# Time Series Forecasting using Sequence Models with Attention

by

Elizaveta Kharlova

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Software Engineering and Intelligent Systems

Department of Electrical and Computer Engineering
University of Alberta

# Abstract

Due to the growing penetration of renewable energy sources, accurate energy forecasts are required to support their effective integration. In this field, deep learning methods are currently demonstrating successful results, but there is still a room for improvement that may eventually lead to their wide deployment. We suggest that sequence processing applications from the field of natural language processing (NLP) can be adopted to forecasting. Some recent advances in deep learning that brought NLP to near human performance include sequence models and an attention mechanism. Considering that these methods apply well to univariate time series such as language, they can be potentially extended to multivariate time series forecasting. This may be applicable, for instance, for prediction of photovoltaic (PV) power generation with added weather features it depends on.

Current approaches popular in PV forecasting include statistical and machine learning methods as well as basic deep learning models such as feedforward neural networks and recurrent long short-term memory based architectures. Although deep learning has claimed some success in this field, there has been no wide adoption of sequence to sequence models yet. This insight inspired an exploration of adapting some NLP techniques to multivariate time series forecasting. We propose a sequence to sequence architecture with attention as a model superior with respect to the baseline architectures. Overall, this thesis recommends an adoption of sequence attention models in PV generation forecasting and validates this proposal by improving the quality of the forecasts.

The model leverages the high resolution multivariate signal by extracting features from the numerical weather predictions and historical information to produce a binned probabilistic forecast. Sequence to sequence models benefit from a more expressive probabilistic forecast due to their recursive structure. The attention mechanism further aids context extraction. The proposed sequence to sequence model with attention outperforms common models, such as long short-term memory networks and a classic sequence attention model, for photovoltaic generation forecasting. $k$-fold cross-validation provides additional insights on the influence of a dataset's arrangement on the model's performance and confirms the validity of the design decisions and architecture choices of the proposed model.

# Preface

Some of the work in this thesis is a result of a collaboration with Daniel May at the University of Alberta, described in article E. Kharlova, D. May, and P. Musilek, "Forecasting photovoltaic power production using a deep learning sequence to sequence model with attention," published in the proceedings of *2020 International Joint Conference on Neural Networks (IJCNN)*. Section 2.2.4, Losses and metrics, in Chapter 2 is adapted from the publication. Parts of Chapter 3 regarding the architecture description and probabilistic targets are also based on the respective section in the article. The experimental setup for benchmark models, section 4.2, is adapted in Chapter 4, as well as section 5.1 of Chapter 5 discussing this experiment. The rest of this thesis is the sole original work of the author.

# Acknowledgements

I would like to thank my supervisor, Professor Petr Musilek, for his guidance in my work as well as his advice in exploring my interests in the field of machine learning. I would also like to thank Daniel May, my colleague and mentor, for patiently supporting me in my pursuits of deep learning knowledge. I also want to thank Electrical and Computer Engineering Department and the Faculty of Engineering at the University of Alberta.

# Table of Contents

# List of Tables

# List of Figures

# List of Symbols

| | |
|---|---|
| $E$ | expected value |
| $F$ | forecast behaviour |
| $K$ | attention key |
| $KL$ | Kullback-Leibler (divergence) |
| $P$ | true behaviour |
| $Q$ | attention query |
| $R_t$ | remaining components of a time series |
| $S$ | skill score |
| $S_t$ | seasonality |
| $T$ | time - forecast horizon |
| $T_t$ | trend |
| $V$ | attention value |
| $\alpha$ | attention alighment scores |
| $\boldsymbol{H}$ | minibatch of activations |
| $\boldsymbol{\theta}$ | network parameters |
| $\boldsymbol{g}$ | gradient |
| $\boldsymbol{v}$ | momentum |

|  |  |
|---|---|
| $\boldsymbol{x}$ | network input |
| $a$ | attention alignment function |
| $c$ | attention context |
| $h$ | hidden state (of a recurrent network) |
| $s$ | hidden state of a decoder network |
| $y$ | network output |

# Abbreviations

| | |
|---|---|
| **ACSWDNB** | - accumulated downwelling shortwave flux at the bottom. |
| **ANN/NN** | - artificial neural network. |
| **ARIMA** | - autoregressive integrated moving average. |
| **CDF** | - cumulative density function. |
| **CRPS** | - continuous ranked probability score. |
| **DL** | - deep learning. |
| **E-D** | - encoder-decoder. |
| **FFNN** | - feedforward neural network. |
| **GRU** | - gated recurrent unit. |
| **HRRR** | - high resolution rapid refresh. |
| **LSTM** | - long short-term memory. |
| **LTF** | - long-term forecast. |
| **MSE** | - mean squared error. |
| **MTF** | - medium term forecast. |

**NLP** - natural language processing.

**nME** - normalized mean error.

**nRMSE** - normalized root mean squared error.

**NWP** - numerical weather predictions.

**PDF** - probability density function.

**PV** - photovoltaic (power).

**RNN** - recurrent neural network.

**S2S** - sequence to sequence.

**STF** - short-term forecast.

**SVM** - support-vector machine.

**SW** - sliding window.

**TF** - teacher forcing.

**VSTF** - very short-term forecast.

# Chapter 1

# Introduction

## 1.1 Motivation

With the growing use of renewable energy sources and the need for its effective integration, the demand for accurate energy forecasts has increased. A requirement for higher accuracy is also accompanied by a demand for more information, such as information supplied by interval or probabilistic forecasts. Deep learning has demonstrated potential for energy forecasting, but it is still underdeveloped in this area [1].

Natural language processing (NLP) has recently achieved near human performance on language tasks. The performance of language models has dramatically improved using novel deep learning approaches. The corresponding techniques were initially based on a sequence to sequence model, which were designed to reflect the purpose of language transformations [2]. Sequence to sequence modelling allows mapping sequences of different lengths, which are connected by non-linear relationships [2]. Furthermore, adding the attention mechanism improved the capacity of the sequence to sequence models to carry information [3].

Language processing applications include translation, text to speech, text generation, question answering, and other univariate sequence processing tasks. NLP models claim positive results at these applications, but have not yet been applied to multivariate time series. Considering the fact that deep learning methods are already claiming state of the art results in forecasting some multivariate time series, aforementioned NLP techniques can potentially bring additional improvements to this area. Therefore, the aim of this research is to extend the use of methods adopted in NLP to multivariate time series forecasting.

The question is whether the existing NLP architectures can be directly applied to forecasting multivariate time series. Solar power forecasting is one example of multivariate time series forecasting, because solar power generation is dependent on weather variables. Together they can be considered a multivariate time series. From the lack of wide adoption of NLP methods in multivariate time series processing [4, 5], it can be concluded that the transfer from univariate to multivariate signals may require substantial changes. The way the model is expected to utilize the data is different for multivariate time series. Having multiple variables as inputs leads to larger amount of information. So, the architectural changes will need to reflect the properties of such signals. Hypothetically, better connectivity between encoder and decoder may aid the information flow in the network. The goal is then to explore and evaluate the architectures of sequence to sequence models in multivariate time series forecasting applications.

## 1.2 Thesis Objectives

The goal of this work is to explore the application of existing univariate time series techniques to multivariate time series and to subsequently find a way of adapting these techniques to benefit multivariate time series forecasting. Several deep learning methods with origins in natural language processing are set to be explored.

1. The first objective is to analyze the sequence to sequence models and their application in forecasting.

2. Next, attention mechanism is introduced to the forecasting model and its benefits to multivariate time series forecasting are examined.
3. Afterward, possible changes surrounding the analyzed model are explored. Design choices that may affect model performance are to be identified and their effects investigated.
4. Finally, further validation of proposed model is to be conducted and opportunities for reducing required computing resources examined.

## 1.3 Thesis Outline

The structure of this thesis is as follows. Chapter 2 begins with an overview of time series forecasting methods related to this research. It is followed by a brief introduction to deep learning and its fundamental techniques. Next, a closer look at the deep learning mechanisms in the field of natural language processing, which inspired this research, is presented. An argumentation for proposed architecture of a time series forecasting model is presented in Chapter 3. Chapter 4 describes the experimental setup. Chapter 5 consists of an overview of the experimental results and their discussion. Conclusions and possible directions for future work are presented in Chapter 6.

# Chapter 2

# Background

## 2.1 Time Series Forecasting

Time series is defined as a stochastic process, a collection of random variables ordered in time. A time series usually has three components: trend, seasonality, and cyclic behaviour. Trend, $T_t$, is a long-term increase or decrease in the process value over time. Seasonality, $S_t$, is a pattern in time series data that occurs with constant and known frequency, such as time of year or day of week. The remaining component, containing cyclic patterns, is referred to as $R_t$ [6]. A cycle is a pattern occurring with a dynamic frequency, such as economic changes. It can be independent or have short-term correlation.

### 2.1.1 History of Time Series Forecasting

Time series forecasting has been dominated for a long time by statistical and mathematical model-based methods such as exponential smoothing, autoregressive integrated moving average (ARIMA), and support-vector machines (SVM). Some of these methods are still popular today [6]. Artificial neural networks (ANNs) were underrepresented, partly due to the lack of a sufficient amount of data [7]. Although networks designed for temporal processing such as recurrent neural networks (RNNs) have been introduced [8], other algorithms such as SVM and random forests have gained advantage in time series forecasting due to their easier implementation and training procedures. However, ANNs started to gain more appreciation following the increase in the availability of computational resources. The possibility to train deep neural networks resulted in new opportunities to explore deep architectures more and to use them in a wider variety of applications [4].

Time series forecasting can be divided into subfields such as energy forecasting, demand forecasting, financial forecasting etc., with each type of data exhibiting different properties. For example, energy forecasting field includes forecasts of load demand and supply, and photovoltaic power. These variables usually depend on several other features, such as temperature, and are considered smooth. Retail demand forecasting data is generally intermittent and sparse. Financial forecasting data is smooth, but depends on a large amount of uncertain variables. The amount and quality of data are also important in the choice of model, because neural networks usually perform better when trained on datasets with large amount of information, which represents the modeled process well [4].

## 2.2 Photovoltaic Power and Load Forecasting

With the rise of use of renewable energy sources, the demand for accurate forecasts of solar power generation and load consumption has increased. Photovoltaic (PV) or solar power is generated by solar panels and depends on the amount of solar irradiation [9]. Electric load is the electric power consumption in either a single household or aggregated for a certain community [5]. Both PV generation and load consumption forecasting have economic benefits in the integration of renewable resources.

