HELLENIC REPUBLIC
National and Kapodistrian University of Athens
EST. 1837

National and Kapodistrian University of Athens
Physics Department
MSc Control & Computing
Master's Thesis

XceptionLSTM: Advanced Sequence-to-Sequence Neural Networks for Short-Term Weather Forecasting Applications

Georgios Venitourakis
7110132100202

Supervisor
Dionysios Reisis, Professor

Evaluation Commitee
Dionysios Reisis, Professor
Anna Tzanakaki, Associate Professor
Dr. Nikolaos Vlassopoulos, Research Associate

January 2024

Abstract

Renewable Energy has been the main global focus of the last two decades, as the energy industry searches for ways to steadily replace fossil fuels with greener energy sources. This transition paved the way to new concepts and techniques for controlling the power production and distribution in renewable energy parks. For improved efficiency, the controller in the Smart Grid concept often acts proactively, where it predicts events in the near future and prepares the underlying infrastructure for the upcoming event.

In the case of photovoltaic parks, a smart system makes short-term weather forecasts about the global horizontal irradiance and the cloud cover in the park's area of interest. This thesis introduces a neural network as a solution to the short-term weather forecasting problem. The proposed model is an image regression recurrent neural network in the form of a spatio-temporal encoder/decoder. The basis of the structure is the Xception layer, which utilizes depwthwise and pointwise convolutions to infer data. The Xception layer is combined with long short-term memory cells to create a recurrent neural network with improved forecasting capabilities. The proposed model is optimized for inference on the edge and is evaluated on the *Archon - Athens, Greece* dataset.

Keywords: deep learning; ConvLSTM; irradiance forecasting; edge computing; photovoltaic parks; ground-based sky images

Περίληψη

Η παγκόσμια τάση των τελευταίων δύο δεκαετιών για ανεξαρτητοποίηση από κάυσιμες πηγές ενέργειας έχει οδηγήσει τη βιομηχανία ενέργειας προς την αξιοποίηση ανανεώσημων πηγών ενέργειας. Η μετάβαση σε πιο πράσινη ενέργεια αποτελεί έμπνευση ιδέων και πρακτικών που αποσκοπούν στον έλεγχο της παραγωγής και διανομής ενέργειας στα πάρκα ενέργειας. Για μεγαλύτερη αποδοτικότητα, αυτοματοποιημένοι ελεγχτές στα έξυπνα δικτύα ισχύος δύναται να δρουν προληπτικά με σκοπό την πρόβλεψη απερχόμενων συμβάντων και την προετοιμασία της υποκείμενης υποδομής για την αντιμετώπιση αυτών.

Στην περίπτωση των φωτοβολταϊκών πάρκων, το έξυπνο σύστημα ελέγχου ενός πάρκου εκτελεί προβλέψεις μικρού ορίζοντα για την ολική ηλιακή ακτινοβολία στο οριζόντιο επίπεδο και την νεφοκάλυψη στην ευρύτερη περιοχή του πάρκου. Η παρούσα εργασία παρουσιάζει ένα νευρωνικό δίκτυο ως λύση για το πρόβλημα της πρόβλεψης καιρού μικρού ορίζοντα. Το προτεινόμενο μοντέλο είναι ένα επαναλαμβανόμενο νευρωνικό δίκτυο για παλινδρόμηση εικόνων σε μορφή κωδικοποιητή-αποκωδικοποιητή χώρου-χρόνου. Η βάση του μοντέλου είναι το επίπεδο Xception, το οποίο αξιοποιεί κατά βάθος και κατά σημείο συνελίξεις για να εξάγει προβλέψεις. Το επίπεδο Xception συνδυάζεται με κύτταρα μακράς βραχυπρόθεσμής μνήμης για τη δημιουργία ενός επαναλαμβανόμενου νευρωνικού δικτύου με βελτιωμένες ικανότητες πρόβλεψης. Το προτεινόμενο μοντέλο έχει βελτιστοποιηθεί για ακροδικτυακή υπολογιστική και έχει αξιολογηθεί στο σύνολο δεδομένων *ΑΡΧΩΝ - Αθήνα, Ελλάδα*.

Λέξεις κλειδιά: βαθιά μηχανική μάθηση; συνελικτικά αναδρομικά νευρωνικά δίκτυα; πρόβλεψη ακτινοβολίας; ακροδικτυακή υπολογιστική; φωτοβολταϊκά πάρκα; επίγειες εικόνες ουράνιου θόλου

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank the Digital Systems Team leader and my supervisor, Professor Dionysios Reisis, as well as the Inaccess Networks for funding and supporting this research with the necessary equipment, and for giving me the opportunity to participate in the Archon Project. This research has been co-financed by the European Regional Development Fund of the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH—CREATE—INNOVATE (project name "ARCHON" and project code: T2EDK-00864). The results of this thesis were published on 18 November 2023 in an open access article in *Information*

I would also like to express my deepest gratitude and love to my parents and sisters for their continuous support and guidance in my decision to pursue my master's degree. Finally, I'm deeply indepted to my brothers from another mother, Chris, Alex, Kostas and Alex Jr, for helping me become a better version of myself and for showing me that life is too short to worry about things I can't change.

Chapter 1

Introduction

The overall increasing need for sustainable energy has led researchers and the energy industry in search for new tools to better manage energy resources. Such tools often include controllers and sensors to monitor the energy production infrastructure and detect any worrisome activity. However, real-time systems may not be flexible enough to react to the environment's changes and detect a problem in a short notice to minimize its effects. Thus, a different approach is considered, where all sensor measurements from the near past are collected and provided to the system's controller. The controller can then make predictions about the near future and take actions proactively, or be prepared for an issue it has predicted to occur. Such smart systems are mostly logically and physically centralised, since predicting the state of a system can be computationally intense and decision making is heavily dependent on it.

In energy parks, predicting the power output can help balance the main grid's power and achieve an overall better power management and energy storage. By doing so, the power distributor and the photovoltaic (PV) park manager benefit from less stressed infrastructure and energy losses, but also from providing smoother services to their corresponding clients. The Smart Grid (SG) concept relies on controllers that can both proactively and reactively make decisions about the power distribution and the configuration of the infrastructure in their scope. This configuration refers to controlling the characteristics of nearby devices. For example, when a gust is expected within the next few minutes, the SG controller informs the nearby wind turbine, working as a secondary and more stable power source, to slowly increase the moment of inertia, so that the frequency of the produced alternating current (AC) would not be heavily disrupted. Short-term weather forecasting is a mission critical task for the PV park controller to base its proactive decisions. There are analytical solutions for weather forecasting, but researchers usually avoid these solutions because of the heavy computational needs and time constraints they set. Machine Learning (ML) based solutions are considered instead, since their execution times and computational complexity best suit the problem. In both cases, the solutions are statistical and a dataset is needed to correlate the forecasts to the ground truth. The Earth's atmosphere is highly unstable; developing a forecasting NN for a system with uncountably many parameters and dependencies (known and unknown) may not be as promising as it sounds. We present in this work that NN-based solutions offer a tempting alternative for short-term weather forecasting and that they can be tailored for both centralized and edge computing applications.

The purpose of this thesis is to provide a method in order for these SG controllers to predict the future state of the systems they monitor. In this thesis, a camera is considered as a sensor that captures the sky dome of a PV park. By parsing the images of the past few minutes with the help of Neural Networks (NN), we can make forecasts about meteorological parameters that directly influence power production outage. We will focus our research on global horizontal irradiance (GHI).

This thesis focuses on image regression based ML techniques that exploit Sequence-to-Sequence (Seq2Seq) recurrent neural networks (RNN) in the form of spatio-temporal encoders/decoders [1]. Specifically, we explore convolutional neural networks (CNN) and their depthwise separable counterparts, the depthwise separable convolutions (DWSC) in order to significantly improve the training effort, as well as the inference execution time and the evaluation results of the proposed model. Also, we study the Xception Layer (XL) [2] as a state-of-the-art layer for deep learning (DL) based applications and combine it with long short-term memory (LSTM) cell, formally named XceptionLSTM. Finally, we introduce the proposed model, a spatio-temporal model that utilizes XLs to form a spatial encoder, XceptionLSTM cells for the temporal ecoders and decoders and a multilayer perseptron (MLP) for the spatial decoder.

The work is presented as follows. First, Chapter 2 presents the recent works in the scope of weather forecasting for RES park controlling. Chapter 3 is an overview of the techniques used throughout this work. It also introduces the dataset used for the training and the evaluation of the tested models. Chapter 4 follows with the training scheme and the presentation of the evaluation results for a set of spatio-temporal models we benchmarked. Chapter 5 concludes the thesis with a discussion of the results and future work.

Chapter 2

Related Work

Researchers and engineers in the renewable energy field are keen for solutions to the short-term irradiance forecasting problem [3]. Especially in the last two decades, they focus on Computer Vision and ML-based systems, which often include image processing for satellite imagery [4, 5, 6, 7]. Given that the satellite images cover a vast area of the Earth's and, measuring the GHI in different areas of an image may provide significantly different values. The alternative is the ground-based imagery [8, 9, 10], which clearly depicts the current weather conditions in the area of interest with a notable application example the case of large PV parks. The PV park controllers use multiple ground-based sensors and they can yield more accurate results [11].

As Ziyabari et al. [1] suggest, researchers often consider spatio-temporal architectures as a solid base for their models because the dimensionality of the input data does not constrain considerably the final structure of their models. This holds whether the input data is multiple time-series of environmental measurements from multiple sensors that are spread in a wide area or, as in this article, a time-series of images from a single sky camera. ConvLSTM-based solutions are reported in image regression related techniques that target irradiance forecasting [12, 13, 14]. This is because CLs are effective in modelling the complex dynamics of the environmental variables, such as the cloud and the wind movement. Moreover, as the name suggests, ConvLSTMs can capture the long-term evolution of the irradiance values. More accurately, ConvLSTMs excel in modeling the long-term dependencies of the target data and extract the correlation among the input data [10]. It is quite common for researchers to utilize image segmentation for cloud cover estimations as a means to enhance the results of ML-based forecasting models [15, 16].

