSE and MAE, and compared to the persistence forecast model. Additionally, a custom performance metric reflecting the impact of forecasting error on business revenues is introduced for secondary evaluation.

5.1 Economic Performance Metric Based on Balance Energy

For German energy providers, one of the key benefits of employing an advanced solar forecasting model is to significantly reduce the penalty payments incurred for balance energy when their forecasts do not align with actual production. Balance energy, also known as balancing power, is essential in maintaining the stability of the electrical grid. It refers to the extra energy that transmission grid operators must either supply or absorb to offset the discrepancies between predicted and actual energy production (TransnetBW et al., 2022).

In Germany, the cost associated with balance energy is determined by the uniform imbalance price (reBAP)[5], which is the market price for balancing energy expressed in Euros per megawatt-hour (MWh). When an energy provider's production forecast is inaccurate, resulting in either an excess or shortfall of energy compared to their forecast, they must buy or sell the difference at the reBAP rate. The reBAP can be either positive (indicating a shortage of energy in the electricity grid) or negative (indicating a surplus of energy in the grid). Consequently, energy providers may either incur costs or generate revenue depending on whether they have overestimated or underestimated their output, as illustrated in Table 6 (TransnetBW et al., 2022). Please note that even when revenue is generated, it might be lower than the revenue that could have been obtained by selling the energy at the regular market price for electricity, rather than at the reBAP rate. This can lead to substantial (opportunity) costs, especially if the imbalance is large and the reBAP rate is high, further accentuating the importance of precise forecasts for solar energy producers.

	Overproduction	**Underproduction**
reBAP positive	Revenue	Cost
reBAP negative	Cost	Revenue

Table 6: Impact of reBAP on Energy Providers' Financial Outcomes

While the loss functions used during training yield a good basis for model comparison from a technical standpoint, they are less helpful when used in a business context. De-

[5]The acronym 'reBAP' stems from the official German name: ***Re****gelenergie-****B****örsen****a****brechnungs****p****reis*

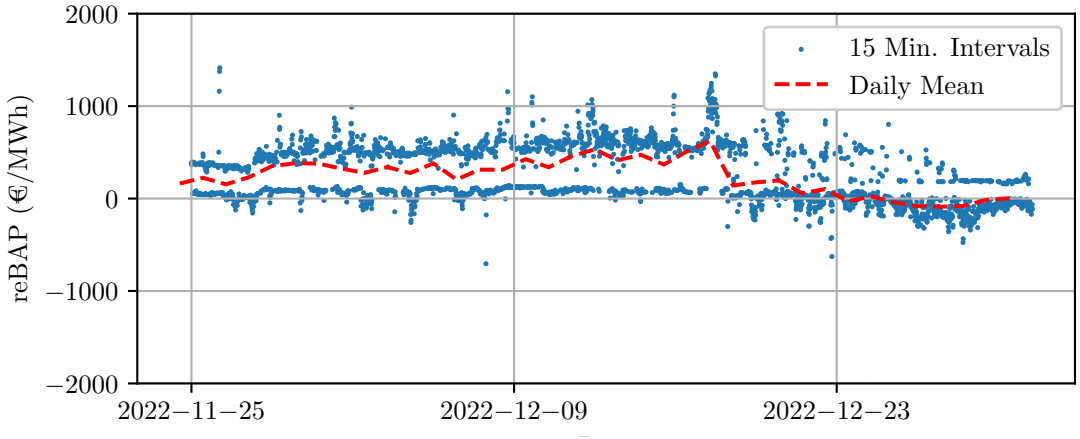

Figure 17: reBAP Rate Over the Period of the Test Set with Daily Mean

pending on the current reBAP, the economic effects of inaccurate forecasts have a varying severity which cannot be reflected using standard loss functions. Therefore, this thesis proposes a custom performance metric that estimates the mean reBAP payments incurred by prediction errors, termed *Mean Balance Energy Cost* (MBEC). The underlying historic reBAP rate was sourced directly from the four transmission system operators in Germany (TransnetBW et al., 2024). Only the data that covers the period of the test set was used, yielding 3,504 price data points (Figure 17). To obtain the MBEC for a given model, a three-step process is performed:

1. Conversion of the actual predicted GHI values to estimated electrical power outputs measured MWh.

2. Multiplication of the difference between the actual and predicted MWh values with the reBAP at that point in time.

3. Calculation of the mean reBAP payment over the period of the test set.

To convert the true and predicted GHI values to estimated electric power outputs of a solar power plant (P_{out}), the following formula (derived from Clack (2017)) can be used as an estimator:

$$P_{out} = \frac{GHI \times A \times \eta}{1000000} \times \frac{15}{60} \times \frac{1}{6} \tag{10}$$

where:

GHI = Observed or predicted GHI measured in Watts per square metre,
A = Total area of the solar panels measured in square metres,
η = Factor to denote the efficiency of the solar panels.

The result is divided by 1000000 to convert it from Watts to MWh and multiplied by $\frac{15}{60}$ to adjust the 15-minute interval data to hourly data. Additionally, since the GHI data

is aggregated from six observation stations, it is multiplied by $\frac{1}{6}$ to obtain the average value across these stations. For the purpose of this thesis, a mid-sized solar power plant with one square kilometre of solar panel area ($A = 1000$) and a modern standard with comparatively low efficiency losses ($\eta = 0.18$) is assumed. This yields:

$$P_{out} = \frac{GHI \times 1000 \times 0.18}{1000000} \times \frac{15}{60} \times \frac{1}{6} = GHI \times 0.0000075 \tag{11}$$

Therefore all GHI values are multiplied by the factor 0.0000075 to obtain an estimated MWh equivalent. This can then be used to calculate the MBEC:

$$MBEC = \frac{\sum_t^T ((GHI_t - \hat{GHI}_t) \times 0.0000075 \times reBAP_t)}{n_{test}} \tag{12}$$

where:

GHI_t = Observed GHI at time t in Watts per square metre,
$\hat{GHI}_t$ = Predicted GHI at time t in Watts per square metre,
$reBAP_t$ = reBAP rate at time t in Euros per MWh,
n_{test} = Number of observations in the test set.

The result is a metric that reflects the mean payment in Euros for balance energy at every 15-minute interval in the test set. This metric can be positive or negative, with lower values indicating better economic performance. It effectively highlights the real-world business implications of solar forecasting accuracy in a simulated environment. As the mean reBAP rate is overall positive for the test set duration (Figure 17), it is to be expected that models that tend to underpredict achieve a lower MBEC. In this thesis, it is utilized solely for model evaluation purposes and not during model training.

5.2 Model Performance at a 15-Minute Forecasting Horizon

Table 7 provides a detailed overview of model performance at a 15-minute forecasting horizon. Overall, the CNN-2-LSTM model with an ISL of 16 achieves the lowest RMSE of 5.877×10^{-3}, making it the top performer in terms of technical accuracy. This model also shows strong performance in MAE, with a value of 4.072×10^{-3}, and a modest MBEC of 0.03930. This indicates that the model is not only comparatively accurate but also relatively cost-effective, as it incurs moderate costs for balance energy.

The CNN-2-LSTM model with an ISL of 1, while having a slightly higher RMSE of 6.357×10^{-3}, achieves the lowest MAE of 3.657×10^{-3}. Its MBEC is -0.00624, ranking second in terms of financial performance. This suggests that although the ISL of 1 compromises slightly on RMSE, it provides better overall economic performance.

Horizon	**#**	**Model**	**ISL**	**RMSE** ($\times 10^{-3}$)	**MAE** ($\times 10^{-3}$)	**MBEC**
15 min.	1	CNN-2-LSTM	16	**5.877**	4.072	0.03930
	2	CNN-2-LSTM	1	6.357	**3.657**	-0.00624
	3	LSTM	4	6.903	4.327	**-0.01044**
	4	LSTM	8	6.960	4.543	0.04368
	5	CNN-2-LSTM	8	7.334	4.819	0.05148
	6	CNN-2-LSTM	4	7.632	4.801	0.03504
	7	CNN-2-LSTM	96	8.141	4.977	0.05232
	8	CNN-2-LSTM	48	8.239	6.329	0.07458
	9	LSTM	16	8.607	6.317	0.09258
	10	LSTM	96	8.959	5.392	0.06870
	11	LSTM	1	9.055	5.573	0.02940
	12	LSTM	48	10.547	8.228	0.11544
	13	Persistence	n.a.	320.455	127.663	0.40560

Table 7: Model Performance at a 15-Minute Forecasting Horizon Ordered by RMSE

Notably, all models outperformed the persistence benchmark model, with the best-performing CNN-2-LSTM model achieving a reduction in RMSE by 98.17% and a reduction in MAE by 96.81%. It also reduces the MBEC by 90.31%, which could potentially yield significant cost savings for solar energy operators. For example, calculated over an entire year, the CNN-2-LSTM model could save a solar operator €12,835.15 in balance energy costs compared to the persistence model. This substantial difference underscores the business value of the CNN-2-LSTM approach.