This section will cover the framework of PV and load forecasting, including possible addition of exogenous variables, choices of aggregation level and horizon, as well as common metrics. Finally, current trends in PV and load forecasting are discussed.

### 2.2.1 Exogenous Variables

Photovoltaic (PV) power generation and load consumption are dependent on a variety of other time series. Together, they are referred to as multivariate time series. For example, PV generation is positively correlated to solar irradiation. Models are found to benefit from addition of exogenous variables such as local measurements of temperature, relative humidity etc., or numerical weather predictions (NWP) that include pressure, relative humidity, solar irradiance, wind speed and direction, etc. [9]. Therefore, weather conditions are considered to be important in forecasting PV generation. Load is affected by weather changes as well [1]. But it also depends on individual consumption patterns such as users leaving the house during weekdays at certain regular times. This includes seasonality such as weekly and daily trends [5].

### 2.2.2 Individual vs. Aggregated Load

Aggregated load is a sum of electricity usage over several individual households, so it has a more stable and repetitive signal [1]. Individual households exhibit a more randomly variant behaviour, because these trends depend more on individual user's behaviour. It is therefore easier to predict aggregated load, hence there is more research on forecasting aggregated load [5].

### 2.2.3 Forecast Horizons

Load and PV forecasts are usually divided into very short-term forecast (VSTF), short-term forecast (STF), medium-term forecast (MTF), and long-term forecast (LTF), based on the length of the forecast horizon [9, 10]. The choice of horizon depends on the purpose with which the forecast is acquired. In electric power systems, short term forecasts usually provide immediate benefit for grid operations, such as maintenance [9]. Long term forecasts are useful for economical purposes [11]. Moreover, different categories also depend on different variables. VSTF and STF can have an immediate connection to weather, while MTF and LTF do not have such a reliable weather forecasts and therefore rely more on past entries [10].

### 2.2.4 Losses and Metrics

Several metrics are generally used to evaluate the forecast and describe the model's performance due to those metrics having different characteristics [1].

There are widely accepted metrics in different research areas for certain applications. In addiction, the target can be deterministic or probabilistic, which also affects the choice of metric.

Quality of forecasting models is usually quantified using direct, possibly normalized, error measures. Normalized error measures are preferable, since they provide better comparison between results independent of the size of the PV installation [1]. Common direct error measures are (normalized) root mean square error, (n)RMSE, (normalized) mean error, (n)ME, for point forecasts, and the continuous ranked probability score, CRPS, for probabilistic forecasts. For a forecast horizon of $T$ steps $t = 1, \dots, T$ over a target variable with amplitude $P_{\text{max}}$, given a forecast $F$ and real behavior $P$, nME and nRMSE are calculated as follows:

$$\text{nME} = \frac{1}{TP_{\text{max}}} \sum_t |F(t) - P(t)|, \tag{2.1}$$

$$\text{nRMSE} = \frac{1}{TP_{\text{max}}} \sqrt{\sum_t \big(F(t) - P(t)\big)^2}. \tag{2.2}$$

If $F(t)$ and/or $P(t)$ are given as probability distributions, the expected values $\boldsymbol{E}(F(t))$ and $\boldsymbol{E}(P(t))$ are substituted. Within this work, $F(t)$ and $P(t)$ are represented as binned probability distributions over $i_{\text{max}}$ bins. The binned distribution has been chosen instead of parametric approaches as they are not suitable to represent distributions of photovoltaic power or load consumption [1]. To calculate CRPS for such $F(t, i)$ and $P(t, i)$, their probability density functions (PDF) are first converted to cumulative density functions (CDF) and then compared as follows:

$$\text{CRPS} = \frac{1}{i_{\max}T}\sum_t\sum_i\big(F_{\text{cdf}}(t,i) - P_{\text{cdf}}(t,i)\big)^2. \tag{2.3}$$

The best values of nRMSE reported in the recent reviews [1, 9, 12–14] are around 7%. However, no single set of uniformly adapted metrics exists. As performance is reported on specific data and PV power production exhibits both local and temporal patterns, direct error measurements are not robust. In order to limit local sensitivity, many reviews argue for the use of forecast skill score based on nRMSE of the model compared to the persistent forecast [9] as the preferred performance metric

$$\text{S}_{\text{nRMSE}} = 1 - \left(\frac{\text{nRMSE}_{\text{model}}}{\text{nRMSE}_{\text{persistence}}}\right). \tag{2.4}$$

The persistent forecast can be defined as the most recent window of the target variable with the same length as the forecast, but without overlap. For a 24 hour ahead window, this would correspond to the behavior of $P_{-23...0}$. This work uses nRMSE, nME, CRPS and $\text{S}_{\text{nRMSE}}$, as performance metrics for model evaluation. $\text{S}_{\text{nRMSE}}$ is also basis for comparison with other works.

### 2.2.5 Deep Learning in PV and Load Forecasting

As previously mentioned, deep learning techniques have gained success in application to time series forecasting in general, and to PV and load forecasting in particular. Artificial neural networks such as feed forward neural networks were among the first DL techniques to be successfully adapted to forecasting time series and still remain a popular forecasting approach [7, 9]. This proves that deep learning can discover trends and nonlinear relationships in data unlike linear methods. Following that, recurrent neural networks (RNNs) and its variations, long short-term memory (LSTM) and gated recurrent unit (GRU), showed success in time series forecasting and started to outperform ANNs due to their specialization for time series processing [15].

### 2.2.6 Energy Forecasting State-of-the-Art

To establish a baseline for performance comparison, a review of works in energy forecasting was conducted. To the best of our knowledge, these are the best results that are related to our application.

For PV generation, some of the best results documented in the literature were achieved by a fully connected neural network, which exhibit $\mathrm{S_{nRMSE}}$ of 42.5% on an individual model and 46% on an ensemble containing several best performing networks [16]. The authors rely on exogenous variables such as sun position and temperature among others. The results were reported on one dataset and no $k$-fold validation was mentioned. Another comparable result was reported on a Random forest model in [9]. The model was supplied with numerical weather predictions (NWP) as exogenous variables. The reported skill score is $\approx$42%.

For load forecasting, Gasparin et al. test several popular DL techniques, such as LSTM, temporal convolutional network (TCN), and sequence to sequence model for three scenarios [5]. The scenario most relevant to our research is the case where the data contained individual household electric power consumption. For this application, the best result is reported for a GRU based model that achieves the NRMSE score of 9.83%. Another comparable scenario is aggregated electric power consumption with exogenous variables such as temperature and date-time information. An LSTM based recursive model achieves NSRME of 4.59% for this scenario [5].

## 2.3 Deep Learning

This section serves as an introduction to deep learning. A concept of neural network is followed by description of neural network training process. Further, an overview of regularization methods is presented.

### 2.3.1 Neural Network

A neural network is a graph of computational nodes with adjustable weights, $w_n$, that aims to best approximate some function $f^*$ for a system $y = f^*(x)$ and achieve statistical generalization. The nodes composing the network are arranged into layers, the first being the input layer and last the output. The layers between the input and output are called hidden. The number of layers describes the depth of the network. Each layer contains units, which define the width of the network. A small network can contain one layer, and a large deep network may consist of hundreds of layers or more. Similarly, the number of units within a layer of a small network may be less than ten, while a large network contains hundreds. A learning algorithm is applied to optimize the parameters of the model to approximate a desired function in the training process [8]. Refer to figure 2.1 for an example of a neural network with one hidden layer, where values $w_n$ represent the weights.

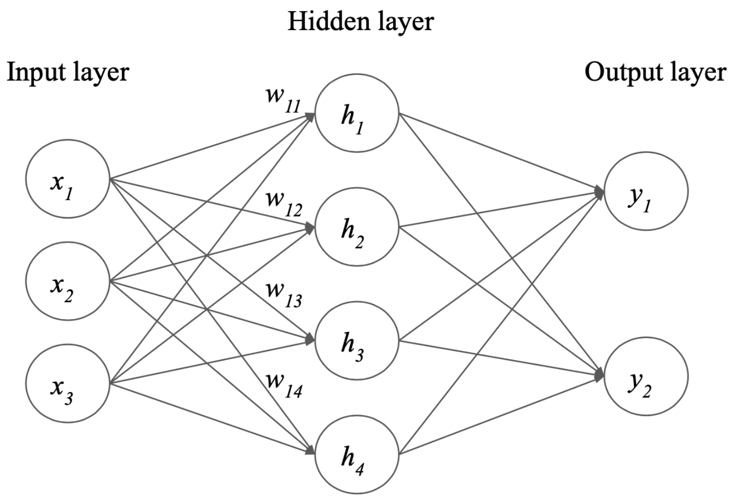

Figure 2.1: A Fully Connected Neural Network.

### 2.3.2 Neural Network Training

The goal of training a neural network is for it to match the true data generating process that is used to obtain the training dataset. This corresponds to the network having zero generalization error and performing accurately on new previously unseen data from the same generating process. Having a low generalization error results from the network having low bias and low variance. Bias is the error between the model's prediction and the target value. Model that has a high bias underfits the data distribution. Variance is a measure of how the model's prediction changes if the dataset is resampled from the underlying distribution. If the variance is high, the model overfits one dataset and does not generalize well to others. From empirical evidence, we know that a large and properly regularized deep neural network fits the data best, which means that it has low bias and variance with respect to the input data. To achieve generalization, different optimization and regularization techniques have been designed [8].

#### Optimization

In order to achieve good performance on the objective function, the network's parameters must to be optimized. To train the model and monitor its performance on previously unseen data, the dataset is split into training set (used for optimization), validation set (used for monitoring the model), and test set (which acts as a new sample and verifies the model's accuracy). Parameter optimization is usually performed by a gradient algorithm through backpropagation.

Gradient descent is an algorithm used to search for a local minimum of the objective function during optimization. After calculating a gradient on the data, the performance on the objective function can be improved by moving in the direction of decreasing gradient. The gradient is computed by backpropagation, which employs the chain rule of calculus. The gradient update is performed in the following way [8]:

$$\hat{\boldsymbol{g}} \leftarrow +\frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), y^{(i)}) \tag{2.5}$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon\hat{\boldsymbol{g}} \tag{2.6}$$

In these equations $\boldsymbol{g}$ is the gradient, $\boldsymbol{\theta}$ represents the model parameters, $\boldsymbol{x}$ and $y$ are input and outputs on the network respectively. $m$ is the size of the training set, or, in case of minibatch gradient descent, the size of the minibatch. And finally, $\epsilon$ is the learning rate.