Zhang et al. [17] compare the results of MLP, CNN and LSTM models that are trained to predict PV power differences by using PV power data and sky images. They conclude that a hybrid model using both PV power data and images has a better-balanced performance across different types of weather conditions. Sun et al. [18] present the SUNSET, a deep CNN architecture that accepts an image sequence and other data produced by the PV park and it outputs PV power and Clear Sky Index (CSI) predictions. The input image sequence is in the form of a single hyperspectral image. Ajith et al. [19] developed a multi-modal fusion network for ultra-short irradiance forecasting using infrared images and past irradiance data. They explain that infrared images of the sky can better capture the cloud dynamics in the small horizon of 15 seconds. Kumari et al. [10] discuss the advantages and drawbacks of using LSTMs, Gated Reccurrent Units (GRU),

CNNs, Deep Belief Networks (DBN), RNNs and Hybrid Artificial Neural Networks (ANN) for solar irradiance forecasting. Basmile et al. [20] review and compare eight different AI models for horizons of a minute, an hour and for daily average forecasts of GHI, DHI and DNI values. Nie et al. [14] explore training tactics for heterogeneous datasets and how transfer learning contributes to reducing the training effort and improving the results of a model. Lyu et al. [21] use deep reinforcement learning (DRL) in order to dynamically change between optimal features of a model by recognising weather patterns.

Chapter 3

Problem Definition

For the purposes of this work, we assume that the state of the target system solely depends on the recent history of the system. Based on this assumption, we can make forecasts about the next state of the target system if we have records of the system's states from the near past. In this work, we aim to predict GHI values in the near future. A sequence of images captured consecutively with a constant time interval (referred to as horizon from now on) is forwarded to a NN. The model outputs a new sequence of values corresponding to consecutive GHI values with the same horizon as the input sequence. The input and output sequence length, as well as the horizon used for the prediction are some of the model's hyperparameters, and can be fine-tuned to achieve the best possible outcome. Other hyperparameters are the model's structure, the training schemes and any data preprocessing.

3.1 Model Structure

A NN need to be complex enough to be able to simulate the system it is trained to describe. Therefore, in highly complex systems such as the Earth's atmosphere, the traditional models that can respond to such fast-changing parameters need a great number of parameters and they are time consuming to train and infer results. This calls for state-of-the-art ML algorithms and techniques that provide sufficient complexity without the intense computational resources their predecessors would need for the same task.

3.1.1 Xception Layer

The proposed model utilises Xception Layers [2], a type of Convolutional NN that combines the characteristics of Inception Modules [22] and Depthwise Separable Convolutions (DWSC) [2, 23]. A DWSC extracts the parallelism of a traditional convolutional layer (CL) by partitioning the operation in two, simpler operations, a depthwise convolution (DWC) and a pointwise convolution (PWC). The former is a convolution in each frame of the channels of the input tensor, while the latter is a convolution in each pixel of the input tensor. Combining the DWC and PWC sequentially results in a CL with the same result-producing capabilities, but much more lightweight in terms of total number of parameters and computational intensity. The computational graph of a DWSC is depicted in Figure 3.1. We note here that for every CL there are two unique and equivalent DWSC

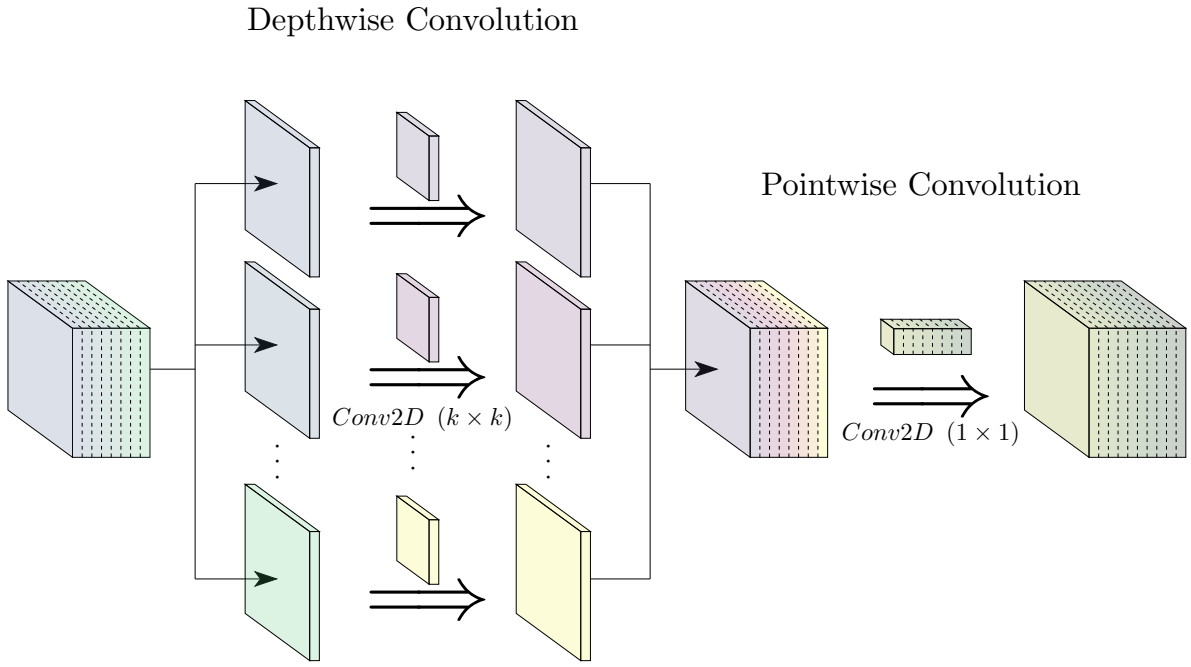


Figure 3.1: Depthwise separable convolution breakdown.

modules, specifically one module that starts with a DWC and ends with a PWC and one more that starts with a PWC and ends with a DWC. Also, the execution order of the PWC and DWC does not affect the result of a DWSC, but may affect the computational complexity, the backpropagation during training and the time and memory overhead of the layer.

An Inception Module consists of nested CL, where all nested layers process the same input in parallel, and all results are concatenated, added or in general reduced to a new output tensor. By parallelising layers, we can achieve a higher level of parallelism, there is a greater degree of data usage and, thus, less memory communication overhead. Inception Modules also help to alleviate the vanishing gradient problem [24], in which gradient becomes insignificant in the first layers of a model during backpropagation when training models with a great number of sequentially connected layers.

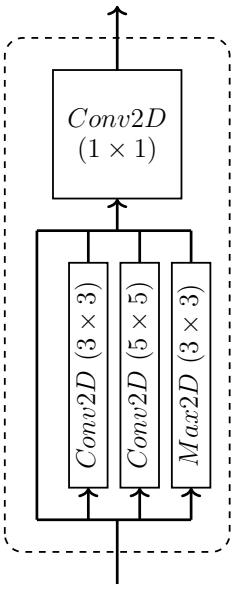
The combination of DWSC and Inception Modules results in Xception Layers (XL), where depthwise operations such as DWC or pooling operations are executed in parallel and then their results are concatenated and forwarded to a PWC. We can also consider the transposed XL, where the result of a PWC is split into chunks that are then forwarded to depthwise operations. This scheme exploits two forms of parallelism, the inter-task parallelism (parallel execution of nested layers in Inception Module) and intra-task parallelism (parallel execution of convolutions in every channel of the input tensor in a Depthwise Convolution.) Although intra-task parallelism is already present in CL, due to the independence of the operations between the kernels of a CL (inter-kernel parallelism,) a greater degree of intra-task parallelism can be achieved with XL and DWCs. This is because the DWCs operate on all channels independently, and intra-task parallelism is achieved by executing the convolutions of each channel in parallel.

The clear advantage of XL in favour of the traditional CL, when compared based on NNs with similar characteristics, is better data management, less overhead in terms of memory access and less number of total parameters and operations. All these advantages can be even further improved by partitioning DWC in two asymmetrical DWC, one for each dimension of the frame, meaning convolutions with kernel size $1 \times N$ and $N \times 1$ [25], where $N \times N$ the kernel size of the traditional CL. Although it is easily applicable, we note that splitting a CL in its three dimensions is out of scope for this work, since its benefits are more apparent in kernels of larger sizes.

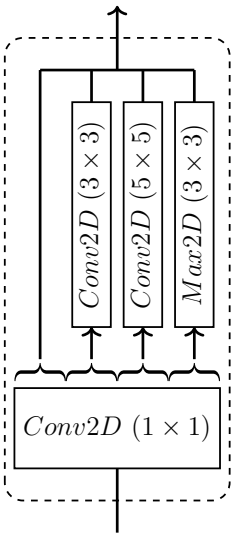
The structures of the XL that are considered throughout this work are shown in Figure 3.2a. It consists of four nested layers, two of which are DWCs with kernel size 3 and 5, a Max pooling layer and the identity function. The identity function is ultimately used as a PWC of the XL's input tensor. Also, using the identity function helps with gradient descent; the gradient is broadcasted to and propagates through all four nested layers, but is unaffected by the identity function. The gradients of all nested layers are then added and (back)propagated to the previous layer. As a result, the gradient of a XL is mostly affected by the PWC and less affected by the nested depthwise operations (they can be viewed as small adjustments in the output gradient.) Thus, layers in the later stages of backpropagation are less likely to experience the vanishing gradient descent problem.

The mathematical equivalent of the XL depicted in Figure 3.2a is the set of Equations (3.1) to (3.5). All three depthwise operations (Equations (3.1) to (3.3)) are padded with a stride of 1 and produce a tensor of the same size as the input tensor. Equation (3.4) is the reduction of the depthwise operations, which, in this work, is the concatenation of their results. Equation (3.5) describes the PWC of the reduced tensor.

A transposed version of the XL is depicted in Figure 3.2b. Although equivalent, choosing between the direct and the transposed version of XL has an impact in the total number of computations during inference. Backpropagation is also affected when choosing between the two forms, but further analysis is needed to determine the advantages and drawbacks in terms of forecasting performance and inference and training time.



(a) Direct form.



(b) Transposed form.

Figure 3.2: Structure of the implemented Xception Layer in its a) direct and b) transposed form.