Generally, at this forecasting horizon, the CNN-2-LSTM model configurations perform better than the numerical LSTM models. This indicates that the inclusion of satellite cloud masks improves forecasting skill for short-term solar forecasting. Furthermore, short to medium ISLs seem to perform better for this forecasting horizon for both the CNN-2-LSTM and LSTM models alike. This shows that very short-term forecasts are dependent on recent observations and longer ISLs do not provide any additional value.

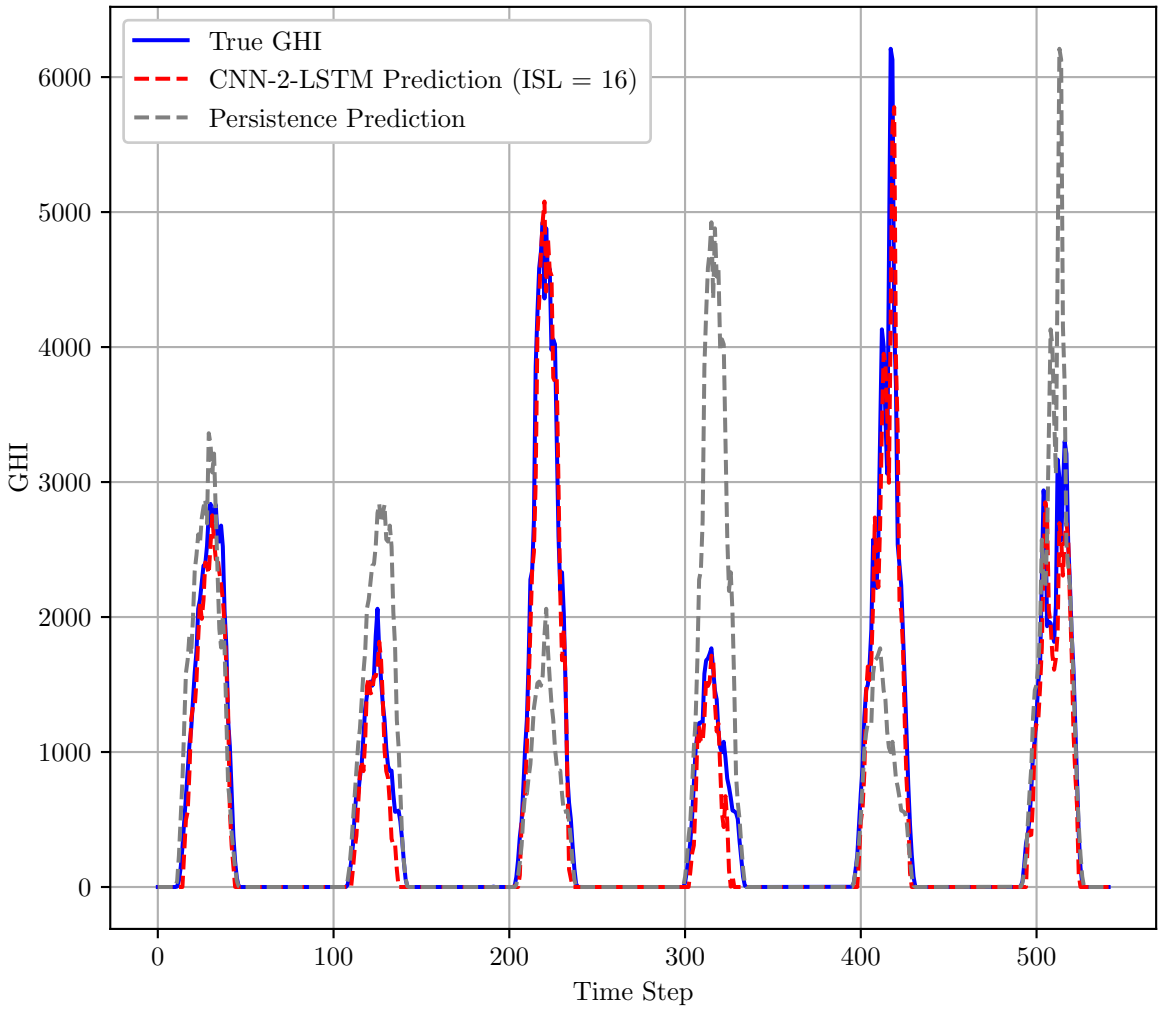

Figure 18: Predictions and True GHI for 15-Minute Forecasting Horizon

Upon visual inspection of the forecasts (Figure 18), it is evident that the CNN-2-LSTM model is capable of accurately predicting the true GHI values at this forecasting horizon. It manages to precisely forecast the daily peaks in GHI and also captures more minor dips in the GHI curves. This supports the notion that the satellite imagery enables the model to predict solar variability caused by cloud cover. There are only minor inaccuracies for the end-of-day predictions where the CNN-2-LSTM model tends to underpredict the true values. It is also obvious that the persistence model struggles to give useful predictions, especially when consecutive days have a significant difference in peak GHI values (e.g., time step 200 to 350 in Figure 18).

Overall, the CNN-2-LSTM shows highly satisfactory results at the 15-minute forecasting horizon both from a technical and economical perspective on performance. These findings highlight the model's robustness and practical applicability and potential value in a real-world business environment.

5.3 Performance at a 3-Hour Forecasting Horizon

Table 8 provides an in-depth analysis of model performance at a 3-hour forecasting horizon. As expected, performance metrics are worse compared to the 15-minute forecasting horizon models, since predicting further into the future is a significantly more challenging task. Among the models evaluated, the CNN-2-LSTM with an ISL of 8 emerges as the most accurate, achieving the lowest RMSE of 24.986×10^{-3}. It also yields a competitive MAE of 17.098×10^{-3} and an MBEC of 0.06948, reflecting its balanced performance in terms of both accuracy and cost-effectiveness.

The LSTM model with an ISL of 8 follows closely, with an RMSE of 25.040×10^{-3} and an MAE of 17.106×10^{-3}. Its MBEC is slightly lower at 0.06654, suggesting that while it is nearly as accurate as the top performer, it manages balance energy costs slightly more effectively.

Interestingly, the CNN-2-LSTM model with an ISL of 96, despite having a higher RMSE of 25.796×10^{-3}, achieves the lowest MAE of 15.897×10^{-3} and an MBEC of -0.02550. This configuration appears to strike a favorable balance in scenarios where minimizing absolute error is more critical than minimizing squared error.

Horizon	#	Model	ISL	RMSE ($\times 10^{-3}$)	MAE ($\times 10^{-3}$)	MBEC
3 hours	1	CNN-2-LSTM	8	**24.986**	**17.098**	0.06948
	2	LSTM	8	25.040	17.106	0.06654
	3	CNN-2-LSTM	96	25.796	15.897	-0.02550
	4	CNN-2-LSTM	4	26.338	17.356	0.04284
	5	LSTM	96	27.166	17.151	0.00138
	6	CNN-2-LSTM	1	27.731	17.318	**-0.10278**
	7	CNN-2-LSTM	16	27.787	17.125	0.16104
	8	CNN-2-LSTM	48	27.948	17.121	0.15708
	9	LSTM	1	28.428	17.651	-0.06510
	10	LSTM	48	29.800	18.748	0.18534
	11	LSTM	4	30.483	20.457	-0.00042
	12	LSTM	16	33.937	22.278	0.23640
	13	Persistence	n.a.	320.455	127.663	0.40560

Table 8: Model Performance at a 3-Hour Forecasting Horizon Ordered by RMSE

Notably, all models considerably outperform the persistence benchmark model. The leading CNN-2-LSTM model reduces RMSE by 92.20% and MAE by 86.60% compared to the persistence model. Additionally, it lowers MBEC by 83.20%, offering significant potential savings in balance energy costs for solar energy operators. For example, over a year, this model has a cost saving potential of € 11,777.64 in balance energy costs for a solar energy providers.