Momentum is a method that is used to accelerate learning. It keeps an exponentially decaying moving average of past gradients. This way, it can decay or accelerate the gradient depending on the direction of its movement [8]. The gradient update becomes:

$$\boldsymbol{v} \leftarrow \alpha\boldsymbol{v} - \epsilon\nabla_{\boldsymbol{\theta}}\left[\frac{1}{m}L(\boldsymbol{f}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})\right] \tag{2.7}$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \epsilon\boldsymbol{v} \tag{2.8}$$

Here $\boldsymbol{v}$ refers to momentum. There exists a variation of the momentum method called Nesterov momentum, which differs from the regular momentum by being applied before calculating the gradient in a certain step of gradient descent.

Gradient descent is significantly affected by the learning rate parameter. If set inappropriately, a large learning rate can lead to the instability of the network, and a small learning rate can result in an algorithm that never converges. It is therefore important to set learning rate correctly for each task [8]. There also are algorithms that adapt leaning rate during training. They include AdaGrad, RMSProp, and Adam. The latter is more robust to the choice of hyperparameters and is popular in current literature [3,7,8,9].

Usually it is infeasible to calculate gradient on the whole training dataset, so it is estimated from a small subset of the set called a minibatch, randomly drawn from the training set. Small batches can act as a regularizer in the network due to the noise they add to the training process, given the learning rate is small enough to compensate for the noise [8]. Minibatch stochastic gradient follows the gradient of true generalization error but only for the first pass through the training set. However, to decrease the training error, it is advised to run additional epochs, while drawing previously seen samples from the dataset [8].

### 2.3.3 Neural Network Regularization

The goal of regularization is to achieve a small validation error or, otherwise, generalization. This is done by applying penalties or constraints to the model parameters to reduce variance and avoid overfitting. Usually, more than one regularization method is applied to a neural network.

One regularization technique is to impose parameter norm penalties, such as weight decay or $L^2$ norm. It restricts the weights to be closer to zero, and results in lower values of weights, whose covariance with the output target is low. Another algorithm penalizing the weight parameters is $L^1$ norm. It makes the weight matrix sparse, which serves as a feature selection mechanism and simplifies training by discarding a subset of weights [8].

Dropout is a regularization algorithm that mimics the behaviour of bagging or ensemble networks of networks. It applies a mask, which drops some hidden units, thus creating a sub network. Ensembling works because different models usually do not make the same errors on the test set, providing generalization. The difference between ensembling and dropout is that the subnetworks formed by dropout share parameters, which makes it computationally cheap. However, dropout reduces capacity of the model, resulting in a need to increase the model's size [17].

A popular regularization technique is batch normalization [18]. It is a method of reparametrization that reduces the problem of coordinating updates across many layers [8]. Each minibatch of activations is normalized to zero mean, $\mu$, and unit variance, $\sigma$.

$$\boldsymbol{H'} = \frac{\boldsymbol{H} - \mu}{\sigma} \tag{2.9}$$

Two new parameters are learned for the minibatch to have an arbitrary mean and variance. Therefore, the network still retains its expressive power, but learning dynamics are easier.

Unlike batch normalization, layer normalization computes the normalization statistics from the summed inputs to the neurons within a hidden layer. That way all the hidden units in a layer share the mean and variance terms. This method does not impose constraints on the batch size and is easier to apply to recurrent neural networks [19].

With increased depth, networks face a degradation problem, which occurs when adding layers to a network results in a decrease in training error. This problem can be solved by training a network to learn residual mapping, which is done by adding identity shortcut connections [20] that do not add extra parameters to the model. It was empirically shown that such networks are easier to optimize, and they provide an increase in accuracy with increased depth [4, 20, 21].

Another way to ensure generalization is to track training and validation performance for overfitting. During training, the network's training error keeps decreasing, but validation error eventually increases. This represents the bias variance trade-off. It is desirable to have parameters that provide a smaller validation error, because it may result in a smaller test error as well. So, if training process is stopped before the validation error start increasing, we obtain a network with parameters closest to optimal. This algorithm is called early stopping [8].

## 2.4 Deep Learning for Language Processing

Neural networks became widely adopted for statistical language modeling, for example in statistical machine translation (SMT), after the introduction of NN language model [22]. Currently, natural language processing (NLP) achieves near human performance on a wide set of language tasks [23]. Following is an overview of seminal methods and models that led up to this achievement starting with recurrent networks and finally discussing the state of the art.

### 2.4.1 Recurrent Neural Networks

Recurrent neural network (RNN) is an architecture specialized for processing sequences, which could be time series or a language sentence. It is called recurrent because its state at time $t$ depends on its own value at time $t-1$. The state is the hidden unit in an RNN, so it is referred to as hidden state. An RNN relies on the ability to perceive sequential information and store memory of previous time steps through parameter sharing across time via hidden states as illustrated in Figure 2.2. Such parameter sharing results in a very deep computational graph, which can lead to an exploding or vanishing gradient problem [8]. Long short-term memory (LSTM) [24] cells help mitigate this problem by modifying the recurrent cell to introduce gated units. However, they also increase the complexity of the network. Gated recurrent unit (GRU) [22] cells address this issue and propose a new type of cell. In this work, we use a term one-block model referring to a multilayer network, because such a network has one input and one output creating a single block structure.

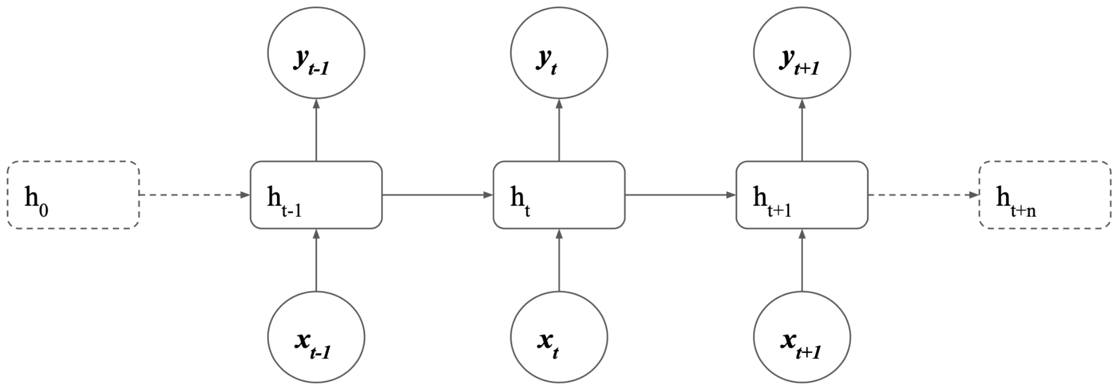

Figure 2.2: A Recurrent Neural Network (RNN).

#### Teacher Forcing

In case a network is recursive, it can be trained with teacher forcing. Recursivity means that the network relies on its previous output to produce the next. This results in a long training process, because the network has to be unrolled for each step. Teacher forcing implies that instead of feeding the network's output back into itself, the network is supplied with the target values [8]. Therefore, the network can be trained in one pass, which simplifies the procedure. However, it can potentially be harmful to train the network this way. Because during inference, when the model is expected to be open-loop, the actual outputs may be different from the targets the network was trained on [8].

### 2.4.2 Sequence to Sequence

Since the success of applying RNNs to language modelling, natural language processing (NLP) became a separate area in deep learning. One of the significant discoveries in this field is the sequence to sequence model by Sutskever [2]. It employs an encoder-decoder architecture, which consists of two complementary recurrent neural networks that have different functions. The encoder processes the input language structure and maps the input sequence to a vector of a fixed dimension, thus learning a latent representation of the sequence. And the decoder recursively produces the desired output, relying on the input sequence representation supplied by the encoder. The networks are connected through state sharing to provide information flow. This means that the initial hidden states of the LSTM network of the decoder are set to be the same as the final hidden states of the encoder network, as illustrated in Figure 2.3. This method elevated the state of the art for translation [2]. However, due to the fixed dimension of the vector, this architecture may suffer from information bottleneck.

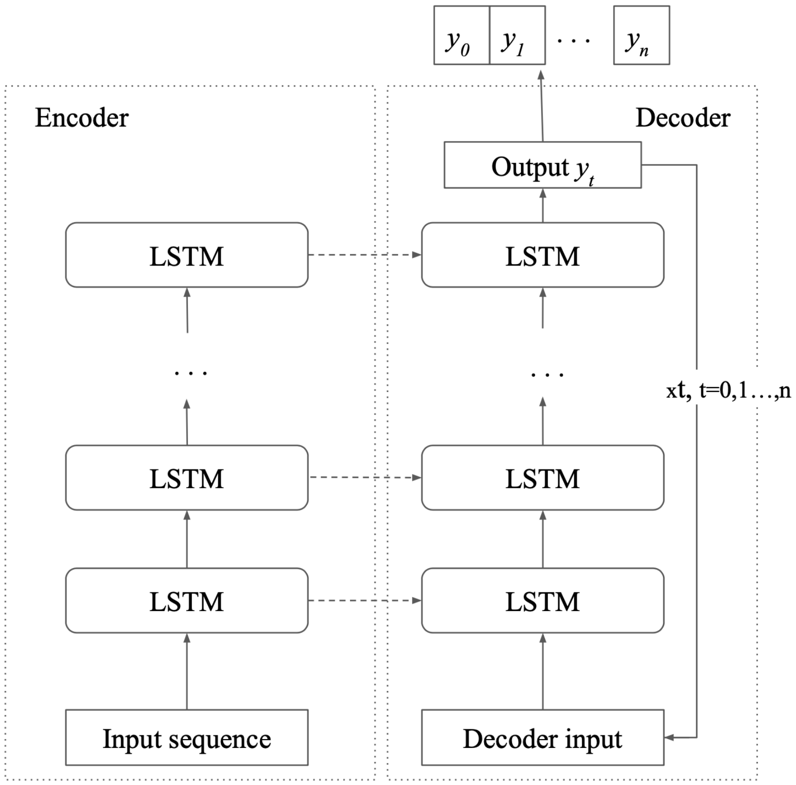

Figure 2.3: Sequence to Sequence network.