3.1.2 ConvLSTM

Convolutional Long-Short Memory (ConvLSTM) Cells [26] are Recurrent NN (RNN) structures that utilize convolutions and operate on tensors, in contrast to LSTMs [27], which operate on vectors. LSTM cells are mainly used when a sequence of data is being processed or generated, and can be found in Deep Learning (DL) based Natural Language Processing (NLP) applications [28] and Time-Series Predictions [29, 30]. ConvLSTM Cells

$$x_{conv3_{ijk}} = b_{conv3_k} + \sum_{i',j'=-1}^{1} \text{pad}(x)_{i+i',j+j',k} \times W_{conv3_{i'j'k}} \tag{3.1}$$

$$x_{conv5_{ijk}} = b_{conv5_k} + \sum_{i',j'=-2}^{2} \text{pad}(x)_{i+i',j+j',k} \times W_{conv5_{i'j'k}} \tag{3.2}$$

$$x_{max3_{ijk}} = \max_{i',j'=-1}^{1} \text{pad}(x)_{i+i',j+j',k} \tag{3.3}$$

$$x_{ext_{ijk}} = \begin{cases} x_{ijk'} & , \quad k' = k \in [0, k_{max}) \\ x_{conv3_{ijk'}} & , \quad k' = k - k_{max} \in [0, k_{max}) \\ x_{conv5_{ijk'}} & , \quad k' = k - 2k_{max} \in [0, k_{max}) \\ x_{max3_{ijk'}} & , \quad k' = k - 3k_{max} \in [0, k_{max}) \end{cases} \tag{3.4}$$

$$y_{ijk} = b_{conv1_k} + \sum_{k'} x_{ext_{ijk'}} \times W_{conv1_{k'k}} \tag{3.5}$$

---

are able to process more complex tasks, such as Next Frame Prediction [31] and other Time-Series Predictions with feature extraction [32]. ConvLSTM Cells can be described by Equations (3.6) to (3.15):

$$x = \text{Concat}(X_t, H_{t-1}) \tag{3.6}$$

$$i_c = x \circledast W_{ci} + b_{ci} \tag{3.7}$$

$$f_c = x \circledast W_{cf} + b_{cf} \tag{3.8}$$

$$c_c = x \circledast W_{cc} + b_{cc} \tag{3.9}$$

$$o_c = x \circledast W_{co} + b_{co} \tag{3.10}$$

$$i_g = \sigma\left(i_c + C_{t-1} \odot W_{hi}\right) \tag{3.11}$$

$$f_g = \sigma\left(f_c + C_{t-1} \odot W_{hf}\right) \tag{3.12}$$

$$o_g = \sigma\left(o_c + C_{t-1} \odot W_{ho}\right) \tag{3.13}$$

$$C_t = f_g \odot C_{t-1} + i_g \odot \text{act}\left(c_c\right) \tag{3.14}$$

$$H_t = o_g \odot \text{act}\left(C_t\right), \tag{3.15}$$

where the new input and the hidden state of the previous inference of the cell are concatenated (Equation (3.6)) and forwarded to four convolutions (input, forget, cell and output convolutions, Equations (3.7) to (3.10)). The input, forget and output gates are calculated by summing the results of the corresponding convolutions with the Hadamard products of the cell state of the previous inference (C_{t-1}) with their corresponding parameters. The new cell state (C_t) is a combination of the previous cell state, the input and forget gate and the cell convolution. The new hidden state (H_t) is the Hadamard product of the output state with the new cell state. Equations (3.11) to (3.13) are called gates because the values they contain range from 0 to 1, and act as a means for features to prevail over other, less beneficial features when calculating the new hidden and cell states. The hidden state of a cell represents the output feature map of the cell, given the current cell state. The cell state carries information of a single cell alongside the temporal dimension (often seen as one because sequences can be easily interpreted as time series.)

The difference between ConvLSTM and common LSTM (also referred to as Fully

Connected LSTM – FC-LSTM) is the operations chosen in Equations (3.7) to (3.10); Fully Connected layers are implemented instead of convolutional layers. In fact, the operation used in these four equations greatly impacts the overall behaviour of the LSTM cells.

A NN may contain more than one LSTM cells; it is often the case that cells are stacked such that the input of one layer is the hidden state of the previous layer, while each cell updates its cell and hidden state in every recurrence [27]. Stacking cells allows for better adaptability of the model in more complex datasets. This is because, as multilayer perceptrons (MLP) have better convergence than linear models, models with multiple layers can describe more complex relationships between the features of the input data. One more way to utilize LSTM cells is by parsing the sequence in both natural and reverse order [27]. Due to the forget gate, elements at the start of a sequence have a lower influence in the result of a RNN than the elements at the end of the sequence. Therefore, parsing a sequence both forwards and backwards can greatly improve a model's response when the nature of the problem it is used for dictates that elements of the input sequence equally influence the output state. Bidirectional LSTMs are extensively used in NLP, where a sentence is parsed in both directions in order for the final state of the model not to favour any of the edges of the sentence and have a more balanced understanding of the context the text.

3.1.3 Xception-LSTM

The proposed model utilises XL [2] as a substitute for convolutions in ConvLSTM cells. The XL used in the LSTM cell is shown in Figure 3.2a.

As explained in Section 3.1.2, the operation implemented in Equations (3.7) to (3.10) greatly affects the prediction capabilities of an RNN. Combining XL and ConvLSTMs can have significant benefits; the final structure is a lightweight RNN that processes the input feature map in more than one ways in the same layer. By doing so, there is better data extraction and the information contained in the cell states is more representative of the input sequence compared to other, straightforward implementations of ConvLSTMs.

3.1.4 Sequence Models

It is often the case that data appear in sequences such as text sequences and time series. Depending on the problem, a model may be classified in one of the following categories:

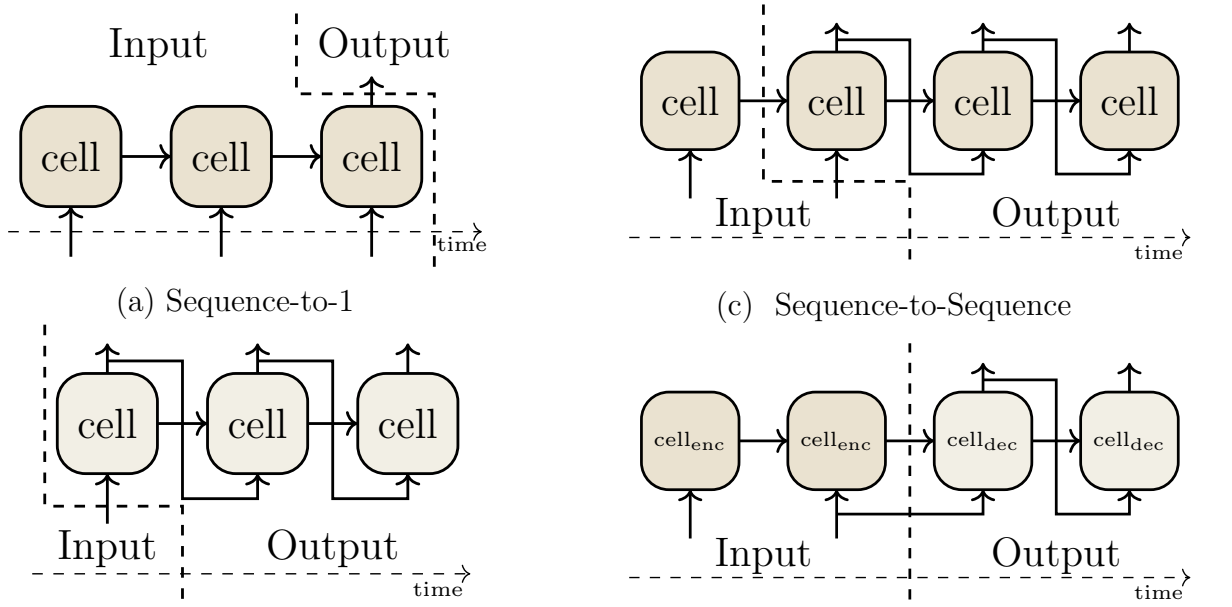
I. Sequence-to-1 (Seq2One): an input sequence produces a single output element (Figure 3.3a),

II. 1-to-Sequence (One2Seq): a single input element produces an output sequence (Figure 3.3b),

III. Sequence-to-Sequence (Seq2Seq): an input sequence produces an output sequence (Figure 3.3c), and

IV. Sequence-to-Sequence with Encoder/Decoder: an input sequence produces an output sequence using an encoder-decoder structure (Figure 3.3d).

The main difference between Figure 3.3c with Figure 3.3d is the use of 2 separate modules in the same model when working with encoders – decoders, instead of a single module switching from accepting an input sequence to generating an output sequence.



(a) Sequence-to-1

(c) Sequence-to-Sequence

(b) 1-to-Sequence

(d) Sequence-to-Sequence with Encoder/Decoder

Figure 3.3: Sequence Model Categories.

The encoder/decoder model utilizes a Sequence–to–1 module to parse the input sequence to a state (or set of states if Stacked LSTM is used) and is often referred to as an encoder. The model further utilizes an 1–to–Sequence module to generate the output sequence, based on the state produced by the encoder. The latter module is also referred to as a decoder. Its input, excluding the state initialization with the encoder's state, may be:

I. either the final element of the input series (when continuity between the input and output sequence is expected,)

II. the first element of the input sequence (eg. generating an enriched version of the input sequence,)

III. the encoder's state again (decoding an encoded state,) or

IV. a default input, independent of the input tensor, usually zero or a parameterized tensor (often implemented as such in NLP.)

In NLP, models are usually based on an encoder/decoder architecture and the input sequence is initiated with a default element, which is also used as the input element of the decoder. RNN-based translators make use of that in order to signal the start of a new sentence and generate text in another language. This signaling also appears in natural languages such as the use of ¿ and ¡ in Spanish.

As for Time-Series data, the output sequence is seen as the logical continuation of the input sequence. Since the decoder accepts the previous prediction and produces a new prediction in each step, and all output elements are timestamped, we can easily conclude that the input element of the decoder should be the start of the output sequence. Due to the expected continuity of the input and output sequence, the input element of the decoder should be no other than the end of the input sequence. The proposed model is operating on Time-Series, so the input element of the decoder is the last element of the input sequence, as shown in Figure 3.3d.