Overall, for the 3-hour forecasting horizon, CNN-2-LSTM configurations generally outperform their LSTM counterparts, albeit only slightly. This suggests that the combi-

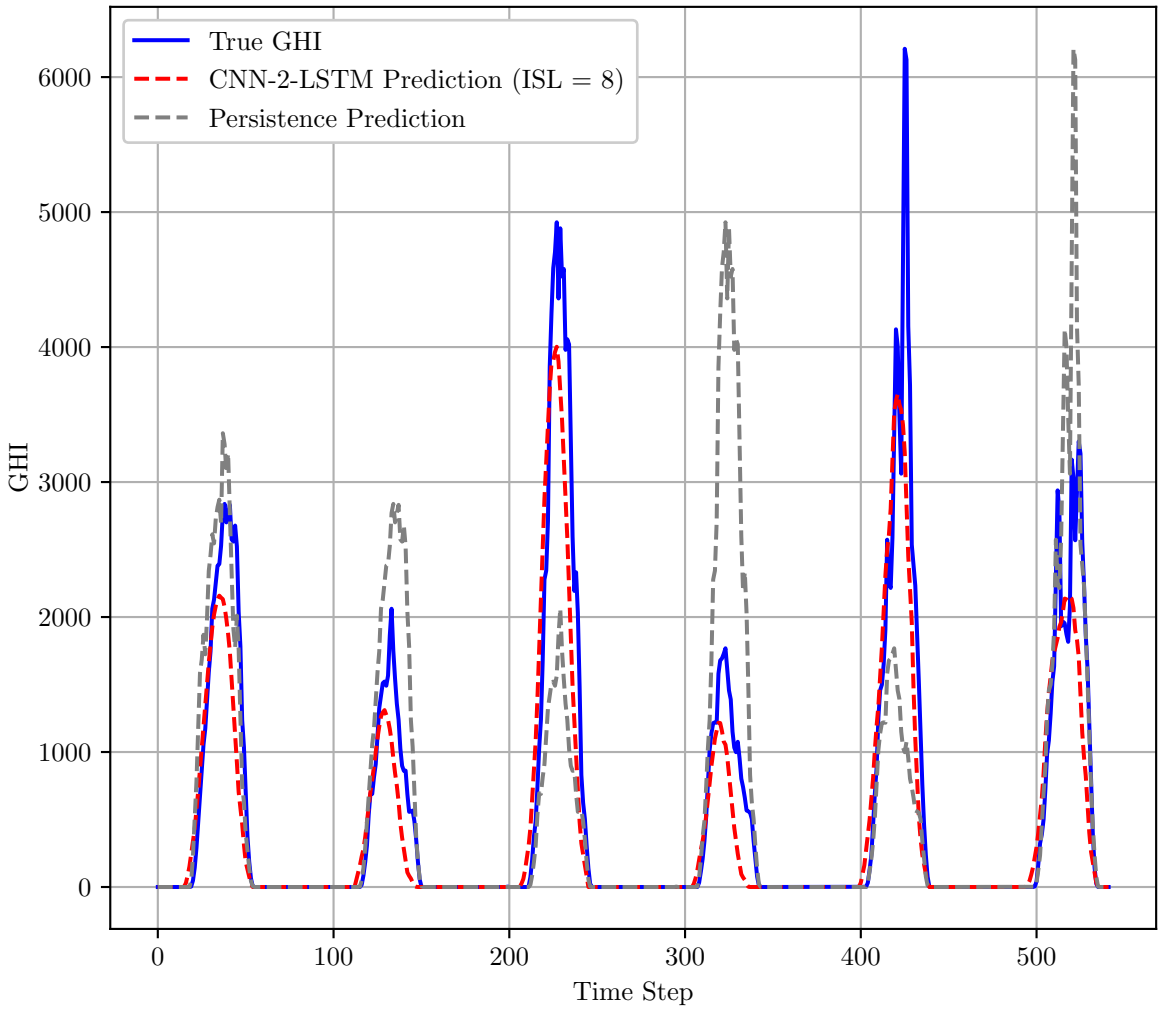

Figure 19: Predictions and True GHI for 3-Hour Forecasting Horizon

nation of convolutional and LSTM layers enhances predictive performance, yet less than at the 15-minute horizon. Additionally, models with medium ISLs tend to perform better, indicating that forecasts benefit longer data sequences compared to the 15-minute forecasting models.

The visual inspection of predicted and true GHI values (Figure 19) reveals that the CNN-2-LSTM model tends to underpredict peak GHI values. Despite this, the model successfully captures the overall dynamics of the true values, albeit with less accuracy concerning smaller variations in GHI measurements. This 'smoothness' in the predicted values can be partially attributed to the averaging of overlapping predictions, as detailed in Section 3.3. Additionally, the model often mispredicts sunset and sunrise times, with the persistence forecast demonstrating superior performance in this aspect. These findings suggest that while the CNN-2-LSTM model is effective in capturing broader patterns, further refinement is needed to enhance its accuracy for peak values and small scale GHI variability.

5.4 Performance at a 6-Hour Forecasting Horizon

As illustrated in Table 9, the CNN-2-LSTM architecture is less effective at a 6-hour forecasting horizon. The numerical LSTM model with an ISL of 8 achieves the lowest RMSE (39.040×10^{-3}) and MAE (25.181×10^{-3}). However, the lowest MBEC (-0.03396) is achieved by the LSTM trained with an ISL of 1. This discrepancy underscores that technical and financial performance do not necessarily coincide for this task. Although all models outperform the benchmark persistence forecast, the LSTMs outperform the CNN-2-LSTMs at this forecasting horizon, indicating that satellite image inputs are less useful for longer horizons. Nevertheless, the best performing CNN-2-LSTM (ISL = 48) model reduces RMSE by 87.10% and MAE by 78.32% compared to the persistence model. Moreover, the CNN-2-LSTM model could potentially yield balancing cost savings of €5,423.79 compared to the benchmark model when calculated over an entire year.

A general observation is that model configurations with shorter ISLs perform significantly worse, both with LSTM and CNN-2-LSTM models. This was to be expected, as with a longer forecasting horizon the model needs longer patterns in the data to produce reliable forecasts.

Horizon	#	Model	ISL	RMSE ($\times 10^{-3}$)	MAE ($\times 10^{-3}$)	MBEC
6 hours	1	LSTM	8	**39.040**	**25.181**	0.21810
	2	LSTM	48	39.268	25.417	0.18834
	3	CNN-2-LSTM	48	41.338	27.678	0.15480
	4	LSTM	96	41.777	27.373	0.04356
	5	LSTM	16	42.124	28.100	0.21120
	6	CNN-2-LSTM	16	42.224	28.641	0.18912
	7	CNN-2-LSTM	8	42.294	28.779	0.15054
	8	CNN-2-LSTM	96	43.037	28.164	-0.02268
	9	LSTM	4	43.181	28.354	0.02472
	10	LSTM	1	43.735	28.343	**-0.03396**
	11	CNN-2-LSTM	1	43.736	28.112	-0.01578
	12	CNN-2-LSTM	4	43.915	29.363	0.05634
	13	Persistence	n.a.	320.455	127.663	0.40560

Table 9: Model Performance at a 6-Hour Forecasting Horizon Ordered by RMSE

Upon visual inspection of the forecasts (Figure 20), it is evident that the CNN-2-LSTM model struggles to accurately predict peak GHI values, even more so than at the 3-hour forecasting horizon. Although the predicted peak values are somewhat related to the true observed values, the model tends to substantially underpredict these peaks. Additionally, it is less effective at predicting sunrise and sunset times, often forecasting non-zero positive values before sunrise and after sunset. This outcome is somewhat expected given the exponentially more challenging forecasting task at this horizon, yet it also highlights the need for improvement or a different approach for longer time horizons. It is anticipated