### 2.4.3 Attention

Another recently introduced concept within NLP is an attention mechanism. Attention was initially proposed to solve the problem of having a fixed length context vector that results in an information bottleneck. Bahdanau et al. [3] introduced a new recurrent architecture RNNsearch, where a GRU cell was modified to include attention context in the calculation of the gates and hidden states. Attention context determines alignment scores between encoder and decoder hidden states in order to provide more information to the model. The hidden state of the decoder, $s_i$, is dependent on the previous output of the recurrent network, $y_{i-1}$, the previous hidden state, $s_{i-1}$, and the attention context, $c_i$:

$$s_i = f(s_{i-1}, y_{i-1}, c_i). \tag{2.10}$$

The context is calculated based on alignment scores, $\alpha$, and the hidden states of the encoder, $h$:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j. \tag{2.11}$$

The alignment scores are a result of a $softmax$ operation applied to an alignment function, $a$. They reflect how much attention the algorithm should pay to a particular hidden state, $h_j$:

$$\alpha_{ij} = softmax(e_{ij}), \tag{2.12}$$

$$e_{ij} = a(s_{i-1}, h_j). \tag{2.13}$$

Alignment function, $a$, can be arbitrary. In [3], it is a feed forward neural network, designed in the following way:

$$a(s_{i-1}, h_j) = v_a^T \tanh(W_a s_{i-1} + U_a h_j), \tag{2.14}$$

where $v_a$, $W_a$, $U_a$ are weight matrices. It is also sometimes referred to as an additive attention model [21]. The attention mechanism is illustrated in Figure 2.4.

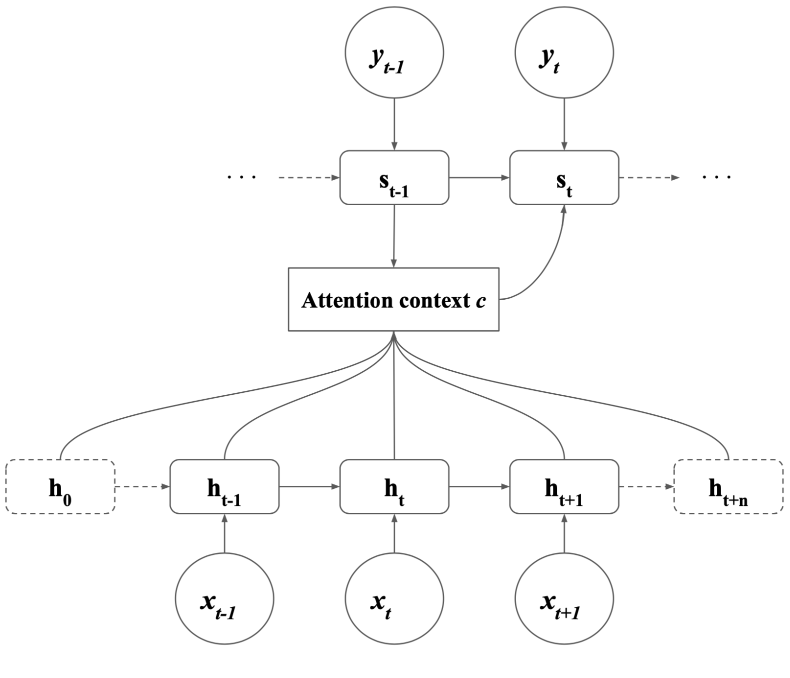

Figure 2.4: Attention Mechanism according to Bahdanau et al. [3]

Luong et al. proposed a simplified way to apply attention with an LSTM based architecture [25]. To take into account the alignment information at previous steps, an input-feeding approach is used in the decoder, where the attention vector is fed back and concatenated with the input at the next time step. The concepts of global and local attention were introduced. Global attention is similar to the one discussed in [3], because it uses all hidden states for the context calculation. On the other hand, local attention selects a small window of context to which to attend to, which is useful for long sequences such as a paragraph of text. The authors tested several types of alignment functions, such as general, dot product, and concatenated. According to the results of the studied translation task, dot product attention performs best [25]. The following equation of the dot product alignment function is adapted from [25] for the ease of notation:

$$a(\mathbf{h}_t, \bar{\mathbf{s}}) = \mathbf{h}_t^T \bar{\mathbf{s}}. \tag{2.15}$$

### 2.4.4 Transformer

The transformer architecture is one of the seminal works in NLP [21]. The main contribution of this work is an introduction of self-attention to replace the RNN. The attention context in self-attention is calculated for a signal with respect to itself. This concept suggests that alignment of a signal with itself extracts features more efficiently and thus can replace a recurrent network in translation and other language tasks.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.16}$$

So, for self-attention query ($Q$), key ($K$), and value ($V$) are equal. The alignment model in this work is a dot-product scaled by a factor $\sqrt{d_k}$ to prevent the softmax operation falling into regions of small gradients [21]. This algorithm outperforms recurrent networks and requires less computational resources than RNNs [21].

### 2.4.5 NLP State-of-the-Art

Current state-of-the-art-language models are based on the transformer. For example, generative pre-training model (GPT) borrows a transformer decoder architecture, and the model called bidirectional encoder representations from transformers (BERT) builds up on it as well [26, 27]. These models achieved state-of-the-art results on such language understanding tasks as question answering, textual entailment, and commonsense inference. One of the ideas that significantly improved the performance of these models is semi-supervised training [27]. This involves pre-training a model on a large unlabelled dataset and fine-tuning to a specific task. Pre-training helps the model gain significant word knowledge and improves generalization, while fine-tuning gives an opportunity to apply the given model to any task and accelerates convergence [27]. Another idea that improved performance of models like GPT and BERT is overparameterization, which occurs when the number of parameters in the model exceeds the size of the dataset. According to experimental evidence, for some models it results in double descent [28]. Double descent is a phenomenon that occurs when an initial dip in the validation curve is followed by a rise and then another dip. This is in contrast to commonly assumed overfitting occurring after the rise in the validation curve. In this case, model's validation and test accuracy increases and after the initial divergence in training and validation curves, it finally converges. This concept is not limited to NLP architecture and was observed for other applications as well.

# Chapter 3

# Architecture Design

This chapter encompasses the process of designing the architectural arrangement used for experiments in this thesis. First, a description of the thought process is presented as a follow-up from Chapter 2. Next, a theoretical argumentation for the proposed sequence to sequence architecture with attention is introduced. Finally, a probabilistic forecasting approach is discussed and model limits identified.

## 3.1 Sequence to Sequence Architecture with Attention

The sequence to sequence model proposed by Sutskever consists of an encoder and a decoder LSTM networks, each having a different input but connected through state sharing [2]. It allows decoupling the functions of both networks. The encoder processes the variables and learns features that appear useful to forecast the main variable, and the decoder uses these features as its initial state. Thus, the sequence to sequence model is said to provide better handling of complicated relationships between sequences [2]. Through successful application to language tasks, this structure provides a tool for analysis of univariate time series, because language can be considered a time series due to being ordered in time and having short-term correlation. In a multivariate time series, the main variable is correlated with several exogenous variables. The encoder-decoder architecture then should be able to learn those relationships better than a general one-block model. Considering these insights, the following hypotheses were made:

1. Encoder-decoder architecture has a potential benefit in forecasting a multivariate time series leveraging exogenous variables and focusing on the target one.
2. Model's forecast will be improved due to the encoder-decoder model's recursive structure.

### Attention

Attention selects which features in the context are most important to the network with respect to the input [3]. Therefore, attention can be seen as a feature extraction tool. A multivariate time series provides a larger amount of information to the network compared to a univariate series. So, the network could potentially benefit more from an attention mechanism, which is designed to alleviate the problem of information loss. Another hypothesis is that attention, serving as an additional input to the decoder, and together with state sharing, will give the decoder an opportunity to leverage more exogenous information and therefore likely produce a more accurate forecast of given time series.

Attention, which is applied between the hidden states of encoder and decoder, serves as an improvement on state sharing mechanism [3]. To keep the state sharing functionality and provide an extra step of feature extraction, attention can be used as an additional input to the decoder [25]. Moreover, for a multivariate time series, attention can extract more context from the features when calculated for each layer in the decoder. The mechanism connects encoder features to the decoder and decides which part of these features to pay attention to. Decoder usually receives one step of a desired variable, but with help of attention more information can be supplied to its layers at input.

### 3.1.1 Applying E-D with Attention to Multivariate Time Series

#### Architecture description

The proposed architecture is a sequence to sequence model consisting of an LSTM encoder and decoder with state sharing, and including scaled dot product attention as the feature extractor between encoder features and decoder inputs. This suggests the following information flow. Please refer to Figure 3.1 for illustration.

The multivariate time series consisting of exogenous variables and the historical signal of the main variable serve as the input to an encoder, which processes them into features. The features are produced through two channels: one being the encoder hidden states, and the second - encoder outputs.

The decoder input is one timestep of the main variable. It is initialized with encoder hidden states and accepts attention features as an additional input. Attention is first performed on the outputs of the encoder and the input to the decoder's first layer concatenated with its hidden states. This serves as the new input to the decoder. Next, attention is performed on the hidden states of the consecutive decoder layer and the encoder outputs and concatenated with the previous layer's output to be passed to the next layer. Thus, encoder features are used in each step. In this case, there is a larger amount of information from the multivariate time series processed in the encoder. Therefore the network will potentially benefit from additional feature extraction steps. Decoder then produces one prediction step, which is fed back as the input to the decoder in the next step of the recursive forecasting process. The decoder unrolls the forecast this way and provides 24 steps as the final output. As discussed in section 2.2.3, different forecast purposes have different horizons and require different exogenous variables. We are interested in (very) short-term forecasts (STF). Within this category, a day ahead hourly sampled forecast (24 hours) is considered the most common [5, 9, 11]. Therefore, it will serve as a good reference point.