3.1.5 Spatiotemporal Encoders/Decoders

The proposed model accepts an input sequence of images and produces an output sequence of arithmetic data. The conflict between data types makes using simple Seq2Seq models complicated; a feedback module would be necessary in order to fit the RNN output to its input. That feedback module would need to create a tensor from a single scalar value. One can guess that such module would not produce satisfactory results.

Instead of using a single RNN module as a model, the proposed model consist of three discrete modules:

I. Spatial Encoder: a CNN-based module that compresses an input image to an abstract tensor that can be seen as an encoded state of the image,

II. Temporal Encoder/Decoder: a Seq2Seq RNN module based on XceptionLSTM that accepts and produces encoded states and

III. Spatial Decoder: a MLP-based module that decodes an encoded state to a scalar value prediction.

Spatial encoders and decoders transform elements of a sequence, while temporal encoders and decoders transform sequences. Temporal encoders and decoders operate on inputs and outputs of the same data types and shapes. This way, no feedback module is necessary for recurrence, thus making the module much simpler. Another advantage is that the module learns to produce tensors with the same characteristics as the ones it accepts. That means the output of the model is potentially closer to tensors that would produce realistic results. The structure of the proposed model is based on the following thought; if the spatial encoder and decoder were combined sequentially, then the input image would be transformed into a scalar value of the same timestamp. We can hypothesize that the encoded state produced by the spatial encoder represents the state for this particular timestamp. It is also given that the proposed model outputs Time-Series. Therefore, we can logically conclude that temporal decoders should only shift the encoded state by a constant time delta (also called horizon,) meaning that temporal decoders emulate the behaviour of the spatial encoders when given an image captured a horizon after the image that is currently being processed.

Since the elements of input and output sequences consist of different types of data, the proposed model utilizes a CNN based module to process input images and a MLP for

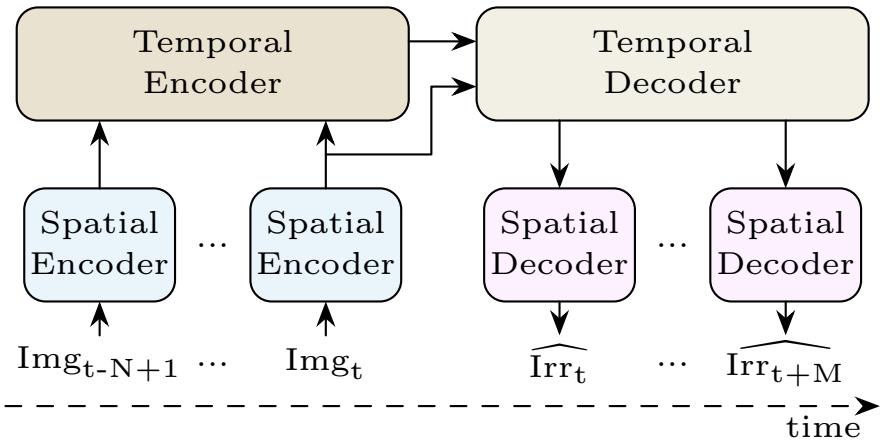


Figure 3.4: Spatiotemporal Encoder/Decoder.

generating scalar values. For the purpose of evaluating Xception-LSTMs, the encoded states are chosen to be 3-dimensional tensors. Before the encoded states are processed by the spatial decoder, the encoded state is flattened to a vector.

3.1.6 Proposed Model

The proposed model is a spatio-temporal encoder/decoder with an XL-based spatial encoder, an XceptionLSTM temporal encoder and decoder and a MLP as the spatial decoder. The structure of spatial encoder and decoder is depicted in Figures 3.5 and 3.6 respectively. The spatial encoder consists of 6 layers; Two input XLs are used to extract data from the input image. Two middle, residual layers are used for refining data. Each middle, residual layer consists of a nested sequential model of two XLs. The layer's output is the sum of the nested module's output and its input. Especially for the second residual layer, the sum is instead calculated with the result of a 1×1 convolution of the input in order to match the number of channels between the input and the desired output tensor. The nested module can be interpreted as an input corrector, thus refining the input tensor's data. Two output XLs are used to compress data to an encoded state. The encoded state is normalized before exiting the spatial encoder. The spatial decoder consists of three Linear Layers. An encoded state produced by the temporal decoder is first reduced to a fixed sized tensor with Adaptive Average Pooling and flattened to a vector. The vector is then normalized and finally forwarded to a 3-layer MLP.

All layers are followed by an activation function. In this work, LeakyReLU is used with a negative slope of 0.1125. ReLU is used as the output activation function in inference, but LeakyReLU is used during training with a near-zero negative slope (10^{-3}) to allow backpropagation when a negative value is produced in the early stages of training. The proposed model accepts any image of frame size of at least 64×64 pixels. In this work, the proposed model has been optimized for images with a frame size of 128×128, as all tested models during development showed improved results for this particular configuration.

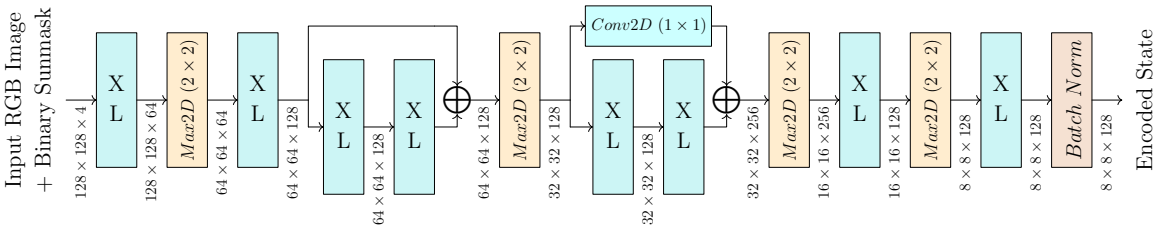


Figure 3.5: Spatial Encoder.

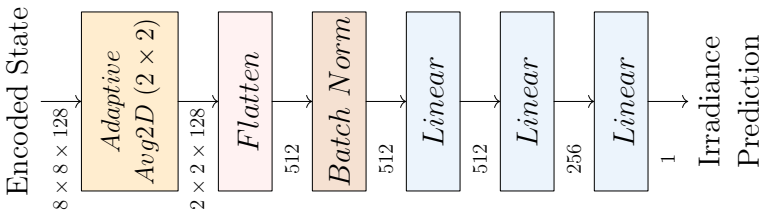


Figure 3.6: Spatial Decoder.

3.2 Dataset

The proposed model has been trained and evaluated on a dataset that, to this day, is being developed for the purposes of the Archon Project [33]. The said dataset consists of sky images with a horizon of one minute and GHI values of the athenian sky from October 25^{th} 2022 till today. Samples of the dataset are provided in Figure 3.7. We use the months January, April and July 2023 for testing and the rest of the dataset for training and validation. We chose these three months because each one represents distinct characteristics in terms of meteorological phenomena and energy yields:

- January has short days with low energy yields and frequent weather changes. It is mostly cloudy with overcast conditions.
- April has balanced day and night hours with average energy yields and frequent weather changes. It is mostly sunny with partly-cloudy conditions.
- July has long days with high energy yields and infrequent weather changes. It is mostly sunny with clear sky conditions.

3.2.1 Input Images

The proposed model accepts images as inputs. It is often the case that images contain redundancy; even with the human eye, shrinking significantly the size of an image won't affect as much our ability to recognize structures in the image. In the world of computer, an image may be shrunk down to just 25% of its original size in bytes. Observing the images of Figure 3.7, we can say that the colours in a sky image do not vary much, as blue, yellow, grey, white and a short range of their intensity usually appear. Therefore, sky images tend to have a lot of redundancy, and thus, reducing their size won't significantly affect the information we are interested in.

The sky imager – the camera that captures the Athenian sky dome for the purpose of creating the dataset – captures images of size 1536×1536 with three channels (RGB)

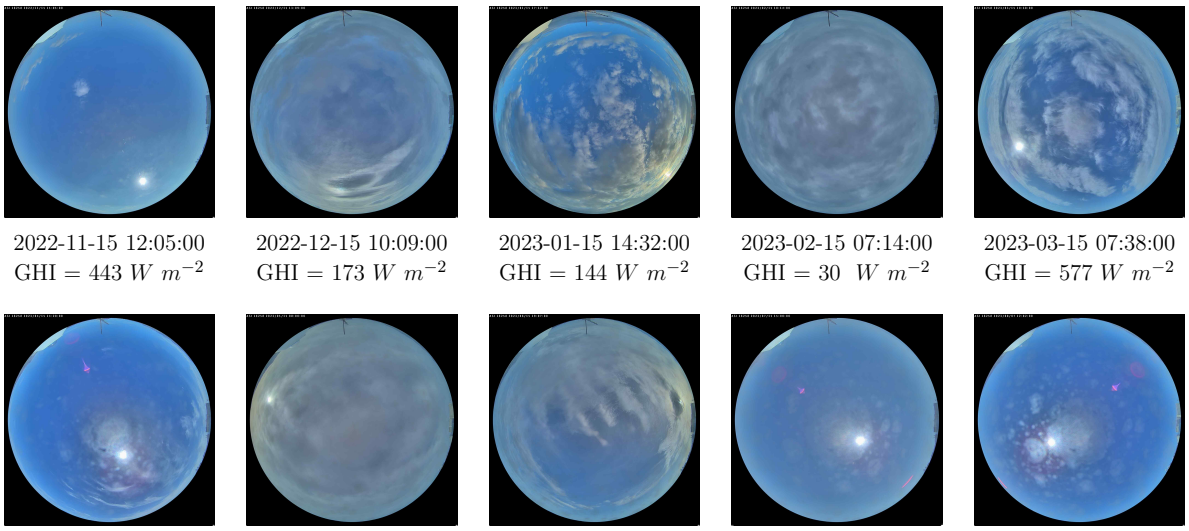


Figure 3.7: The Archon – Athens, Greece Dataset.

of 8-bit integers. The proposed model has been optimised in images of size 128×128 of single floats, which corresponds to a memory allocation for the input equal to 2.77% of the original image during inference. The vast difference in memory allocation during inference also positively affects the time performance of the model and its portability, meaning it would be easier for an edge device to infer data.