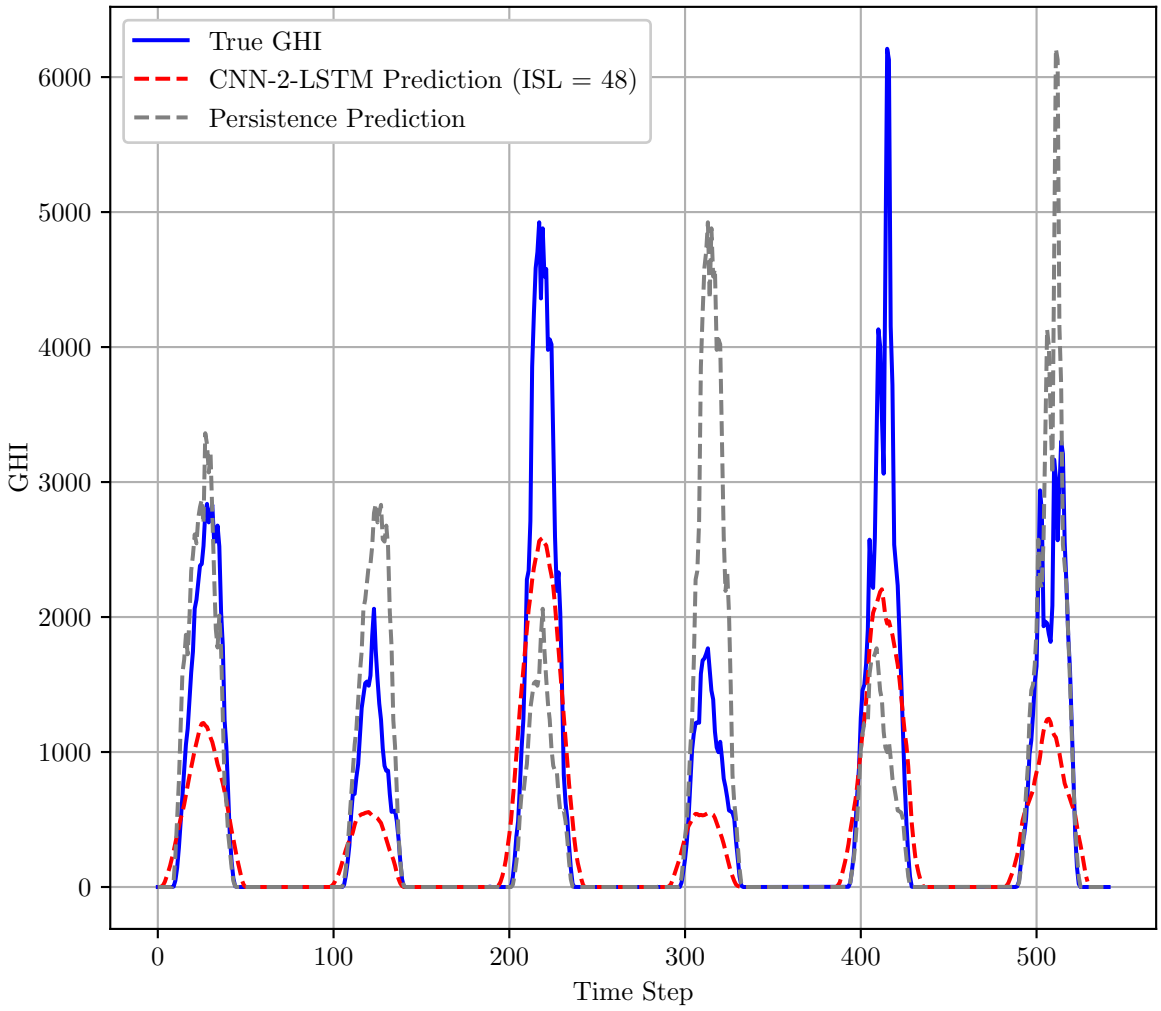

Figure 20: Predictions and True GHI for 6-Hour Forecasting Horizon

that this issue will be exacerbated at even longer forecasting horizons.

Overall, the CNN-2-LSTM model shows unsatisfactory results at the 6-hour forecasting horizon. Despite significantly outperforming benchmark models and demonstrating some forecasting skill, there is a clear need for improvement before considering real-world implementation of the model.

5.5 Conclusion of Model Performance and Evaluation

The evaluation of models at different forecasting horizons reveals the strengths and weaknesses of the CNN-2-LSTM architecture compared to traditional LSTM models and the benchmark persistence forecast. At shorter horizons, like the 15-minute interval, the CNN-2-LSTM model performs better, significantly reducing RMSE, MAE, and MBEC compared to the persistence model. This shows that including satellite cloud masks enhances short-term solar forecasting by capturing cloud-induced solar variability.

However, the CNN-2-LSTM model's effectiveness decreases at 3 and 6-hour horizons.

At the 3-hour horizon, while generally outperforming LSTM models, the CNN-2-LSTM struggles with accurately predicting peak GHI values and critical times of the day. This issue is more evident at the 6-hour horizon, where LSTM models consistently outperform CNN-2-LSTM models, indicating that satellite image inputs are less beneficial for longer-term forecasts.

6 Discussion

This section explores the outcomes of the proposed model, interpreting their scientific relevance and practical applications in the renewable energy sector. It addresses limitations affecting the findings' applicability and concludes with recommendations for future research to refine and expand the current work.

6.1 Interpretation and Implications

The findings from this thesis indicate that the CNN-2-LSTM model is highly effective at a 15-minute forecasting horizon, where it surpasses the performance of other tested models, including the pure LSTM approach. However, its effectiveness diminishes at a 3-hour forecasting horizon, where the pure LSTM model demonstrates about equal performance and is very limited at a 6-hour forecasting horizon. These results align with existing research, which suggests that while cloud cover satellite images are beneficial for short-term forecasting, their utility decreases as the forecasting horizon extends.

Theoretically, these outcomes suggest that simpler models can be as effective as more complex methods. For example, while NWP models provide detailed forecasts, they require significant computational resources. The CNN-2-LSTM model offers a less resource-intensive yet highly accurate alternative for short forecasting intervals, balancing computational complexity and forecast accuracy.

Practically, although many solar energy companies use sophisticated forecasting methods, this research highlights the potential for cost-effective alternatives. In regions like Baden-Württemberg, these findings could prompt local solar energy stakeholders to adopt new models that balance cost, complexity, and accuracy more effectively.

The CNN-2-LSTM model's effectiveness at shorter forecasting horizons could boost its adoption in scenarios requiring rapid, accurate predictions, such as energy trading and grid management. This study enhances the understanding of solar irradiance forecasting, showing that simpler, more cost-effective methods can offer significant benefits based on specific needs and conditions.

6.2 Limitations

Despite its innovative integration of data sources and forecasting horizons, several limitations must be acknowledged. The merged dataset used in this thesis was confined to the year 2022, limiting the potential to apply separate years for training and testing which could enhance the model's robustness and its predictive power. Additionally, the satellite images employed had a spatial resolution of of only three kilometers. While functional, this resolution may be inadequate for capturing finer meteorological variations crucial for more precise GHI predictions. The historical GHI data also exhibited instances of

missing values. Although various techniques were employed to address these gaps, the inconsistency in data quality could still influence the accuracy of the forecasts.

Furthermore, while the CNN-2-LSTM model outperformed both the persistence forecast model and a standalone LSTM model (at the 15-minute and 3-hour horizons), the margin of improvement was not overly substantial compared to the LSTM – especially at the 3-hour horizon. Moreover it was underperforming at the 6-hour horizon. This suggests potential ceilings in model capability or in the data used for training. The observation that validation loss was generally lower than training loss suggests an influence from the dropout technique used during the model training, which is not applied during validation. This discrepancy can make the training phase artificially harder compared to validation, potentially obscuring true model performance and underlying issues with data distribution or the training process itself. Moreover, the necessity to maintain a lightweight model architecture to manage computational demands may have restricted the exploration of more complex or potentially more effective configurations. The maximum input sequence lengths for the LSTM and CNN-2-LSTM were limited to 96, mainly due to limitations in available system memory. It is possible that better performance would have been possible if the models had had access to more previous observations, as this would have allowed them to better recognize patterns in temporal features.

Moreover, the model's training was based specifically on data from Baden-Württemberg, which may limit its applicability to other regions with different climatic and meteorological conditions. Even within Germany, Baden-Württemberg could be considered an outlier region in terms of solar irradiance, being the federal state with the highest GHI index nationwide. This geographical limitation may restrict the generalizability of the model findings and underscores the need for additional studies across various locales to verify and enhance the model's utility.

These limitations highlight the need for further research incorporating more extensive data sets, higher resolution imagery, and expanded geographical contexts. Additionally, further investigation into the architectural capabilities of hybrid models like CNN-2-LSTM could provide deeper insights into enhancing model performance across various forecasting scenarios.

6.3 Future Research

Several enhancements to the model architecture could significantly advance the forecasting capabilities demonstrated in this thesis. To improve the predictive accuracy and robustness of the proposed model, further exploration into more complex architectures is recommended. Increasing the model depth by adding more layers could capture more nuanced patterns in the data. Additionally, incorporating pretrained models through transfer learning, particularly in the CNN component, could leverage pre-existing neural

networks trained on large datasets to enhance feature extraction capabilities. Exploring other hybrid or ensemble approaches, such as attention-based mechanisms found in transformers, could also offer significant improvements.

Expanding and integrating additional data types is another crucial area for future research. Extending the dataset to include multiple years and different geographic locations would provide a more robust validation of the model's effectiveness across various conditions and improve generalization. While higher resolution satellite imagery offers greater detail, it often involves trade-offs in temporal resolution and increased costs, which must be carefully considered. Additionally, integrating other types of satellite data, such as cloud albedo measurements (capturing how much solar irradiance is reflected by clouds), and incorporating environmental factors like temperature and humidity, could provide a more comprehensive input dataset, potentially enhancing forecasting accuracy.