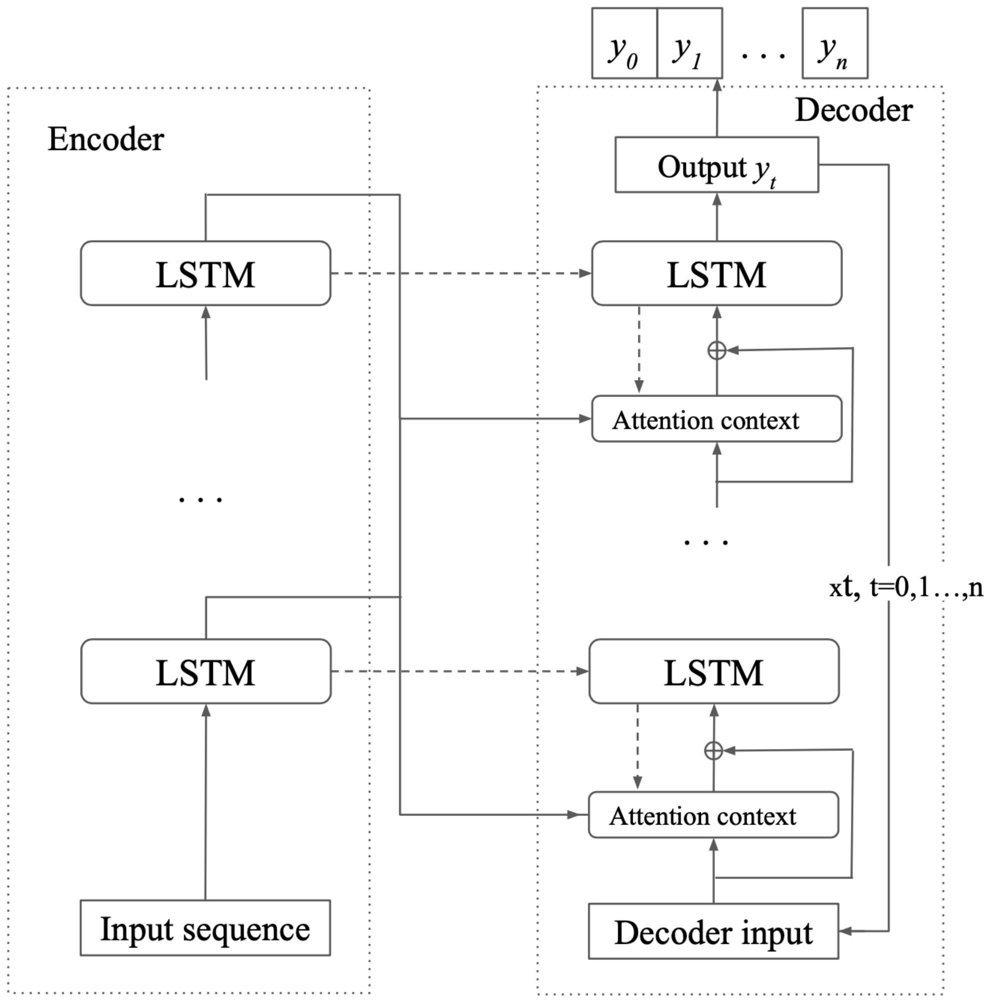

Figure 3.1: Proposed Encoder-Decoder Model with Attention.

### Architecture Implementation

The proposed architecture, *S2S-Attn*, is implemented using LSTM cells as recurrent units, as illustrated in Figure 3.1. The type of attention used here is the scaled dot-product attention proposed by Vaswani et al. [21].

The encoder consist of $n$ stacked LSTM layers. The input to the encoder is a sliding window $t_{-\mathrm{SW}\ldots 0}$ of length SW up to the current timestep $t_0$. After the encoder processes the SW input, it passes its final states to the decoder as the new initial states. The output of the encoder serves as value and key for all attention mechanisms of the decoder.

Sutskever et. al. found that reversing the input sequence order at the input of the encoder helped improve the model, possibly because there is a better connection between the first few words in a sentence [2]. However, the last few points in a time series are the most important in providing information for prediction. So, the time series sequence is not reversed.

The decoder features the same LSTM configuration with the addition of the attention layers. These layers attend to the hidden states of the corresponding LSTM layer as query and encoder outputs as value. The input to the decoder is one step of the binned probabilistic distribution of the PV signal at time zero.

### 3.1.2 Probabilistic Targets

The provided time series signal has 1 minute resolution, and the targets have 1 hour step size. In order to forecast at a required hourly sample rate and utilize the information from a high frequency signal, probabilistic targets were chosen. They inherently convey more information in one step, and thus are potentially more useful to the consumer, because they provide a measure of uncertainty. Learning a more expressive, probabilistic forecast may also provide a way to leverage the improved information propagation of the encoder-decoder model. This allows for more information about past forecast steps to pass on to the next forecast step, freeing up parameters in the model's memory. Recent research has also been moving towards probabilistic forecasts [9, 11].

The model attempts to learn the underlying distribution $P_{\text{true}}(t)$ that generates the actual values of PV power, independent of the forecast format, $P(t)$ or expected value $\boldsymbol{E}(P(t))$. Therefore, if a sufficient number of values are available to construct an approximation $P(t) \approx P_{\text{true}}(t)$, it is beneficial to train with $P(t)$ as target because it is closer to the modeled behavior.

One popular method of probabilistic forecasting is a parametric approach, when a particular probability distribution is chosen and the predicted parameters are the mean and variance [11]. For example, a Gaussian or normal distribution can be used. However, a parametric approach is often considered invalid or suboptimal for PV or load, and a nonparametric approach is preferable [1]. To avoid the bias of the parametric method in constructing $P(t)$, we chose a binned probability distribution of the output variable over each time interval. To affirm the second hypothesis made earlier in this chapter, we suggest the following. Although all models benefit from targets that are closer to the actual behavior in general, the encoder-decoder models benefit further due to their self-recurrence.

Since the goal of the model is to approximate the probability distribution of $P_{\text{true}}(t)$ but only an imperfect, observed $P(t)$ is available, the Kullback–Leibler (KL) divergence emerges as a natural choice of loss function. The KL divergence between two binned probability distributions with bins $i$, forecast $F(t, i)$ and true signal $P(t, i)$, can be calculated as:

$$\text{KL}(F, P) = \sum_t \sum_i -P(t, i)\ ln\left(\frac{F(t, i)}{P(t, i)}\right). \tag{3.1}$$

**Architecture Limits**

State sharing means that encoder and decoder architectures have to be identical [2]. The shapes of hidden states in encoder and decoder have to be the same in order to be reused. This limits the capability to experiment with different shapes of model components and different sizes of inputs. Eliminating state sharing or using architectures that do not require state sharing is a potential solution to this issue.

Another problem is the computational capacity. LSTM and other types of RNNs are well known to require large computational resources, as well as the time required to train such a network because the structure of an RNN requires sequential processing [8]. This also puts a limit on the size of the model we can experiment with.

# Chapter 4

# Experimental Design

This chapter describes the experimental process. It begins with the description of PV and NWP data used in this work. Next, to analyze the hypothesis made in Chapter 3, the experiment is set up to compare the proposed model with a set of benchmarks. To validate the model's performance, a $k$-fold cross-validation is to be performed and design choices made in the first experiment are to be evaluated with further tests. Another experiment is introduced to address the architecture limitations discussed in section 3.1.2. Finally, model's size is tested within possible limits.

## 4.1 Data Description

The PV power data used in the experiments has been provided by *Landmark Homes*. It contains PV power generation values collected with 1 minute resolution at a net zero house in Edmonton between January 2016 and December 2017. The NWP data is a simulation equivalent to the High Resolution Rapid Refresh (HRRR) model, with hourly resolution. HRRR is an NWP model used to provide short-range hourly sampled weather forecasts. It is limited to variables common in PV power forecasting literature: ambient temperature, atmospheric pressure, solar irradiation, wind speed, and relative humidity. The data is smooth without lumpiness or intermittency.

The dataset was created using TFRecord dataset pipeline [29]. One sample contains standardized NWP features, raw historical PV generation, and PDF targets.

The data has been randomly divided into 70%-15%-15% splits for training, testing and validation. For each set, the samples were selected randomly from the available dataset such that the forecast intervals from different sets that overlap are discarded. Thus, no data is repeated in the training, test, and validation sets.

The input to the encoder of a sequence to sequence model or an input to one-block model is presented in Figure 4.1. It contains the NWP features concatenated with the raw PV signal. Since one-block models cannot process datasteams with different resolution without significant architectural change, we consolidate the two data streams (NWP and PV historical data) to one data stream with 15 min resolution by interpolation and averaging, respectively. For each sample, every model receives a 5-day length equivalent sliding window from this data stream as input. The sample contains temperature, ACSWDNB (accumulated downwelling shortwave flux at the bottom), solar irradiance, relative humidity at 2 meters height, surface level pressure, wind speed and direction, and photovoltaic generation. The NWP variables are shifted to provide the weather forecast for the next day. The PV signal consists of historical PV generation. The NWP and PV sample that does not include the NWP forecast is demonstrated in Figure 4.2.

Figure 4.3 depicts the corresponding binned probability distribution. The target is plotted to the right of the red line. Historical signal is added for illustration purposes and is not included in the model training sample. Persistent forecast, which is used to calculate the baseline metrics, is also demonstrated.

### 4.1.1 Training Setup

The experimental setup was implemented in Tensorflow using Keras library [29].

Models were trained with stochastic gradient descent with learning rate of 0.003 and Nesterov momentum of 0.75 as optimizer. The loss was set as the KL divergence as discussed in 3.1.2. Batch size was 128, and was limited by the computational resources. Early stopping was used after 10 epochs in case the nRMSE error on the