3.2.1.1 Binary Sunmask

To increase the model's performance, an extra channel is added to each image. The added frame is a binary sunmask, a mask of true or false values, later converted to single floats, that highlights the sun disk in an image in clear sky conditions. The introduction of this binary mask is one way to provide the models with useful data related to the image. Such data include the solar azimuth and the solar elevation and by using this binary mask they can be easily correlated with the image. It also hints that the highlighted areas are the region in the image expected to correspond to the sky fragments providing the larger fraction of GHI. Finding the sun's position in an image is not a trivial task; in this work, this problem is partitioned in two separate problems with known solutions:

- locating the sun in a sky dome given the time and coordinates of the observer, and
- projecting a point of a hemisphere to a flat surface.

Combining the two solutions results in the pinpointing the centre of the sun in an image, given its timestamp, the location and orientation of the camera and the intrinsic characteristics of the camera.

Locating Sun in a Sky Dome: This work utilizes astronomy equations for the calculation of the centre of the sun in a sky dome. Specifically, all necessary functions are implemented in pvlib python [34], a Python package for simulating the performance of photovoltaic energy systems. Based on the location of the camera and the timestamp, the solar elevation and solar azimuth can be calculated with the help of the pvlib package. The solar azimuth is corrected in order to account for the orientation of the camera.

Projecting a Point of a Hemisphere: This problem originates from the way the camera collects light from its environment and creates an image. The projection of a point is dependent to the intrinsic characteristics of the camera [35]. A camera can be described by a camera matrix and distortion coefficients. A calibration of the camera may be done by capturing an image of a chess board, but this is not always possible since the camera, like in this case, is in a remote place. Instead, mapping functions are defined [36] for fisheye distortion. These functions are radial and are based on usual ways fisheye lens are being implemented. The following mapping functions are implemented:

$$\begin{aligned} \textbf{Rectilinear:} &\quad r = f \tan\theta \\ \textbf{Stereographic:} &\quad r = 2f \tan {}^{\theta}\!/_{2} \\ \textbf{Equidistant:} &\quad r = f\ \theta \\ \textbf{Equisolid:} &\quad r = 2f \sin {}^{\theta}\!/_{2} \\ \textbf{Orthographic:} &\quad r = f \sin\theta \end{aligned}$$

where r is the distance of the projected point from the centre of the image, f is the focal length of the camera and θ is the solar zenith (zero is the centre of the sky dome and 90° is

the horizon.) Since distance in images is counted in pixels, the focal length is normalized and is equal to

$$f_n = \frac{1}{Radius(px)} \times \frac{focal\ length\ (mm)}{pixel\ size\ (^{mm}/_{px})}. \tag{3.16}$$

The normalized distance is then adjusted to the size of the image and transformed into cartesian coordinates on the image. The start of the cartesian axi is the pixel (0,0), which differs from the start of the polar axi ($^{R}/_{2}$,$^{R}/_{2}$). Therefore, the centre of the sun is calculated by Equation (3.17)

$$\begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} i_c \\ j_c \end{bmatrix} + r_n \; R \begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \tag{3.17}$$

which corresponds to a clockwise rotation starting from the North, scaling the vector to the size of the distance of the point from the sky dome's centre and adding an offset to the vector, or equivalently shifting the start of the axi from the centre of the dome to the first pixel of the image. For this dataset, the characteristics of the camera are unknown. The most accurate masks are produced with the equisolid mapping function and a normalized focal length of 0.695. Examples of the binary mask application in samples of the input image sequences are shown in Figure 3.8.

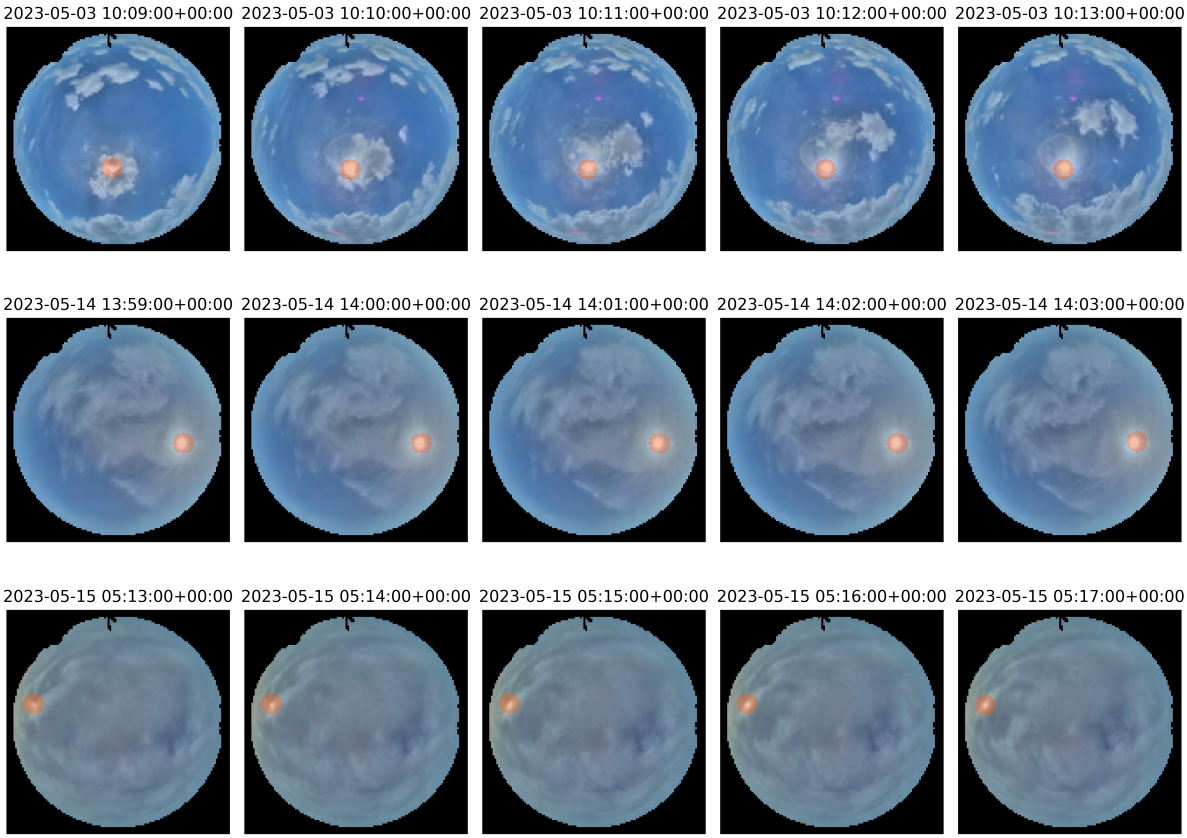


Figure 3.8: Samples of the input image sequences. The calculated sunmasks that highlight the solar area in clear sky conditions appear as orange filters.

3.2.2 Ouput Irradiace

The proposed model outputs irradiance values as single floats. The target values are integers and range from zero to around $1460\,\mathrm{W\,m^{-2}}$. The model is tasked with predicting values in the same range, which is 4 orders of degree between the lowest and highest possible value.

3.3 Metrics

Metrics are used in order to quantify the performance of a model when evaluated in a dataset rather than visually comparing the predictions and targets. The literature provides a variety of metrics for evaluating solar forecasts, which are envisaged from different perspectives [3]. In this work, we evaluate the results of the tested models with mean bias error (MBE), mean absolute error (MAE), mean absolute percentage error (MAPE), root mean square error (RMSE), and forecast skill (FS) shown in Equations (3.18) to (3.22):

$$\text{Mean bias error:} \qquad \mathrm{MBE} = \frac{1}{N}\sum_{i=1}^{N}\left(\widehat{y}_i - y_i\right), \tag{3.18}$$

$$\text{Mean absolute error:} \qquad \mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}\left\|\widehat{y}_i - y_i\right\|, \tag{3.19}$$

$$\text{Mean absolute percentage error:} \qquad \mathrm{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\left\|\frac{\widehat{y}_i - y_i}{y_i}\right\|, \tag{3.20}$$

$$\text{Root mean square error:} \qquad \mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\widehat{y}_i - y_i\right)^2}, \tag{3.21}$$

$$\text{Forecast skill:} \qquad \mathrm{FS} = 1 - \frac{\mathrm{RMSE}}{\mathrm{RMSE}_{\mathrm{pers}}}, \tag{3.22}$$

where N is the size of the test dataset, $\widehat{y}_i$ is the forecast and y_i is the target value for a horizon H. The MBE highlights whether a model shows bias when forecasting and hence, whether the results tend to consistently under- or overestimate the target value. The MAE and RMSE show the measured deviation of the results in respect to the target values. We can interpret the former as the expected deviation in the lower range of the GHI values, whereas the latter refers to the expected deviation in the upper range of the GHI values. The FS provides a more dataset-independent way to evaluate models [37]. This is accomplished by comparing the models to the Persistence Model, which forecasts that no change will occur to the target value after a horizon. The Persistence Model is a baseline model that often appears in short- and ultra short-term irradiance forecasting solutions, where the forecast horizon ranges from 15 s to 2 min. The MAPE metric indicates the normalized deviation of the forecasts from the target values, which also helps in assessing the models' performance more comprehensively.

Chapter 4

Results

This section presents the evaluation results of the proposed model for the task of short-term irradiance forecasting. Moreover, it presents the results of the study on the performance of models with various temporal encoders/decoders and it compares their results to that of the proposed model. The comparison models include ConvLSTMs, stacked ConvLSTMs, bidirectional ConvLSTMs and their respective depthwise separable (DWSConvLSTM) and Xception versions. The presented benchmark also compares the temporal encoders and decoders that are based on convolutional gated recurrent Units (ConvGRU) [38], an RNN initially intended for spatio-temporal feature learning from videos. We note here that bidirectionality only applies to the temporal encoder, as it is the only module that accepts a sequence as an input, and the results of the forward and backward pass of the input sequence are summed and forwarded to the temporal decoder. All layers of the tested temporal models accept and generate tensors of size $8 \times 8 \times 128$. Table 4.1 is an overview of the models evaluated in this work. All hyperparameters are listed in Table 4.2.