The adoption of new technologies could also provide substantial improvements in solar forecasting. The application of newer models, such as vision transformers, could fundamentally change how spatial-temporal data is processed in solar forecasting. Another promising direction involves using generative artificial intelligence to predict future satellite imagery (e.g., using diffusion models or generative adversarial networks for next-frame prediction) and employing these predictions as inputs to the forecasting model. This approach could dynamically and accurately mimic real-world changes in cloud cover and other atmospheric conditions without complex weather models.

Practical application and industry collaboration offer promising pathways for translating this research into operational achievements. Collaboration with solar energy providers and transmission system operators could drive the practical application of these forecasting models. These stakeholders could benefit from more precise GHI predictions for better grid management and operational efficiency. Implementing the model in real-time systems, by integrating live ground-level solar irradiance measurements and automating satellite imagery data retrieval, would significantly enhance its utility in operational settings.

7 Conclusion

This thesis explored forecasting GHI using a novel CNN-2-LSTM model. By integrating satellite imagery and historical numerical data, the primary objective was to evaluate this hybrid model's capability in improving solar energy forecast accuracy. A new financial performance metric considering balance energy prices was also introduced. The study contributes to renewable energy forecasting, providing insights into machine learning utilization.

The CNN-2-LSTM model outperformed benchmarks in forecasting GHI at 15 minutes and 3 hours but was outperformed by the pure LSTM at 6 hours. It effectively captured spatial features from satellite images and temporal patterns from historical data. Despite its performance, incremental performance improvements suggest a potential performance ceiling with current configurations and data quality.

The study's implications are significant for the renewable energy sector. Energy providers and grid operators, especially in Baden-Württemberg, can use such models to enhance operational efficiency and grid management. Accurate GHI predictions help optimize solar energy production, supporting reliable and efficient energy systems.

Limitations included a limited dataset from a single year and region, resolution constraints of satellite images, and lack of additional meteorological inputs. These limitations necessitate cautious interpretation of the findings.

Future research opportunities include exploring more sophisticated model architectures and incorporating broader data. Employing technologies like vision transformers and generative artificial intelligence in forecasting models shows promise. Expanding the geographic and temporal data scope could improve model robustness and applicability.

This thesis contributes to the United Nations' SDGs, particularly SDG 7: Affordable and Clean Energy, and SDG 13: Climate Action (United Nations Department of Economic and Social Affairs, 2023). By advancing solar energy forecasting accuracy, this research supports optimizing solar energy production, promoting renewable energy use, and reducing fossil fuel reliance. Enhanced forecasting accuracy aids in efficient grid management and energy distribution, ensuring a reliable clean energy supply. This work emphasizes leveraging machine learning technologies to address climate challenges, aligning with global efforts to mitigate climate change impacts by fostering sustainable energy practices. The insights and methodologies contribute to the global agenda for sustainable development and environmental stewardship.

In summary, this thesis advances the understanding of solar forecasting and demonstrates the practical applications and challenges of integrating machine learning into the environmental and energy sectors. The insights from this research are poised to influence future developments, driving innovations that bridge the gap between theoretical models and real-world applications.

References

R. Ahmed, V. Sreeram, Y. Mishra, and M. Arif. A review and evaluation of the state-of-the-art in PV solar power forecasting: Techniques and optimization. *Renewable and Sustainable Energy Reviews*, 124:109792, May 2020. doi: 10.1016/j.rser.2020.109792.

M. Ahsan, M. Mahmud, P. Saha, K. Gupta, and Z. Siddique. Effect of Data Scaling Methods on Machine Learning Algorithms and Model Performance. *Technologies*, 9(3): 52, July 2021. doi: 10.3390/technologies9030052.

A. Al-lahham, O. Theeb, K. Elalem, T. A. Alshawi, and S. A. Alshebeili. Sky Imager-Based Forecast of Solar Irradiance Using Machine Learning. *Electronics*, 9(10):1700, Oct. 2020.

R. Aler, R. Martín, J. M. Valls, and I. M. Galván. A Study of Machine Learning Techniques for Daily Solar Energy Forecasting Using Numerical Weather Models. In *Intelligent Distributed Computing VIII*, pages 269–278, 2015.

J. R. Andrade and R. J. Bessa. Improving Renewable Energy Forecasting With a Grid of Numerical Weather Predictions. *IEEE Transactions on Sustainable Energy*, 8(4): 1571–1580, Oct. 2017.

J. Antonanzas, N. Osorio, R. Escobar, R. Urraca, F. Martinez-de Pison, and F. Antonanzas-Torres. Review of photovoltaic power forecasting. *Solar Energy*, 136: 78–111, Oct. 2016. doi: 10.1016/j.solener.2016.06.069.

S. Atique, S. Noureen, V. Roy, V. Subburaj, S. Bayne, and J. Macfie. Forecasting of total daily solar energy generation using ARIMA: A case study. In *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0114–0119, Las Vegas, NV, USA, Jan. 2019. IEEE. doi: 10.1109/CCWC.2019.8666481.

S. Atique, S. Noureen, V. Roy, S. Bayne, and J. Macfie. Time series forecasting of total daily solar energy generation: A comparative analysis between ARIMA and machine learning techniques. In *2020 IEEE Green Technologies Conference(GreenTech)*, pages 175–180, Oklahoma City, OK, USA, Apr. 2020. IEEE. doi: 10.1109/GreenTech46478.2020.9289796.

P. Bauer, A. Thorpe, and G. Brunet. The quiet revolution of numerical weather prediction. *Nature*, 525(7567):47–55, Sept. 2015. ISSN 0028-0836, 1476-4687. doi: 10.1038/nature14956.

M. K. Behera and N. Nayak. A comparative study on short-term PV power forecasting using decomposition based optimized extreme learning machine algorithm. *Engineering Science and Technology, an International Journal*, 23(1):156–167, Feb. 2020.

K. Chen, Z. He, K. Chen, J. Hu, and J. He. Solar energy forecasting with numerical weather predictions on a grid and convolutional networks. In *2017 IEEE Conference on Energy Internet and Energy System Integration (EI2)*. IEEE, Nov. 2017.

L. Cheng, H. Zang, Z. Wei, T. Ding, R. Xu, and G. Sun. Short-term Solar Power Prediction Learning Directly from Satellite Images With Regions of Interest. *IEEE Transactions on Sustainable Energy*, 13(1):629–639, Jan. 2022. Publisher: Institute of Electrical and Electronics Engineers (IEEE).

C. T. M. Clack. Modeling Solar Irradiance and Solar PV Power Output to Create a Resource Assessment Using Linear Multiple Multivariate Regression. *Journal of Applied Meteorology and Climatology*, 56(1):109–125, Jan. 2017. doi: 10.1175/JAMC-D-16-0175.1.

J. Coiffier. *Fundamentals of numerical weather prediction*. Cambridge University Press, Cambridge, 2011. ISBN 978-1-139-19067-1.

S. Cros, O. Liandrat, N. Sebastien, and N. Schmutz. Extracting cloud motion vectors from satellite images for solar power forecasting. In *2014 IEEE Geoscience and Remote Sensing Symposium*. IEEE, July 2014.

L. Datta, M. Rizwan, and P. Anand. Comprehensive Review and Evaluation of Machine Learning Approaches for Solar PV Output Power Forecasting. In *2023 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES)*. IEEE, Apr. 2023.

G. Dellino, T. Laudadio, R. Mari, N. Mastronardi, C. Meloni, and S. Vergura. Energy production forecasting in a PV plant using transfer function models. In *2015 IEEE 15th International Conference on Environment and Electrical Engineering (EEEIC)*. IEEE, June 2015.

M. Diagne, M. David, P. Lauret, J. Boland, and N. Schmutz. Review of solar irradiance forecasting methods and a proposition for small-scale insular grids. *Renewable and Sustainable Energy Reviews*, 27:65–76, Nov. 2013. doi: 10.1016/j.rser.2013.06.042.

A. Dittmann, F. Dinger, W. Herzberg, N. Holland, S. Karalus, C. Braun, R. Zähringer, W. Heydenreich, and E. Lorenz. PV-Live dataset - Measurements of global horizontal and tilted solar irradiance, Apr. 2024.

Z. Dong, D. Yang, T. Reindl, and W. M. Walsh. Satellite image analysis and a hybrid ESSS/ANN model to forecast solar irradiance in the tropics. *Energy Conversion and Management*, 79:66–73, Mar. 2014.