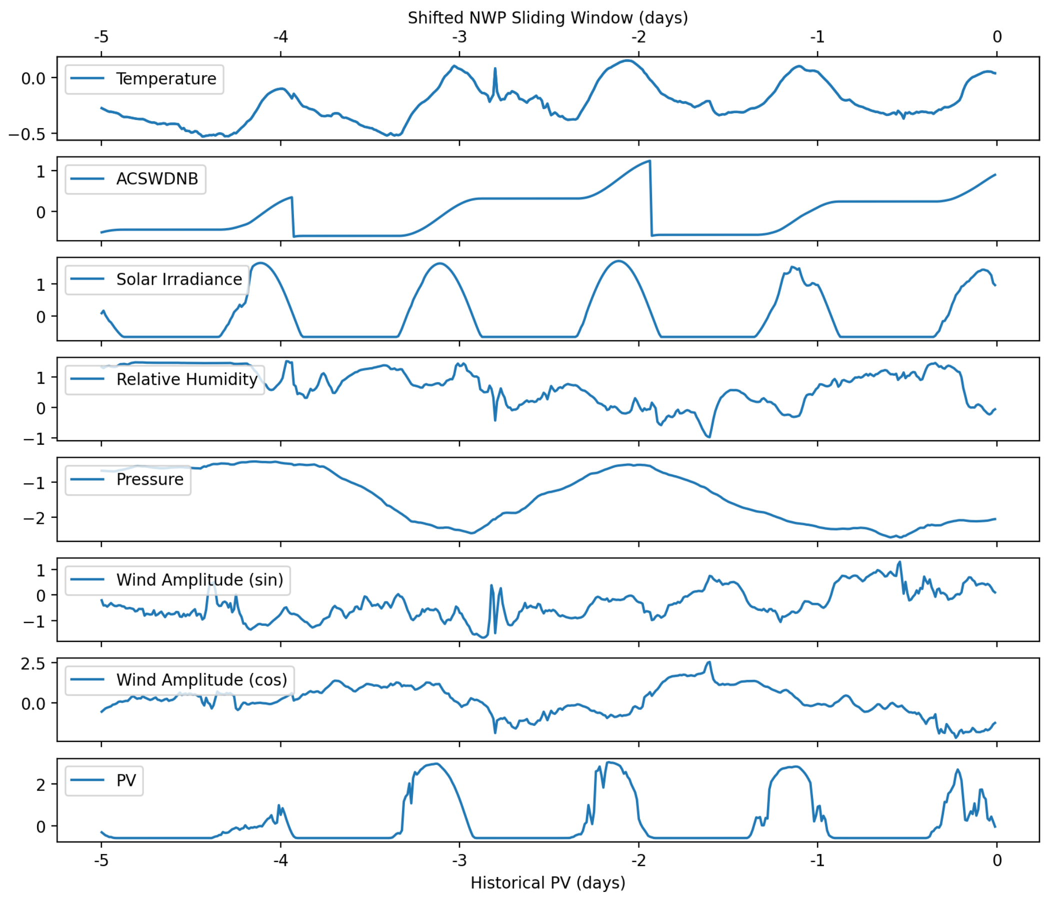

Figure 4.1: Shifted NWP and PV Input Sample.

validation set did not improve.

The models forecasting $\boldsymbol{E}(P(t))$ were trained using mean squared error (MSE) as a loss function

$$\text{MSE} = \frac{1}{T}\sum_{t}(F(t) - \boldsymbol{E}(P(t)))^2, \tag{4.1}$$

while $\boldsymbol{E}(P(t))$ ranges from 0 to 1.

## 4.2 Benchmarks

Initially we formulate two hypotheses:

1. The proposed model can outperform common one-block models used to forecast

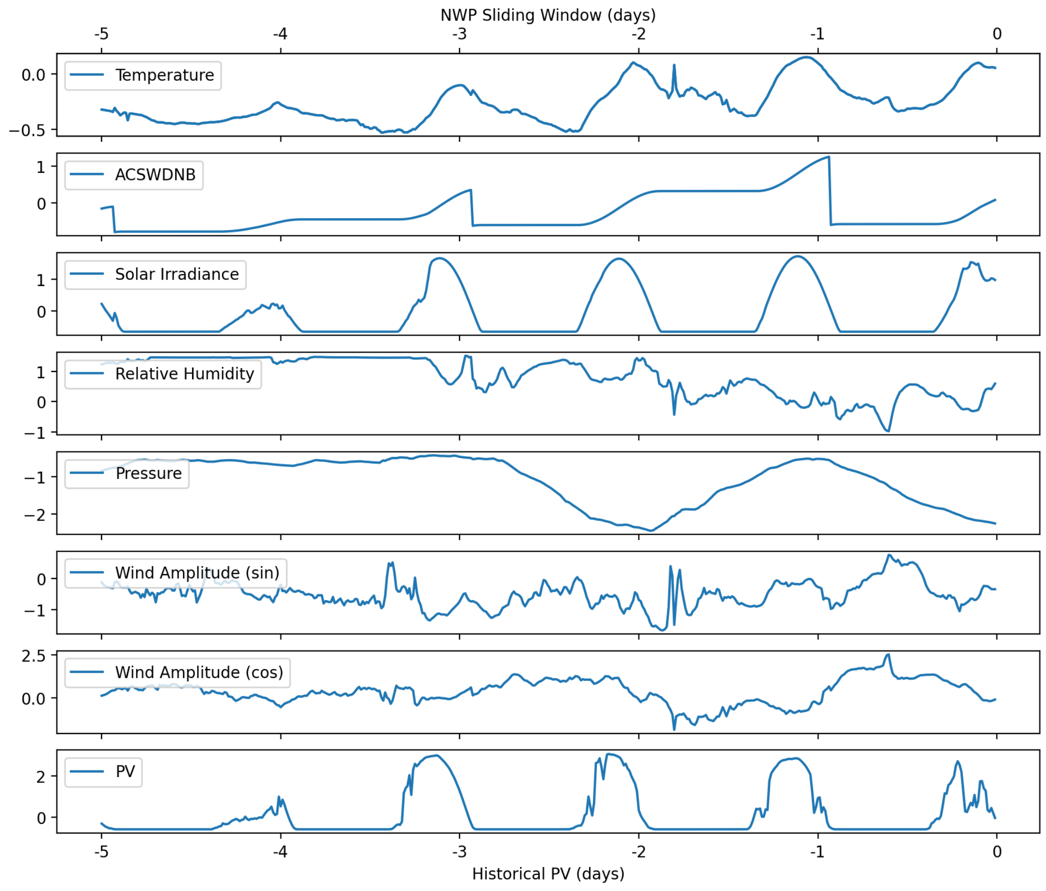

Figure 4.2: NWP and PV Input Sample.

PV power and achieve or surpass state of the art performance in terms of the established error measures.

2. Performance improvements stem from predicting $P(t)$ instead of $\boldsymbol{E}(P(t))$ on one hand, and better context extraction through the encoder-decoder architecture with attention and self-recurrence on the other.

The first claim is evaluated by testing the proposed model, *S2S-Attn*, against a set of benchmark models. The second claim is quantified by training 2 types of each model: one trained to directly forecast $\boldsymbol{E}(P(t))$ over the target time interval, denoted *model-E*, and one trained to forecast $P(t)$, denoted *model-pdf*. To further assess the impact of

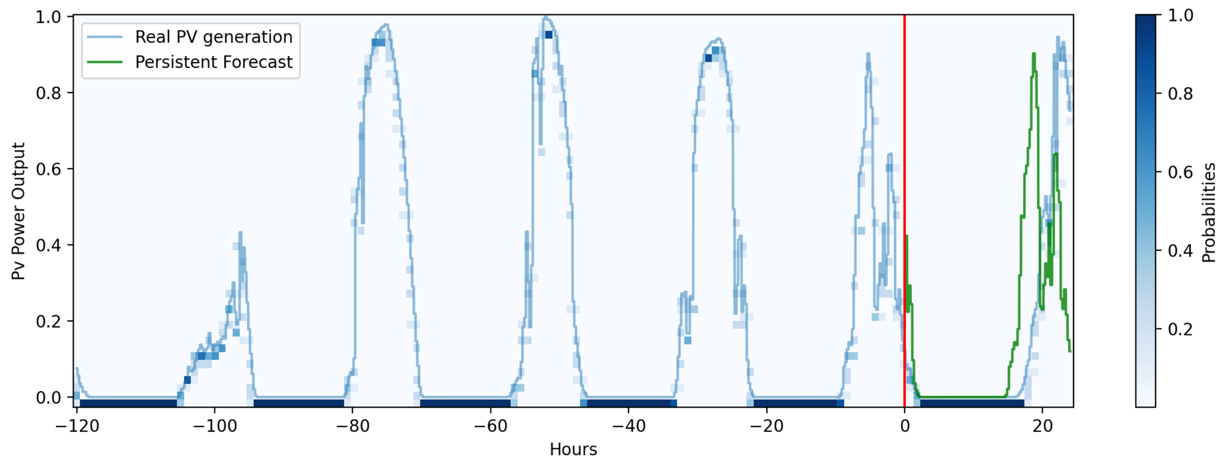

Figure 4.3: Binned Probabilistic Signal.

the attention mechanism we also train a classic encoder-decoder model corresponding to the proposed network without attention, *S2S*. The baseline models were also chosen among those used in energy forecasting literature [5, 16]. In summary, the evaluated models are:

1. *Persistence*: the persistent forecast model as probabilistic model, using the previous day $P(-23, ..., 0)$ as the forecast $F(1, ..., 24)$. This model is used to calculate $\mathrm{S}_{\mathrm{nRMSE}}$ and enable comparison with other published models.

2. *FFNN-pdf*, *FFNN-E*: probabilistic and expected value versions of a one block FFNN model.

3. *LSTM-pdf*, *LSTM-E*: probabilistic and expected value versions of a one-block LSTM model.

4. *S2S-pdf*, *S2S-E*: probabilistic and expected value versions of a sequence to sequence model without attention.

5. *S2S-Attn-pdf*, *S2S-Attn-E*: probabilistic and expected value versions of the proposed sequence to sequence model with attention.

### 4.2.1 Implementation of Benchmark Algorithms

We attempt to isolate architectural performance influences by keeping the number of parameters, hyperparameter settings and received input data consistent. For simplicity, the number of units per layer in each model is kept constant. Each *S2S* model has a 2-layer encoder and a 2-layer decoder similarly to the *S2S-Attn* model as discussed in section 3.1.1. The *S2S* model is designed according to [2] and is illustrated in Figure 2.3. Both one-block *FFNN* and *LSTM* models consist of stacked feedforward and LSTM layers respectively. Additionally, they feature a temporal transformation layer. It projects the output of the last layer of the network into the required output shape. In this article, the transformation is a feedforward layer. It reduces the number of timesteps to 24 as per the setup and then transforms the output to the shape of $\mathrm{E}(P(t))$ or $P(t)$. Please refer to Figure 4.4 for one-block *LSTM* model. The *FFNN* model is constructed similarly, but with feedforward layers.

Each model is trained to perform hourly forecasts up to 24 hours ahead. For *-pdf* models, each hour is a probability distribution with 50 bins from 0 to maximum rated power of the PV installation, while for *-E* models it is the normalized expected PV power.

To accommodate the probabilistic forecast, the output of the decoder is projected into a binned probability distribution $F(t)$ by a projection layer that features a softmax operator. In the first decoder step, to forecast $F(1)$, the input is the available $P(0)$. This stands in contrast to similar models for language oriented applications, where the first input is usually a start-of-sentence token. The decoder is trained with teacher forcing, meaning it receives $P(t-1)$ rather than $F(t-1)$ to learn how to forecast $F(t)$. During evaluation, the decoder is fully self-recurrent and predicts $F(t)$ based on $F(t-1)$. This has been established as the best practice to speed up convergence for encoder-decoder models in language-related fields.