All the models are trained and evaluated in a Linux workstation with an Intel(R) Core(TM) i7-9700K CPU @ 3.60 GHz and a NVIDIA GeForce RTX 3080 GPU. We deploy a Raspberry Pi 4 Model B 8 GB and a Raspberry Pi Zero 2W for time performance tests as devices on the edge, configured as a Linux workstation with a quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5 GHz for the former and as a Linux workstation with a quad-core Arm Cortex-A53 64-bit SoC @ 1 GHz for the latter device. We use Python 3.9.13 and Pytorch 2.0.0+cuda11.7 for the development of the evaluated models.

4.1 Training Scheme

In order to reduce the total training time of all models that are evaluated in this article, we used transfer learning and partitioned the models' training in two stages, as shown in Figure 4.1. In the first stage, we trained the spatial encoder and decoder in the training dataset for the problem of irradiance estimation. Specifically, the spatial model accepts an image and estimates the GHI value for this particular image. This stage is common to all the models we tested; therefore, the spatial encoder and decoder were trained only once. We train the second stage's spatio-temporal model using as initial weights: (a) for the spatial encoder and decoder those resulting of the first stage and (b) for the temporal encoder and decoder arbitrary weights. This scheme allows us to test whether the spatial

Table 4.1: Overview of spatio-temporal models' number of parameters and operations and the training time per epoch for an input sequence of five $128 \times 128 \times 4$ images and an output sequence of fifteen irradiance values.

Temporal Model	Kernel Size	Temporal Encoder / Decoder		Spatio-Temporal Encoder / Decoder		Training Time per Epoch (min)
		Param.	OPs (MAC)	Param.	OPs (MAC)	
Spatial Encoder	-	-	-	833 K	1.03 G	3.26
Spatial Decoder	-	-	-	658 K	0.66 M	
ConvGRU	3	1.79 M	1.13 G	3.90 M	6.27 G	19.16
	5	4.93 M	3.15 G	7.05 M	8.28 G	20.54
ConvLSTM	3	2.43 M	1.51 G	4.94 M	6.65 G	19.33
	5	6.62 M	4.19 G	9.13 M	9.33 G	20.79
bi-ConvLSTM	3	4.85 M	1.89 G	6.80 M	7.02 G	20.89
	5	13.2 M	5.24 G	15.8 M	10.4 G	21.68
Stacked ConvLSTM	$3, 3$	6.06 M	3.02 G	8.57 M	8.16 G	21.67
	$3, 5$	12.3 M	5.71 G	14.9 M	10.8 G	23.54
	$5, 5$	16.5 M	8.39 G	19.1 M	13.5 G	25.45
DWSConvLSTM	3	334 K	172 M	2.85 M	5.31 G	19.66
	5	342 K	177 M	2.85 M	5.31 G	19.73
bi-DWSConvLSTM	3	668 K	215 M	3.18 M	5.35 G	20.13
	5	684 K	221 M	3.20 M	5.36 G	20.23
Stacked DWSConvLSTM	$3, 3$	826 K	343 M	3.34 M	5.48 G	20.64
	$3, 5$	839 K	349 M	3.35 M	5.49 G	20.75
	$5, 5$	847 K	354 M	3.36 M	5.49 G	20.87
XceptionLSTM	XL	871 K	516 M	3.38 M	5.65 G	19.63
bi-XceptionLSTM	XL	1.74 M	645 M	4.25 M	5.78 G	19.75
Stacked XceptionLSTM	2×XL	2.17 M	1.03 G	4.68 M	6.17 G	20.79

encoder and decoder can effectively forecast GHI values. In order to reduce the total training time of all models that are evaluated in this article, we used transfer learning and partitioned the models' training in two stages, as shown in Figure 4.1. In the first stage, we trained the spatial encoder and decoder in the training dataset for the problem of irradiance estimation. Specifically, the spatial model accepts an image and estimates the GHI value for this particular image. This stage is common to all the models we tested, therefore the spatial encoder and decoder were trained only once. We train the second stage's spatio-temporal model using as initial weights: a) for the spatial encoder and decoder those resulting of the first stage and b) for the temporal encoder and decoder arbitrary weights. This scheme allows us to test whether the spatial encoder and decoder can effectively forecast GHI values.

For the first stage, we used RMSProp with decay of 0.9 and $\epsilon = 1.0$. We used a learning rate of 0.001 decaying every epoch using an exponential rate of 0.94 and the MSELoss as

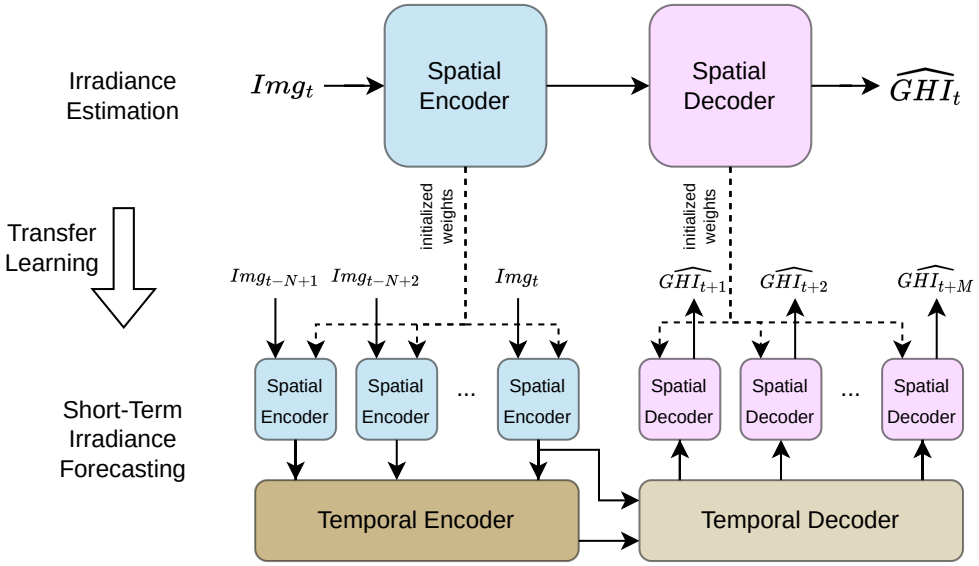


Figure 4.1: Training scheme.

Table 4.2: The hyperparameters that were examined and chosen for the benchmark.

Hyperparameter	Tested Options	Final
Input Sequence Length	{5, 10, 15}	5
Output Sequence Length	{5, 10, 15}	15
Image Frame Size	{64, 128, 256}	128
Concatenate Sunmask	{True, False}	True
Removed Foreign Objects	{True, False}	True
Encoded State's Channels	{16, 32, 64, 128, 256}	128
Optimizer	{Adam, RMSProp}	RMSProp
Scheduler	{ReduceOnPlateau, Exponential, Step}	Exponential
Learning Rate	$\{5 \cdot 10^{-2},\ 10^{-3},\ 5 \cdot 10^{-4},\ 10^{-4},\ 5 \cdot 10^{-5}, 10^{-5}\}$	1st Stage : 10^{-3} 2nd Stage : $5 \cdot 10^{-5}$
Loss Function	{L1, SmoothL1 Huber, MSE}	MSE Loss
Batch Size	{8, 12, 16, 20, 24}	16

the criterion for calculating the loss. The spatial model was trained for 29 epochs and achieved a RMSE of $35.3\,\mathrm{W\,m^{-2}}$ for the solar irradiance estimation. For the second stage, we used the same scheduler, optimizer and loss function with an initial learning rate of $5 \cdot 10^{-5}$. With this training scheme we were able to reduce the total epochs from 15-30 to just 3-6 epochs for the spatio-temporal models. Moreover, all the models achieved better metrics when they were trained with this scheme compared to training the whole spatio-temporal model without any transfer learning.

Table 4.3: Evaluation results of the models in Table 4.1 for the horizons of 1, 5, 15 min and the average for the first 15 min. The best models for each metric and horizon are **bolded**.

Model	Kernel Size	MBE ($W\,m^{-2}$)				MAE ($W\,m^{-2}$)				MAPE(%)	
		1 min	5 min	15 min	Mean	1 min	5 min	15 min	Mean	1 min	5 min
Persistence	–	**¡0.025**	**-0.0841**	0.1795	**¡0.001**	**20.61**	**44.1**	74.8	52.9	**6.21**	**15.89**
ConvGRU	3	-3.19	9.63	9.09	8.83	34.6	47.2	59.6	50.4	24.66	32.5
	5	-1.508	7.86	8.17	7.52	33.0	46.2	59.0	49.7	22.72	29.34
ConvLSTM	3	-0.686	5.68	3.82	3.60	33.1	47.0	59.6	49.9	21.41	27.73
	5	2.080	1.898	-1.321	0.1465	33.0	47.1	60.9	50.8	21.73	27.87
bi-ConvLSTM	3	3.02	2.520	**¡0.025**	0.1741	34.1	49.2	63.3	53.5	19.89	26.69
	5	-4.24	-5.07	-7.30	-6.10	34.3	47.4	60.0	50.8	19.91	24.87
Stacked ConvLSTM	3, 3	3.10	3.26	1.304	1.741	33.5	47.3	59.5	50.3	22.08	29.31
	3, 5	0.2465	0.313	-4.12	-1.500	33.0	47.0	60.0	50.4	21.49	27.68
	5, 5	-2.170	-1.474	-4.08	-2.824	35.0	47.7	59.9	51.0	19.76	26.35
DWSConvLSTM	3	-14.13	-5.37	-2.234	-4.49	35.5	46.6	59.8	50.3	22.11	27.69
	5	-11.49	-5.97	-3.01	-4.75	34.8	46.3	59.0	49.8	21.56	26.20
bi-DWSConvLSTM	3	-15.90	-5.63	0.988	-3.94	37.0	46.0	58.5	49.5	22.71	26.88
	5	3.54	3.81	-4.48	1.335	34.5	46.6	60.1	50.1	23.07	28.81
Stacked DWSConvLSTM	3, 3	-6.38	0.678	2.481	1.022	35.0	46.4	**58.5**	49.6	23.14	29.92
	3, 5	-8.51	-3.71	1.160	-1.831	34.7	45.4	58.3	**49.0**	20.60	26.18
	5, 5	1.994	0.452	-6.07	-1.518	34.5	46.7	59.6	50.1	23.01	28.48
XceptionLSTM	XL	-3.01	1.139	-4.87	-0.924	32.5	46.1	59.9	49.7	20.46	24.43
bi-XceptionLSTM	XL	-7.03	-2.992	-8.16	-4.52	32.7	45.7	59.0	49.2	19.57	22.98
Stacked XceptionLSTM	2×XL	-2.650	-0.963	-3.35	-1.657	33.0	45.8	60.4	49.9	19.38	23.27