C. R. Hamilton, F. Maier, and W. D. Potter. Hourly Solar Radiation Forecasting Through Model Averaged Neural Networks and Alternating Model Trees. In H. Fujita, M. Ali, A. Selamat, J. Sasaki, and M. Kurematsu, editors, *Trends in Applied Knowledge-Based Systems and Data Science*, volume 9799, pages 737–750. Springer International Publishing, Cham, 2016. ISBN 978-3-319-42006-6 978-3-319-42007-3. doi: 10.1007/978-3-319-42007-3_63. Series Title: Lecture Notes in Computer Science.

A. Hammer, D. Heinemann, E. Lorenz, and B. Lückehe. Short-term forecasting of solar radiation: a statistical approach using satellite data. *Solar Energy*, 67(1-3):139–150, July 1999.

J. G. Hernandez-Travieso, C. M. Travieso, J. B. Alonso, and M. K. Dutta. Solar radiation modelling for the estimation of the solar energy generation. In *2014 Seventh International Conference on Contemporary Computing (IC3)*. IEEE, Aug. 2014.

H. Hiser and H. Senn. Mesoscale mapping of available solar energy at the earth's surface by use of satellites. *Solar Energy*, 24(2):129–141, 1980.

S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9 (8):1735–1780, Nov. 1997. doi: 10.1162/neco.1997.9.8.1735.

K. J. Iheanetu. Solar Photovoltaic Power Forecasting: A Review. *Sustainability*, 14(24): 17005, Dec. 2022. doi: 10.3390/su142417005.

R. H. Inman, H. T. Pedro, and C. F. Coimbra. Solar forecasting methods for renewable energy integration. *Progress in Energy and Combustion Science*, 39(6):535–576, Dec. 2013. doi: 10.1016/j.pecs.2013.06.002.

H. S. Jang, K. Y. Bae, H.-S. Park, and D. K. Sung. Solar Power Prediction Based on Satellite Images and Support Vector Machine. *IEEE Transactions on Sustainable Energy*, 7(3):1255–1263, July 2016.

I. Jebli, F.-Z. Belouadha, and M. I. Kabbaj. The forecasting of solar energy based on Machine Learning. In *2020 International Conference on Electrical and Information Technologies (ICEIT)*. IEEE, Mar. 2020.

K. Kaliappan, M. Sankar, B. Karthikeyan, B. Vineeth, and V. C. Raju. Analysis of solar energy technology in leading countries. *International Journal of Power Electronics and Drive Systems (IJPEDS)*, 10(4):1995, Dec. 2019. doi: 10.11591/ijpeds.v10.i4.pp1995-2004.

J. O. Kamadinata, T. L. Ken, and T. Suwa. Sky image-based solar irradiance prediction methodologies using artificial neural networks. *Renewable Energy*, 134:837–845, Apr. 2019.

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization, 2014. Version Number: 9.

W. Kong, Y. Jia, Z. Y. Dong, K. Meng, and S. Chai. Hybrid approaches based on deep whole-sky-image learning to photovoltaic generation forecasting. *Applied Energy*, 280: 115875, Dec. 2020.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, May 2017. doi: 10.1145/3065386.

V. Kushwaha and N. M. Pindoriya. Very short-term solar PV generation forecast using SARIMA model: A case study. In *2017 7th International Conference on Power Systems (ICPS)*, pages 430–435, Pune, Dec. 2017. IEEE. doi: 10.1109/ICPES.2017.8387332.

W. F. Lamb, T. Wiedmann, J. Pongratz, R. Andrew, M. Crippa, J. G. J. Olivier, D. Wiedenhofer, G. Mattioli, A. A. Khourdajie, J. House, S. Pachauri, M. Figueroa, Y. Saheb, R. Slade, K. Hubacek, L. Sun, S. K. Ribeiro, S. Khennas, S. De La Rue Du Can, L. Chapungu, S. J. Davis, I. Bashmakov, H. Dai, S. Dhakal, X. Tan, Y. Geng, B. Gu, and J. Minx. A review of trends and drivers of greenhouse gas emissions by sector from 1990 to 2018. *Environmental Research Letters*, 16(7):073005, July 2021. doi: 10.1088/1748-9326/abee4e.

Lasanthika H. Dissawa, A. Agalgaonkar, D. Robinson, R. Godaliyadda, J. Ekanayake, Parakrama B. Ekanayake, and S. Perera. On-Site Solar Power Forecasting Using Sky-Images. *Australasian Universities Power Engineering Conference*, 2020.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov. 1998. doi: 10.1109/5.726791.

I. Majumder, M. K. Behera, and N. Nayak. Solar power forecasting using a hybrid EMD-ELM method. In *2017 International Conference on Circuit ,Power and Computing Technologies (ICCPCT)*. IEEE, Apr. 2017.

I. Majumder, R. Bisoi, N. Nayak, and N. Hannoon. Solar power forecasting using robust kernel extreme learning machine and decomposition methods. *International Journal of Power and Energy Conversion*, 11(3):260, 2020. Publisher: Inderscience Publishers.

M. Malvoni, M. G. De Giorgi, and P. M. Congedo. Forecasting of PV Power Generation using weather input datapreprocessing techniques. *Energy Procedia*, 126:651–658, Sept. 2017.

R. Marquez, H. T. Pedro, and C. F. Coimbra. Hybrid solar forecasting method uses satellite imaging and ground telemetry as inputs to ANNs. *Solar Energy*, 92:176–188, June 2013.

R. Martin, R. Aler, J. M. Valls, and I. M. Galvan. Machine learning techniques for daily solar energy prediction and interpolation using numerical weather models. *Concurrency and Computation: Practice and Experience*, 28(4):1261–1274, Aug. 2015. Publisher: Wiley.

M. Matricardi, F. Chevallier, G. Kelly, and J. Thépaut. An improved general fast radiative transfer model for the assimilation of radiance observations. *Quarterly Journal of the Royal Meteorological Society*, 130(596):153–173, Jan. 2004. doi: 10.1256/qj.02.181.

L. Mazorra-Aguiar and F. Díaz. Solar Radiation Forecasting with Statistical Models. In *Wind Field and Solar Radiation Characterization and Forecasting*, pages 171–200. Springer International Publishing, 2018.

J. F. Meirink, K.-G. Karlsson, I. Solodovnik, I. Hüser, N. Benas, E. Johansson, N. Håkansson, M. Stengel, N. Selbach, S. Marc, and R. Hollmann. CLAAS-3: CM SAF CLoud property dAtAset using SEVIRI - Edition 3, Dec. 2022. Artwork Size: 78.3 TiB Pages: 78.3 TiB.

S. Monjoly, M. André, R. Calif, and T. Soubdhan. Hourly forecasting of global solar radiation based on multiscale decomposition methods: A hybrid approach. *Energy*, 119:288–298, Jan. 2017. doi: 10.1016/j.energy.2016.11.061.

M. Z. Mukaram and F. Yusof. Solar radiation forecast using hybrid SARIMA and ANN model. *Malaysian Journal of Fundamental and Applied Sciences*, 13(4-1):346–350, Dec. 2017. doi: 10.11113/mjfas.v13n4-1.895.

P. Mathiesen, J. Kleissl, L. Spiegel, Masao Kanamitsu, and Manajit Sengupta. Evaluation of numerical weather prediction for intra-day solar forecasting in the continental United States. *Solar Energy*, 85:967–977, 2013. doi: http://dx.doi.org/10.1016/j.solener.2011.02.013.

F. Pandžić and T. Capuder. Advances in Short-Term Solar Forecasting: A Review and Benchmark of Machine Learning Methods and Relevant Data Sources. *Energies*, 17(1):97, Dec. 2023. ISSN 1996-1073. doi: 10.3390/en17010097.

C. Paoli, C. Voyant, M. Muselli, and M.-L. Nivet. Solar Radiation Forecasting Using Ad-Hoc Time Series Preprocessing and Neural Networks. In *Emerging Intelligent Computing Technology and Applications*, pages 898–907. Springer Berlin Heidelberg, 2009.

T. Parvanyan and R. Fox. World - Photovoltaic Power Potential (PVOUT) GIS Data, (Global Solar Atlas), 2019.

S. Pelland, G. Galanis, and G. Kallos. Solar and photovoltaic forecasting through post-processing of the Global Environmental Multiscale numerical weather prediction model. *Progress in Photovoltaics: Research and Applications*, 21(3):284–296, Nov. 2011. Publisher: Wiley.

A. A. Prasad and M. Kay. Prediction of Solar Power Using Near-Real Time Satellite Data. *Energies*, 14(18):5865, Sept. 2021. Publisher: MDPI AG.