In order to keep the model size equivalent for all models, the number of units was

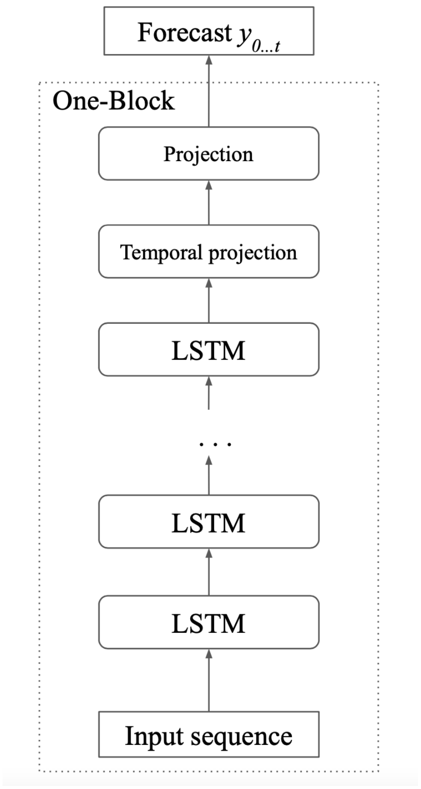

Figure 4.4: One-block LSTM Model.

changed. However, for simplicity, all layers in a given model have the same number of units. Table 4.1 enumerates the parameters used for model evaluation.

## 4.3 Design Choices: *k*-fold Cross-Validation

When a universal benchmark dataset is not available, it is useful to validate model performance. Using skill score as a metric aids in model performance comparison across other works. Additionally, further experiments can be conducted on data from different domains or other datasets available from the same domain. However, it is also necessary to assert model performance on different splits of data within one application.

It was originally proposed to test the models on a load consumption data. However, initial experiments determined that the current experimental setup is not directly transferable to this application. This can be explained by the fact that, according to

| Model | Number of units per layer | Number of parameters |
|---|---|---|
| *FFNN-E* | 640 | ∼428k |
| *FFNN-pdf* | 616 | ∼428k |
| *LSTM-E* | 184 | ∼425k |
| *LSTM-pdf* | 184 | ∼434k |
| *S2S-E* | 132 | ∼425k |
| *S2S-pdf* | 128 | ∼431k |
| *S2S-Attn-E* | 115 | ∼441k |
| *S2S-Attn-pdf* | 110 | ∼423k |

Table 4.1: Benchmark Model Architecture Details.

section 2.2.1, load requires a different set of exogenous variables and possibly a change of hyperparameters controlling the data pipeline, for example sliding window size. So, with an appropriate dataset generation and experimental setup, the networks' performance will potentially be comparable on load data.

The results in [30] are reported on one dataset. And it was further observed that the creation of the dataset by random sampling affects the model performance. In order to validate the test error of the model, $k$-fold cross validation is commonly used. Train, validation, and test sets are randomly chosen from the dataset $k$ times [8]. Therefore, before implementing the changes required for a load dataset, it is proposed to perform $k$-fold validation on the PV generation dataset.

As noted above, the dataset contains two years of PV and NWP data. The amount of data available for training the model is important, since models usually perform better when trained on larger datasets [4, 27]. In addition, about seven months of that data are used to construct validation and test sets. Therefore, two years may not be enough for the model to capture the data generating distribution well. Moreover, the PV generation data in Edmonton exhibits a problem, where in winter, during the periods of snowfall, the generation of solar power is erratic. This decreases the quality of the datasets and reduces the information quantity. Randomly distributing this dataset into train, validation, and test subsets can result in unreliable samples.

An experiment is set up to investigate this. The original dataset used in [30] was lost. So, the new datasets were generated from the same NWP and PV data with a new pipeline and imitating the settings used to obtain the original dataset, such as the procedure used for splitting and saving the samples, but with a different random split into train, validation, and test sets. The random seeds were changed for each split.

Additionally, in the original experiment the models were trained on an NWP sample shifted a day ahead concatenated with the historical PV data. In order to confirm this selection, the samples containing the non-shifted NWP sliding window were created. The same sample from Figure 4.1 is changed to Figure 4.2. To further justify the advantage of the exogenous variables, the models are trained excluding the historical PV variable from the encoder input sample and with shifted input sliding window.

A further untested design choice is teacher forcing. As discussed in section 2.4.1, training models with teacher forcing may be harmful. Therefore, another parameter changed in this experiment is training with or without teacher forcing.

Consequently, the experiment comprised of training three models: *LSTM-pdf*, *S2S-pdf*, and *S2S-Attn-pdf*, on four datasets, additionally testing the application of teacher forcing and the NWP shift.

## 4.4 Original Attention Test

Another hypothesis implied in the proposal is that our model outperforms a classic encoder-decoder with attention model. However, the proof of this was omitted in the publication [30].

To assess the claim that the proposed model benefits from additional attention steps and achieves better performance than a sequence to sequence model with one attention layer, an encoder-decoder model structured according to [25] is used as a baseline to compare to the proposed model.

The network used for this experiment follows a sequence to sequence architecture similar to the proposed model. The attention mechanism is calculated based on the dot product alignment model discussed in [25]. The attention context is derived from the hidden state of the top LSTM layer in the decoder and the hidden states of the encoder. The context is then concatenated with the hidden state of the top decoder LSTM layer and is used to calculate attentional vector, which in turn produces the output through softmax function. The input-feeding approach is also adapted, concatenating the attentional vector with the input of the decoder at the next time step. See Figure 4.5 for an illustration of the architecture.

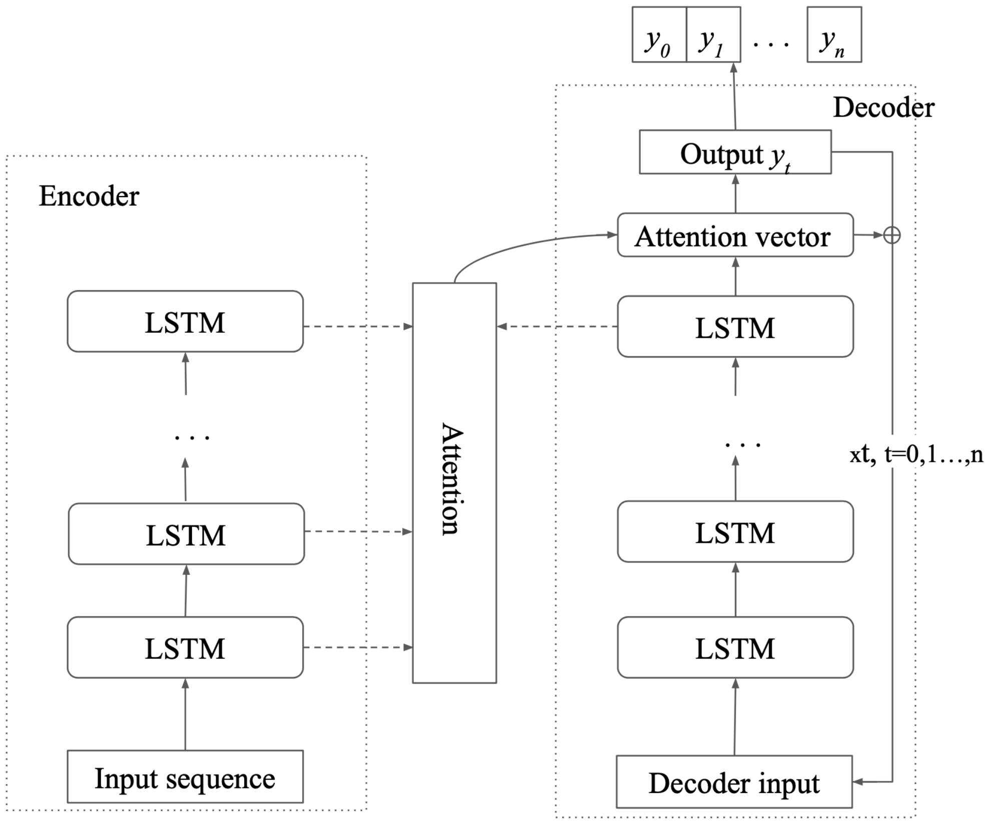

Figure 4.5: Encoder-decoder Model with Attention according to Luong [25].

The models in the experiment are set up to have the same number of layers and number of units for a fair comparison. At $t = 0$ the input is not concatenated with the attentional vector. So, an input projection layer is added to the decoder at $t = 0$ to provide the required dimension to the input. The rest of the training setup, such as regularization, is also kept the same.

## 4.5 Recurrent Cell Test

LSTM is the chosen type of recurrent cell because it is assumed to have the best performance compared to RNN, reducing the problem of exploding/vanishing gradient problem by introducing gated units. The addition of these gated units, however, adds parameters to the network thus further increasing the requirements on computational resources. And the possibility of utilizing different cells was not explored in the publicatoin [30]. GRU [22] is another type of recurrent cell, that also utilizes gated units, but reduces the number of parameters compared to LSTM. Recurrent networks with GRU type cells have similar performance to those with LSTM cells. This experiment is set to explore the possibility of using GRU type cells in the sequence to sequence model with attention.

Firstly, the two models are set to the same size. This will help determine whether GRU type cells can perform similarly to LSTM. Then, the models are set up such that the number of trainable parameters is the same. This means the model with GRU type cells will be larger as detailed in Table 4.2. The result of this experiment will help obtain understanding of whether further performance improvements are possible with less computational requirements.

| **Model** | **Number of units per layer** | **Number of parameters** |
|---|---|---|
| *S2S-Attn-LSTM* | 110 | ∼423k |
| *S2S-Attn-GRU-small* | 110 | ∼320k |
| *S2S-Attn-GRU-large* | 128 | ∼428k |

Table 4.2: LSTM/GRU Model Parameters.

## 4.6 Model Parameter Tuning

In [30], the model size was chosen arbitrarily, based on trial and error tests as well as local memory limitations. It is nevertheless important to confirm that selection or possibly find another well performing configuration. To obtain optimal performance and keep the training within accessible computational bounds, it is necessary to find the suitable model size. The parameters tuned in this experiment are the width and number of layers in the proposed model. Grid search was applied, with parameters ranging from ~69k to ~4.8m considering the numbers of units below and above those in the proposed model. This was done to identify the direction in which the model's performance demonstrates improvement. Table 4.3 presents the model sizes tested in this experiment.