		RMSE ($W\,m^{-2}$)				FS (%)				MAPE (%)	
		1 min	5 min	15 min	Mean	1 min	5 min	15 min	Mean	15 min	Mean
Persistence	–	75.2	113.3	146.6	122.4	–	–	–	–	38.5	30.7
ConvGRU	3	69.4	94.8	113.1	99.9	7.70	16.37	22.88	17.77	38.5	30.7
	5	68.3	95.4	114.6	100.0	9.12	15.77	21.83	17.74	36.8	31.4
ConvLSTM	3	69.5	95.5	114.5	100.0	7.50	15.69	21.92	17.67	38.5	30.7
	5	70.0	98.2	116.2	102.1	6.94	13.34	20.75	15.93	37.2	30.5
bi-ConvLSTM	3	70.2	96.5	115.5	101.3	6.61	14.85	21.22	16.58	33.2	28.38
	5	70.1	95.9	115.6	100.9	6.79	15.36	21.13	16.94	31.1	26.53
Stacked ConvLSTM	3, 3	69.7	96.3	115.1	100.7	7.27	15.06	21.47	17.09	36.7	31.0
	3, 5	70.2	97.8	116.5	102.2	6.67	13.66	20.52	15.87	34.3	29.43
	5, 5	70.7	96.5	115.0	101.1	5.93	14.88	21.56	16.72	31.9	27.64
DWSConvLSTM	3	72.2	96.7	117.0	101.7	3.96	14.65	20.17	16.15	37.6	30.5
	5	71.4	95.2	115.2	100.3	5.02	15.97	21.39	17.31	34.9	28.99
bi-DWSConvLSTM	3	72.3	95.0	115.1	100.0	3.79	16.14	21.50	17.51	35.7	29.55
	5	71.0	95.7	114.8	100.2	5.57	15.57	21.71	17.37	38.8	31.6
Stacked DWSConvLSTM	3, 3	73.3	95.9	**113.7**	100.3	2.51	15.38	**22.43**	17.16	36.7	31.8
	3, 5	72.1	95.2	114.3	100.0	4.09	16.01	22.01	17.55	35.2	28.72
	5, 5	71.1	95.5	114.4	100.2	5.41	15.71	21.99	17.42	39.4	31.4
XceptionLSTM	XL	68.8	94.6	116.0	**99.8**	8.53	16.52	20.88	**17.85**	32.3	27.09
bi-XceptionLSTM	XL	**68.7**	94.5	117.3	100.3	**8.52**	16.57	20.01	17.52	**28.91**	**24.50**
Stacked XceptionLSTM	2×XL	69.2	**94.4**	115.8	99.8	7.94	**16.69**	21.04	17.85	31.3	25.66

4.2 Model Evaluation

Table 4.1 presents the implementation details of the models evaluated in this work: the ConvLSTM, the DWSConvLSTM and the XceptionLSTM-based models in their single-cell, bidirectional and double-stacked versions and the ConvGRU-based models. Table 4.3 presents the evaluation results of the metrics for the tested models. All the models considered for this comparison infer a sequence of five images with a horizon of 1 min and output 15 GHI values that correspond from 1 to 15-min forecasts. Note here that the XceptionLSTM cell in the three versions we examined prevails with respect to the RMSE and FS scores. Specifically, the single XceptionLSTM cell scores 68.8 W m^{-2}, 94.6 W m^{-2} and 116 W m^{-2} RMSE for the horizons of 1, 5, 15 min with a mean score of 99.8 W m^{-2} and the double-stacked XceptionLSTM scores 69.2 W m^{-2}, 94.4 W m^{-2} and 115.8 W m^{-2} RMSE with a mean score of 99.8 W m^{-2}. Moreover, the proposed cell in its bidirectional form scores 68.7 W m^{-2}, 94.5 W m^{-2} and 117.3 W m^{-2} in the RMSE metric for the same horizons and reports a mean score of 100.3 W m^{-2}. The bidirectional XceptionLSTM is the best-performing model based on the MAPE metric, achieving a mean score of 24.5% across the examined horizons. Furthermore, the XceptionLSTM cell achieves low MAE scores for the 1 min horizon, but the stacked DWSConvLSTM cells report lower MAE metrics in all the other horizons.

The MBE metric reports no noticable bias for ConvLSTM and XceptionLSTM-based spatio-temporal models, which means that these two kinds of models have no tendency to over- or underestimate forecasts. On the other hand, most DWSConvLSTM-based models tend to systematically underestimate the forecasts of the first few horizons. The same tendency appears in the bidirectional versions of all models. In contrast to the above, the ConvGRU based models tend to overestimate forecasts. Moreover, the models with bidirectional temporal encoders appear to forecast more accurately for greater horizons when compared to their unidirectional counterparts, but less accurately when one more layer is added to the unidirectional models. As we can conclude by the metrics of Table 4.3, the increased accuracy of the models with stacked temporal encoders and decoders seems to be more intense in the cells that use depthwise operations. In addition, given that all the benchmarked models score RMSE and MAE values within 4.6 W m^{-2} from one another, we conclude that the structure of the temporal encoder and decoder does not significantly change the reported metric scores.

4.3 Timing Reports

Figure 4.2 is a diagram of the mean execution time of the evaluated models on the edge devices. The proposed model, in its three benchmarked versions, executes as a single cell in 2.69 s and 7.54 s, as a bidirectional cell in 2.91 s and 7.99 s and as a double-stacked cell 3.31 s and 8.78 s for the Raspberry Pi 4 Model B and the Raspberry Pi Zero 2W. We notice that the inference time of all models are within 10 % of the slowest recorded time on both edge devices, which corresponds to the model that includes stacked XceptionLSTMs in its temporal encoder and decoder. This is because a major fraction of the complexity derives from the repeated execution of the spatial encoder and decoder, that is, once per element of the input and output sequences. The graph also shows the expected behavior for the models that are based on the same LSTM cells, a fact deducing that the model

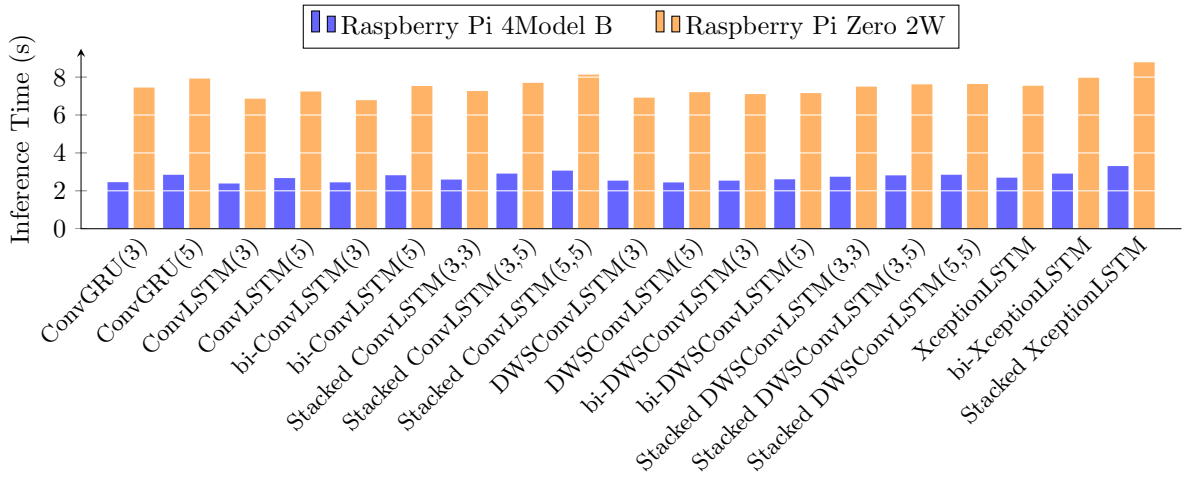


Figure 4.2: Timing reports for inference on low-cost, edge computing devices.

with less parameters executes faster. That behaviour is not true for cells of different structure; despite the great difference in the amount of parameters and total operations, XceptionLSTM cells appear to have comparable execution times with ConvLSTM cells due to the optimizations that the Pytorch library performs in convolutions. We believe that the reason behind this is that the operation of concatenating the channels of the results of all the nested depthwise operations in an XL to form a single tensor for the pointwise convolution to process causes the reported execution time overhead. One way to cope with this is to avoid concatenation by executing the pointwise operation first, then splitting the intermediate tensor and finally, forwarding the chunks to the nested depthwise operations. These modifications improve the reported execution times but they result in degraded metrics. During the evaluation of the inference times, both devices reported a constant power consumption measured at 5.1 W for the Raspberry Pi 4 Model B and 0.7 W for the Raspberry Pi Zero 2W.

Chapter 5

Conclusions

The proposed model targets implementations for short-term irradiance forecasting on low-power devices. In this article, we presented a benchmarking of known ConvLSTM based spatio-temporal models [26, 10, 1] for the latter task with the evaluation of the models in terms of metric scores and execution times. The proposed XceptionLSTM cell and spatio-temporal model show notable performance for horizons over 10 minutes and improved forecasting skills for smaller horizons. We noticed that, when taking into account the evaluation results of other related works [13, 39], the proposed model exhibit a lower drop of forecasting skill as the horizon increases. This means that our model constitutes a very attractive solution for short-term irradiance forecasting; moreover it can be integrated in SG systems that are based on ultra short-term forecasts. Furthermore, the proposed RNN is significantly lightweight when compared to traditional RNNs and models from the bibliography [13, 14], as it requires half of the memory that the weights of the ConvLSTM-based spatio-temporal models need.