R. Prasad, M. Ali, Y. Xiang, and H. Khan. A double decomposition-based modelling approach to forecast weekly solar radiation. *Renewable Energy*, 152:9–22, June 2020. Publisher: Elsevier BV.

W. Rawat and Z. Wang. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Computation*, 29(9):2352–2449, Sept. 2017. doi: 10.1162/neco_a_00990.

A. Shadab, S. Ahmad, and S. Said. Spatial forecasting of solar radiation using ARIMA model. *Remote Sensing Applications: Society and Environment*, 20:100427, Nov. 2020. doi: 10.1016/j.rsase.2020.100427.

F. Shahid, Z. Aneela, A. Mudasser, and H. Muhammad. Short term solar energy prediction by machine learning algorithms. 2020. doi: https://doi.org/10.48550/arXiv.2012.00688.

X. Shao, S. Lu, and H. F. Hamann. Solar radiation forecast with machine learning. In *2016 23rd International Workshop on Active-Matrix Flatpanel Displays and Devices (AM-FPD)*. IEEE, July 2016.

G. O. Shoaga, A. Ikuzwe, and A. Gupta. Forecasting of Monthly Hydroelectric and Solar Energy in Rwanda using SARIMA. In *2022 IEEE PES/IAS PowerAfrica*, pages 1–5, Kigali, Rwanda, Aug. 2022. IEEE. ISBN 978-1-66546-639-4. doi: 10.1109/PowerAfrica53997.2022.9905311.

Z. Si, M. Yang, and Y. Yu. Hybrid Solar Forecasting Method Using Satellite Visible Images and Modified Convolutional Neural Networks. In *2020 IEEE/IAS 56th Industrial and Commercial Power Systems Technical Conference (I&CPS)*. IEEE, June 2020.

C. V. Silva, L. Lim, D. Stevens, and D. Nakafuji. Probabilistic Models for One-Day Ahead Solar Irradiance Forecasting in Renewable Energy Applications. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. IEEE, Dec. 2015.

E. Subramanian, M. M. Karthik, G. P. Krishna, D. V. Prasath, and V. S. Kumar. Solar Power Prediction Using Machine Learning. 2023. Publisher: arXiv.

T. Schmidt. *High resolution solar irradiance forecasts based on sky images*. PhD thesis, Carl von Ossietzky Universität Oldenburg, 2017.

K. Tangwongsan, M. Hirzel, and S. Schneider. Sliding-Window Aggregation Algorithms. In A. Zomaya, J. Taheri, and S. Sakr, editors, *Encyclopedia of Big Data Technologies*, pages 1–7. Springer International Publishing, Cham, 2022. ISBN 978-3-319-63962-8. doi: 10.1007/978-3-319-63962-8_157-2.

TransnetBW, Amprion, TenneT, and 50Hertz. Berechnung des regelzonenübergreifenden einheitlichen Bilanzausgleichsenergiepreises (reBAP) Modellbeschreibung, Dec. 2022. URL https://www.transnetbw.de/_Resources/Persistent/a/9/7/e/a97e1271f054e5584fe47c090b060fcd4841427c/Ermittlung%20reBAP%20und%20Umgang%20mit%20Korrekturen%20%28gltig%20ab%20dem%2008.12.2022%29.pdf.

TransnetBW, Amprion, TenneT, and 50Hertz. Uniform Imbalance Price (reBAP) Historic Data, 2024. URL https://www.netztransparenz.de/en/Balancing-Capacity/Imbalance-price/Uniform-imbalance-price-reBAP.

United Nations Department of Economic and Social Affairs. *Sustainable Development Goals Report 2023*. United Nations, 2023. ISBN 978-92-1-101460-0.

V. Jayadevan, Jeffrey J. Rodríguez, V. Lonij, and A. Cronin. Forecasting solar power intermittency using ground-based cloud imaging. 2012.

C. Vennila, A. Titus, T. S. Sudha, U. Sreenivasulu, N. P. R. Reddy, K. Jamal, D. Lakshmaiah, P. Jagadeesh, and A. Belay. Forecasting Solar Energy Production Using Machine Learning. *International Journal of Photoenergy*, 2022:1–7, Apr. 2022. Publisher: Hindawi Limited.

H. Verbois, R. Huva, A. Rusydi, and W. Walsh. Solar irradiance forecasting in the tropics using numerical weather prediction and statistical learning. *Solar Energy*, 162:265–277, Mar. 2018. Publisher: Elsevier BV.

Z. Wang, C. Tian, Q. Zhu, and M. Huang. Hourly Solar Radiation Forecasting Using a Volterra-Least Squares Support Vector Machine Model Combined with Signal Decomposition. *Energies*, 11(1):68, Jan. 2018. ISSN 1996-1073. doi: 10.3390/en11010068.

H. Wen, Y. Du, X. Chen, E. Lim, H. Wen, L. Jiang, and W. Xiang. Deep Learning Based Multistep Solar Forecasting for PV Ramp-Rate Control Using Sky Images. *IEEE Transactions on Industrial Informatics*, 17(2):1397–1406, Feb. 2021.

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

D. Yu, S. Lee, S. Lee, W. Choi, and L. Liu. Forecasting Photovoltaic Power Generation Using Satellite Images. *Energies*, 13(24):6603, Dec. 2020.

M. D. Zeiler and R. Fergus. Visualizing and Understanding Convolutional Networks, Nov. 2013. arXiv:1311.2901 [cs].

G. Zhang, D. Yang, G. Galanis, and E. Androulakis. Solar forecasting with hourly updated numerical weather prediction. *Renewable and Sustainable Energy Reviews*, 154:111768, Feb. 2022. doi: 10.1016/j.rser.2021.111768.

W. Zhang, H. Dang, and R. Simoes. A new solar power output prediction based on hybrid forecast engine and decomposition model. *ISA Transactions*, 81:105–120, Oct. 2018.

P. Zhao and W. Tian. Research on Prediction of Solar Power Considering the Methods of Statistical and Machine Learning – Based on the Data of Australian Solar Power Market. *IOP Conference Series: Earth and Environmental Science*, 1046(1):012006, June 2022.

Z. Zhen, Z. Wang, F. Wang, Z. Mi, and K. Li. Research on a cloud image forecasting approach for solar power forecasting. *Energy Procedia*, 142:362–368, Dec. 2017.

A Data Generators Code

```
# Load numerical data
numerical_data = pd.read_csv(drive_path + 'pyrano_data_avg_minmax.csv')
numerical_data = numerical_data.rename(columns={"00_AVG": "Gg_pyr"})
scaler = MinMaxScaler()
numerical_data[["Gg_pyr", "month", "day", "hour", "minute"]] = scaler.fit_transform(
    numerical_data[["Gg_pyr", "month", "day", "hour", "minute"]])

# Load image data if needed
image_data = None
if any(image_modes):
    npz_file = np.load(drive_path + 'year22_cma_bw_64.npz')
    image_data = np.array([np.mean(npz_file[file], axis=2) for file in npz_file.files])

# Prepare data function
def prepare_data(input_sequence_length, target_sequence_length, image_mode):
    if image_mode:
        image_data_reshaped = image_data.reshape(35040, 4096)
        comb_array = np.concatenate([numerical_data, image_data_reshaped], axis=1)
    else:
        comb_array = numerical_data

    train_data, test_data = train_test_split(comb_array, test_size=0.1, shuffle=False)
    train_data, val_data = train_test_split(train_data, test_size=0.2, shuffle=False)

    total_sequence_length = input_sequence_length + target_sequence_length
    batch_size = 32

    train_gen = tf.keras.preprocessing.timeseries_dataset_from_array(
        data=train_data[:-target_sequence_length],
        targets=None,
        sequence_length=total_sequence_length,
        sequence_stride=1,
        sampling_rate=1,
        batch_size=batch_size,
        shuffle=True)

    val_gen = tf.keras.preprocessing.timeseries_dataset_from_array(
```

```
        data=val_data[:-target_sequence_length],
        targets=None,
        sequence_length=total_sequence_length,
        sequence_stride=1,
        sampling_rate=1,
        batch_size=batch_size,
        shuffle=True)

    test_gen = tf.keras.preprocessing.timeseries_dataset_from_array(
        data=test_data[:-target_sequence_length],
        targets=None,
        sequence_length=total_sequence_length,
        sequence_stride=1,
        sampling_rate=1,
        batch_size=batch_size,
        shuffle=False)

    def reshape_sequences(batch):
        input_features = batch[:, :input_sequence_length, :]
        numerical_data_seq = input_features[:, :, :5]
        if image_mode:
            image_data_flat = input_features[:, :, 5:]
            image_data_seq = tf.reshape(image_data_flat, (-1, input_sequence_length,
                ↪ 64, 64, 1))
        target_features = batch[:, input_sequence_length:input_sequence_length +
            ↪ target_sequence_length, 0]
        target_features = tf.expand_dims(target_features, axis=-1)
        if image_mode:
            return (image_data_seq, numerical_data_seq), target_features
        else:
            return (numerical_data_seq), target_features

    train_gen = train_gen.map(reshape_sequences)
    val_gen = val_gen.map(reshape_sequences)
    test_gen = test_gen.map(reshape_sequences)

    return train_gen, val_gen, test_gen
```