Focusing on the edge devices, the proposed model is optimized for inference on low-power devices and can process up to 22.27 sequences per minute in the low power device Raspberry Pi 4 Model B and up to 7.81 sequences per minute in the ultra low-power device Raspberry Pi Zero 2W. Hence, less powerful IoT devices suffice for the execution of forecasting tasks that were once considered power intensive and computationally demanding. Considering that the number of PV parks added to the power grids constantly increases and that low-end devices are easy to maintain and cost-effective, the proposed model offers a very tempting alternative to the high performance and high cost controllers.

Abbreviations

The following abbreviations are used in this thesis:

AI	Artificial intelligence
ANN	Artificial neural network
CC	Cloud cover
CL	Convolutional layer
CNN	Convolutional neural network
ConvLSTM	Convolutional long short-term memory
ConvGRU	Convolutional gated recurrent unit
CPU	Central processing unit
CSI	Clear sky index
DBN	Deep belief network
DL	Deep learning
DHI	Diffuse horizontal irradiance
DNI	Direct normal irradiance
DRL	Deep reinforcement learning
DWSC	Depthwise separable convolution
DWSConvLSTM	Depthwise separable convolutional long short-term memory
FC-LSTM	Fully connected long short-term memory
FPGA	Field programmable gate array
FS	Forecast skill
GHI	Global horizontal irradiance
GPU	Graphics processing unit
GRU	Gated recurrent unit
IFS	Irradiance forecasting system
IoT	Internet of Things
LeakyReLU	Leaky rectified linear unit
LSTM	Long short-term memory
MAE	Mean absolute error
MBE	Mean bias error
ML	Machine learning
MLP	Multilayer perceptron
NLP	Natural language processing
NN	Neural network
PV	Photovoltaic
ReLU	Rectified linear unit

RES	Renewable energy source
RGB	Red green blue
RMSE	Root mean square error
RNN	Recurrent neural network
Seq2Seq	Sequence-to-sequence
SG	Smart grid
SoC	System-on-chip
VLSI	Very large-scale integration
XceptionLSTM	Xception long short-term memory
XL	Xception layer

Bibliography

[1] S. Ziyabari, L. Du, and S. K. Biswas, "Multibranch attentive gated resnet for short-term spatio-temporal solar irradiance forecasting," *IEEE Transactions on Industry Applications*, vol. 58, no. 1, pp. 28–38, 2022.

[2] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 1800–1807, IEEE Computer Society, 2017.

[3] Y. Chu, M. Li, C. F. Coimbra, D. Feng, and H. Wang, "Intra-hour irradiance forecasting techniques for solar power integration: A review," *iScience*, vol. 24, no. 10, p. 103136, 2021.

[4] S. D. Miller, M. A. Rogers, J. M. Haynes, M. Sengupta, and A. K. Heidinger, "Short-term solar irradiance forecasting via satellite/model coupling," *Solar Energy*, vol. 168, pp. 102–117, 2018. Advances in Solar Resource Assessment and Forecasting.

[5] A. Ayet and P. Tandeo, "Nowcasting solar irradiance using an analog method and geostationary satellite images," *Solar Energy*, vol. 164, pp. 301–315, 2018.

[6] L. E. Ordoñez Palacios, V. Bucheli Guerrero, and H. Ordoñez, "Machine learning for solar resource assessment using satellite images," *Energies*, vol. 15, no. 11, 2022.

[7] O. Boussif, G. Boukachab, D. Assouline, S. Massaroli, T. Yuan, L. Benabbou, and Y. Bengio, "What if we enrich day-ahead solar irradiance time series forecasting with spatio-temporal context?," 2023.

[8] Y. Nie, X. Li, Q. Paletta, M. Aragon, A. Scott, and A. Brandt, "Open-source ground-based sky image datasets for very short-term solar forecasting, cloud analysis and modeling: A comprehensive survey," *CoRR*, vol. abs/2211.14709, 2022.

[9] B. Juncklaus Martins, A. Cerentini, S. L. Mantelli, T. Z. Loureiro Chaves, N. Moreira Branco, A. von Wangenheim, R. Rüther, and J. Marian Arrais, "Systematic review of nowcasting approaches for solar energy production based upon ground-based cloud imaging," *Solar Energy Advances*, vol. 2, p. 100019, 2022.

[10] P. Kumari and D. Toshniwal, "Deep learning models for solar irradiance forecasting: A comprehensive review," *Journal of Cleaner Production*, vol. 318, p. 128566, 2021.

[11] D. S. Kumar, G. M. Yagli, M. Kashyap, and D. Srinivasan, "Solar irradiance resource and forecasting: a comprehensive review," *IET Renewable Power Generation*, vol. 14, no. 10, pp. 1641–1656, 2020.

[12] R. A. Rajagukguk, R. Kamil, and H.-J. Lee, "A deep learning model to forecast solar irradiance using a sky camera," *Applied Sciences*, vol. 11, no. 11, 2021.

[13] Q. Paletta, A. Hu, G. Arbod, and J. Lasenby, "ECLIPSE : Envisioning cloud induced perturbations in solar energy," *CoRR*, vol. abs/2104.12419, 2021.

[14] Y. Nie, Q. Paletta, A. Scott, L. M. Pomares, G. Arbod, S. Sgouridis, J. Lasenby, and A. Brandt, "Sky-image-based solar forecasting using deep learning with multi-location data: training models locally, globally or via transfer learning?," *CoRR*, vol. abs/2211.02108, 2022.

[15] S. Park, Y. Kim, N. J. Ferrier, S. M. Collis, R. Sankaran, and P. H. Beckman, "Prediction of solar irradiance and photovoltaic solar energy product based on cloud coverage estimation using machine learning methods," *Atmosphere (Basel)*, vol. 12, 3 2021.

[16] P. Manandhar, M. Temimi, and Z. Aung, "Short-term solar radiation forecast using total sky imager via transfer learning," *Energy Reports*, vol. 9, pp. 819–828, 2023. 2022 9th International Conference on Power and Energy Systems Engineering.

[17] J. Zhang, R. Verschae, S. Nobuhara, and J.-F. Lalonde, "Deep photovoltaic nowcasting," *Solar Energy*, vol. 176, pp. 267–276, 2018.

[18] Y. Sun, V. Venugopal, and A. R. Brandt, "Short-term solar power forecast with deep learning: Exploring optimal input and output configuration," *Solar Energy*, vol. 188, pp. 730–741, 2019.

[19] M. Ajith and M. Martínez-Ramón, "Deep learning based solar radiation micro forecast by fusion of infrared cloud images and radiation data," *Applied Energy*, vol. 294, p. 117014, 2021.

[20] O. Bamisile, D. Cai, A. Oluwasanmi, C. Ejiyi, C. C. Ukwuoma, O. Ojo, M. Mukhtar, and Q. Huang, "Comprehensive assessment, review, and comparison of ai models for solar irradiance prediction based on different time/estimation intervals," *Scientific Reports*, vol. 12, no. 1, p. 9644, 2022.

[21] C. Lyu, S. Eftekharnejad, S. Basumallik, and C. Xu, "Dynamic feature selection for solar irradiance forecasting based on deep reinforcement learning," *IEEE Transactions on Industry Applications*, vol. 59, pp. 533–543, Jan 2023.

[22] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 2818–2826, IEEE Computer Society, 2016.

[23] L. Kaiser, A. N. Gomez, and F. Chollet, "Depthwise Separable Convolutions for Neural Machine Translation," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, OpenReview.net, 2018.

[24] S. Basodi, C. Ji, H. Zhang, and Y. Pan, "Gradient amplification: An efficient way to train deep neural networks," *Big Data Mining and Analytics*, vol. 3, no. 3, pp. 196–207, 2020.

[25] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA* (S. Singh and S. Markovitch, eds.), pp. 4278–4284, AAAI Press, 2017.

[26] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 802–810, 2015.

[27] R. C. Staudemeyer and E. R. Morris, "Understanding LSTM - a tutorial into long short-term memory recurrent neural networks," *CoRR*, vol. abs/1909.09586, 2019.

[28] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing [review article]," *IEEE Comput. Intell. Mag.*, vol. 13, no. 3, pp. 55–75, 2018.

[29] X. Zhang, X. Liang, A. Zhiyuli, S. Zhang, R. Xu, and B. Wu, "At-lstm: An attention-based lstm model for financial time series prediction," *IOP Conference Series: Materials Science and Engineering*, vol. 569, p. 052037, jul 2019.

[30] K. K. A. Ghany, H. M. Zawbaa, and H. M. Sabri, "Covid-19 prediction using lstm algorithm: Gcc case study," *Informatics in Medicine Unlocked*, vol. 23, p. 100566, 2021.

[31] Y. Zhou, H. Dong, and A. El-Saddik, "Deep learning in next-frame prediction: A benchmark review," *IEEE Access*, vol. 8, pp. 69273–69283, 2020.

[32] C.-W. Lin, M. Lin, and S. Yang, "SOPNet Method for the Fine-Grained Measurement and Prediction of Precipitation Intensity Using Outdoor Surveillance Cameras," *IEEE Access*, vol. 8, pp. 188813–188824, 2020.

[33] "Archon Project." Accessed : 29/08/2023.

[34] W. F. Holmgren, C. W. Hansen, and M. A. Mikofski, "pvlib python: a python package for modeling solar energy systems," *Journal of Open Source Software*, vol. 3, no. 29, p. 884, 2018.

[35] "Camera Calibration." Accessed : 29/08/2023.

[36] F. Bettonvil, "Fisheye lenses," *WGN, Journal of the International Meteor Organization*, vol. 33, pp. 9–14, Feb. 2005.

[37] R. Marquez and C. F. M. Coimbra, "Proposed Metric for Evaluation of Solar Forecasting Models," *Journal of Solar Energy Engineering*, vol. 135, p. 011016, 10 2012.

[38] N. Ballas, L. Yao, C. Pal, and A. C. Courville, "Delving deeper into convolutional networks for learning video representations," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2016.

[39] M. Choi, B. Rachunok, and R. Nateghi, "Short-term solar irradiance forecasting using convolutional neural networks and cloud imagery," *Environmental Research Letters*, vol. 16, p. 044045, apr 2021.