B CNN-2-LSTM Code

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.callbacks import TensorBoard, ReduceLROnPlateau,
    ↪ EarlyStopping
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, TimeDistributed, Conv2D, MaxPooling2D,
    ↪ Flatten, LSTM, BatchNormalization, Concatenate, Dense
import datetime



# Defining RMSE and make it saveable to .keras models
@tf.keras.saving.register_keras_serializable()
def rmse(y_true, y_pred):
    return tf.sqrt(tf.reduce_mean(tf.square(y_true - y_pred)))

def create_model(input_sequence_length, target_sequence_length, image_mode):

    if image_mode == True:
      # CNN-LSTM Branch for Image Processing (only added when image_mode ==
          ↪ True)
      img_input = Input(shape=(input_sequence_length, 64, 64, 1), name='img_input')
      img_conv_1 = TimeDistributed(Conv2D(32, (3, 3), activation='relu'))(img_input)
      img_pool_1 = TimeDistributed(MaxPooling2D(pool_size=(2, 2)))(img_conv_1)
      img_conv_2 = TimeDistributed(Conv2D(16, (3, 3), activation='relu'))(img_pool_1
          ↪ )
      img_pool_2 = TimeDistributed(MaxPooling2D(pool_size=(2, 2)))(img_conv_2)
      img_flatten = TimeDistributed(Flatten())(img_pool_2)
      img_lstm_1 = LSTM(32, return_sequences=True)(img_flatten)
      img_lstm_2 = LSTM(32, return_sequences=True)(img_lstm_1)
      img_dropout_1 = TimeDistributed(Dropout(0.2))(img_lstm_2)
      img_lstm_3 = LSTM(64, return_sequences=True)(img_dropout_1)
      img_lstm_4 = LSTM(64, return_sequences=False)(img_lstm_3)
      img_norm = BatchNormalization()(img_lstm_4)

    # LSTM Branch for Numerical Features
    num_input = Input(shape=(input_sequence_length, 5), name='num_input')
    num_lstm_1 = LSTM(32, return_sequences=True)(num_input)
```

```
    num_lstm_2 = LSTM(32, return_sequences=True)(num_lstm_1)
    num_dropout_1 = TimeDistributed(Dropout(0.2))(num_lstm_2)
    num_lstm_3 = LSTM(64, return_sequences=True)(num_dropout_1)
    num_lstm_4 = LSTM(64, return_sequences=False)(num_lstm_3)
    num_norm = BatchNormalization()(num_lstm_4)


    # Combining outputs if image_mode == True
    if image_mode == True:
      combined = Concatenate()([num_norm, img_norm])
    else:
      combined = num_norm

    # Fully connected layers
    dense_1 = Dense(16, activation='linear')(combined)
    dense_2 = Dense(8, activation='linear')(dense_1)
    output = Dense(target_sequence_length, activation='relu')(dense_2)

    # Define model before compilation
    if image_mode == True:
      model = Model(inputs=[img_input, num_input], outputs=output)
    else:
      model = Model(inputs=[num_input], outputs=output)

    # Compile model
    model.compile(optimizer=tf.keras.optimizers.Adam(), loss= rmse, metrics=['mae'])

    return model

def train_model(input_sequence_length, target_sequence_length, image_mode, train_gen,
    ↪ val_gen, test_gen, epoch_limit):
  model = create_model(input_sequence_length, target_sequence_length, image_mode =
      ↪ image_mode)
  model.summary()


  reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5, min_lr
      ↪ =0.000001)
```

```
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights
    ↪ =True)

# Train the model
history = model.fit(
    train_gen,
    epochs=epoch_limit,
    validation_data=val_gen,
    callbacks=[reduce_lr, early_stopping]
)

return history, model
```

C Sliding Window Aggregation for Overlapping Predictions Code

```
def aggregate_predictions(model, data_gen, input_sequence_length,
    ↪ target_sequence_length, steps):
# Initialize lists to hold predictions and actual targets
all_preds = []
all_targets = []

# Iterate through the data generator
step = 0
for (num_data), targets in data_gen:
    if step >= steps:
        break

    # Make predictions using the model
    preds = model.predict([num_data], verbose = 0)

    # Append the predictions and actual targets to the lists
    all_preds.append(preds)
    all_targets.append(targets.numpy())
    step += 1

# Concatenate all the predictions and targets
all_preds = np.concatenate(all_preds, axis=0)
all_targets = np.concatenate(all_targets, axis=0)

# Initialize arrays for aggregated predictions and counts
aggregated_preds = np.zeros((all_targets.shape[0] + target_sequence_length - 1, 1))
counts = np.zeros((all_targets.shape[0] + target_sequence_length - 1, 1))

# Aggregate the predictions
for i in range(len(all_preds)):
    for j in range(target_sequence_length):
        if i + j < len(aggregated_preds):
            aggregated_preds[i + j] += all_preds[i, j]
            counts[i + j] += 1
```

```
    # Compute the final averaged predictions
    aggregated_preds /= counts

    # Return the aggregated predictions and actual targets
    return aggregated_preds[:len(all_targets)], all_targets
```

D Model Training and Evaluation Code

```
def automate_training(input_sequence_lengths, target_sequence_lengths, epoch_limit,
    ↪ batch_size, image_modes):
  for image_mode in image_modes:
    for input_sequence_length in input_sequence_lengths:
      for target_sequence_length in target_sequence_lengths:
        train_gen, val_gen, test_gen = prepare_data(input_sequence_length, image_mode)
        print("Training Model | ISL:" + str(input_sequence_length) + " | TSL:" +
            ↪ str(target_sequence_length) + " | Image Mode:", image_mode)
        history, model = train_model(input_sequence_length, target_sequence_length =
            ↪ target_sequence_length, image_mode = image_mode, train_gen =
            ↪ train_gen, val_gen = val_gen, test_gen = test_gen, epoch_limit =
            ↪ epoch_limit)

        test_steps = np.ceil(3504 / batch_size).astype(int)



        print("Aggregating predictions...")
        aggregated_preds, actual_targets = aggregate_predictions(
            model, test_gen, input_sequence_length, target_sequence_length, test_steps
        )
        plt.figure(figsize=(6.299212813062128, 5))
        plt.plot(aggregated_preds[], label = ('ISL:'+str(input_sequence_length)))
        plt.plot(actual_targets[].reshape(672,1), label = 'True GHI')
        plt.xlabel('Timestep')
        plt.ylabel('Normalized GHI')
        plt.legend()
        plt.show()
        print("Evaluating model performance...")

        model_rmse, model_mse, model_mae = model.evaluate(test_gen, steps=
            ↪ test_steps, verbose = 0)

        print("RMSE:", model_rmse, " MSE:", model_mse, " MAE:", model_mae)
        # Plot training vs validation loss
        plt.figure(figsize=(6.299212813062128, 3))
        plt.plot(history.history['loss'], label='Training Loss')
        plt.plot(history.history['val_loss'], label='Validation Loss')
```

```
        plt.xlabel('Epochs')
        plt.ylabel('RMSE')
        plt.legend()
        plt.show()

        with open(f'/content/drive/MyDrive/0_C9/{image_mode}_ISL_{
            ↪ input_sequence_length}_TSL_{target_sequence_length}_history', 'wb') as
            ↪ file_pi:
          pickle.dump(history.history, file_pi)

        print("Saving model...")
        model_save_path = f'/content/drive/MyDrive/0_C9/{image_mode}_ISL_{
            ↪ input_sequence_length}_TSL_{target_sequence_length}.keras'
        model.save(model_save_path)
        print("Model saved to", model_save_path)
        print("-------------------------------------------------")




input_sequence_lengths = [1, 4, 8, 16, 48, 96]
target_sequence_lengths = [1, 12, 24]
image_modes = [True, False]

automate_training(input_sequence_lengths, target_sequence_lengths, 50, 32, image_modes
    ↪ )
```