The experiment was set up using the same training settings as other experiments above, but training the models on one dataset. The only exception was the *S2S-Attn-4x256* model, which was trained with batch size 64 due to local memory limitations.

| **Model** | **Number of layers** | **Number of units per layer** | **Number of parameters** |
|---|---|---|---|
| *S2S-Attn-1x64* | 1 | 64 | ~69k |
| *S2S-Attn-1x128* | 1 | 128 | ~237k |
| *S2S-Attn-1x256* | 1 | 256 | ~867k |
| *S2S-Attn-2x64* | 2 | 64 | ~152k |
| *S2S-Attn-2x128* | 2 | 128 | ~566k |
| *S2S-Attn-2x256* | 2 | 256 | ~2180k |
| *S2S-Attn-4x64* | 4 | 64 | ~317k |
| *S2S-Attn-4x128* | 4 | 128 | ~1223k |
| *S2S-Attn-4x256* | 4 | 256 | ~4805k |

Table 4.3: Parameter Tuning Setup.

# Chapter 5

# Discussion

## 5.1 Experiment 1: Benchmarks

The initial experiment included testing the proposed model against a set of benchmarks, which were designed to have the same number of parameters as discussed in section 4.2. The point error metrics for probabilistic versions of the models were calculated after taking an expected value of the binned probability distribution. The results are reported on the dataset described in section 4.1 and with the training setup described in 4.1.1.

As can be seen from Table 5.1, according to the test set performance, the proposed *S2S-Attn-pdf* model outperforms all baseline models ($\mathrm{S}_{\mathrm{nRMSE}} = 48.1\%$). The second best performance on the test set is achieved by model *S2S-pdf* ($\mathrm{S}_{\mathrm{nRMSE}} = 45.6\%$). This confirms the first hypothesis that encoder-decoder models are better suited for a setup with high resolution data. Comparing validation and test performance, *S2S-Attn-pdf* exhibits a smaller generalization gap then *S2S-pdf*. This is likely due to the ability of attention to aid generalization through added context.

Another interesting observation is that performance gains from performing probabilistic forecasts are obvious and strong for the tested encoder-decoder models *S2S-Attn* and *S2S*. This is not the case for one-block models whose results are less conclusive: model *FFNN-pdf* outperforms *FFNN-E*, while *LSTM-E* outperforms *LSTM-pdf* for the specific set of hyperparameters listed in section 4.2. Thus, it appears that the per-

|  | nRMSE |  | nME |  | CRPS |  | $S_{\text{nRMSE}}$ |  | $S_{\text{CRPS}}$ |  |
|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | *Val* | *Test* | *Val* | *Test* | *Val* | *Test* | *Val* | *Test* | *Val* | *Test* |
| *Persistence* | 0.145 | 0.133 | 0.063 | 0.052 | 2.285 | 1.944 | - | - | - | - |
| *FFNN-E* | 0.078 | 0.083 | 0.054 | 0.057 | - | - | 0.464 | 0.376 | - | - |
| *FFNN-pdf* | 0.072 | 0.074 | 0.041 | 0.040 | 1.008 | 1.032 | 0.501 | 0.446 | 0.559 | 0.469 |
| *LSTM-E* | 0.073 | 0.080 | 0.049 | 0.054 | - | - | 0.496 | 0.395 | - | - |
| *LSTM-pdf* | 0.080 | 0.087 | 0.045 | 0.047 | 1.166 | 1.386 | 0.450 | 0.344 | 0.490 | 0.287 |
| *S2S-E* | 0.089 | 0.100 | 0.058 | 0.065 | - | - | 0.388 | 0.249 | - | - |
| *S2S-pdf* | 0.068 | 0.072 | **0.039** | 0.039 | 0.938 | 1.003 | 0.529 | 0.456 | 0.589 | 0.484 |
| *S2S-Attn-E* | 0.119 | 0.121 | 0.091 | 0.094 | - | - | 0.184 | 0.089 | - | - |
| *S2S-Attn-pdf* | **0.067** | **0.069** | **0.039** | **0.038** | **0.917** | **0.937** | **0.536** | **0.481** | **0.599** | **0.518** |

Table 5.1: Benchmark Model Performance Comparison

formance improvement of the encoder-decoder models is not only due to the more expressive, probabilistic target, as the *-pdf* variants of the one-block models are not consistently affected in the same way. Conversely, the performance improvement cannot solely be attributed to self-recurrence of encoder-decoder models, as *S2S-Attn-E* and *S2S-E* significantly under perform. The performance of *S2S-Attn-pdf* and *S2S-pdf* therefore stems from the combination of self-recurrence and rich output representation. Latter enables efficient utilization of the former and is especially relevant for $F(1)$, as the probabilistic input $P(0)$ conveys more information about the state of the system at $t = 0$ than $\boldsymbol{E}(P(0))$. The same argumentation can be applied to the attention mechanism. When compared to model *S2S-Attn-E*, the probabilistic output of *S2S-Attn-pdf* provides significantly more features to the first layer's attention mechanism. This allows to construct better attention-based context and results in an additional boost of performance. This validates the hypothesis from the beginning of Chapter 3.

There is no universally agreed upon benchmark dataset for PV forecasting and many authors use custom datasets for experiments. Additionally, our experimental setup necessitates the use of a custom dataset to demonstrate efficient utilization of data with high temporal resolution. Although the proposed model *S2S-Attn-pdf* is well within the range of state of the art methods concerning the reported direct metrics, comparison based on such metrics is unreliable given the circumstance. Many sources argue that using nRMSE based Skill compared to a persistent baseline largely alleviates dataset dependency. From reviewed works, the highest reported values of $\mathrm{S}_{\mathrm{nRMSE}}$ are between 42.5-46% [9, 16]. This is significantly lower than the skill of the proposed model *S2S-Attn-pdf* with $\mathrm{S}_{\mathrm{nRMSE}}$ of 53.6% and 48.1% for validation and test sets, respectively. In terms of $\mathrm{S}_{\mathrm{CRPS}}$ the proposed model also features the largest relative increase of this metric (51.8%) with respect to other models evaluated by van der Meer and Munkhammar [1]. However only one study [31] reports a CRPS improvement over persistence in a scenario similar to that considered in this experiment.

## 5.2 Experiment 2: $k$-fold Cross Validation

According to section 4.3, for this experiment, four datasets were created and three models were used for evaluation: *LSTM*, *S2S*, *S2S-Attn*. Two scenarios were considered. One was the application of teacher forcing and another shifting the NWP data to include the forecast for the following day. The results are reported in Table 5.2. Teacher forcing is not applicable to the one-block models in this experiment, so they were omitted from the experiment for those cases. The examined scenarios are:

1. SW/no TF: sliding window not including the next day forecast and no teacher forcing used for training.

2. SW/TF: sliding window, training with teacher forcing.

3. Shifted SW/no TF: shifted sliding window including the next day forecast, no teacher forcing.

4. Shifted SW/TF: shifted sliding window and training with teacher forcing.

| | SW/no TF | | SW/TF | | Shifted SW/no TF | | Shifted SW/TF | |
|---|---|---|---|---|---|---|---|---|
| **Model** | *Val* | *Test* | *Val* | *Test* | *Val* | *Test* | *Val* | *Test* |
| *LSTM* | 12.6% | 11.2% | - | - | 28.9% | 29.9% | - | - |
| *S2S* | 15.4% | 15.8% | 14.9% | 16.7% | 31.2% | 34.5% | 30.1% | 33.1% |
| *S2S-Attn* | 15.6% | 17.1% | 16.4% | 15.4% | 30.3% | 35.6% | 30.3% | **36.2%** |

Table 5.2: Skill nRMSE for $k$-fold experiment.

The last scenario is used in the experiment described in section 5.1.

For each dataset, validation, test, and training sets were sampled randomly in order to avoid seasonal bias in training and evaluation.

Considering the results in Table 5.2 (refer to Table A.1 for full results), the *S2S-Attn* model outperforms other models for the Shifted SW/TF case, with the average test nRMSE skill score of 36.2% for the discussed training setting. However, it is evident that the performance of the models do not directly replicate the above reported scores. As discussed in section 4.3, this can be explained by the specific dataset issues such as small amount of data. The conclusion that the performance of the models is affected by the dataset and not by the model architectures is made considering that the relative model performance is consistent in the $k$-fold cross-validation.

A significant performance gain can be observed for the models that received shifted sliding window (SW) comparing to those that did not. This confirms the intuition that providing the next day forecast aids the photovoltaic power forecasting due to the correlation of the NWP variables with PV generation. Moreover, it is visible from the table that encoder-decoder model with attention trained with teacher forcing outperforms that without teacher forcing for the shifted SW scenario. Thus, the design choice of applying teacher forcing is justified. However, an interesting observation is that the *S2S* model achieves better results without teacher forcing. Perhaps, attention benefits when it is trained on actual targets. Another notable fact is that *S2S-Attn* model demonstrates a larger generalization gap compared to *LSTM* and *S2S*, although

| | Shifted SW/no TF | | Shifted SW/TF | |
|---|---|---|---|---|
| **Model** | *Val* | *Test* | *Val* | *Test* |
| *LSTM* | 28.5% | 29.9% | - | - |
| *S2S* | 29.2% | 31.3% | 28.6% | 28.2% |
| *S2S-Attn* | 27.4% | 31.7% | 26.9% | **33.4%** |

Table 5.3: Skill nRMSE for PV ablation experiment.

it achieves better performance on the test set than other models.

Table 5.3 presents the results of the PV signal ablation study. For complete results refer to Table A.2 in Appendix A. The values follow similar trends as those including the PV history, such as teacher forcing improving the performance of the attention model but harming *S2S*, and the attention model having a larger generalization gap. And although the models' performance obviously suffers from excluding the PV signal, the models nevertheless achieve results similar to above. In fact, this experiment demonstrates that the majority of the performance gain for the models comes from the NWP variables with the attention model achieving the nRMSE skill score of 33.4%. The proposed attention model also performs the best among the benchmarks. It further confirms that attention aids feature extraction.

Figures 5.1 and 5.2 demonstrate, respectively, the actual target and the forecast distribution produced by the proposed model. The historical signal is added for context, both target and forecast are on the right of the red line. This illustrates the case where the model succeeds in predicting PV generation despite the persistent forecast failing. For more examples of forecasts please see Appendix A.

## 5.3 Experiment 3: Attention

The proposed model was compared to the classical sequence to sequence model with attention built according to [25]. The reported results are averaged over 4 datasets

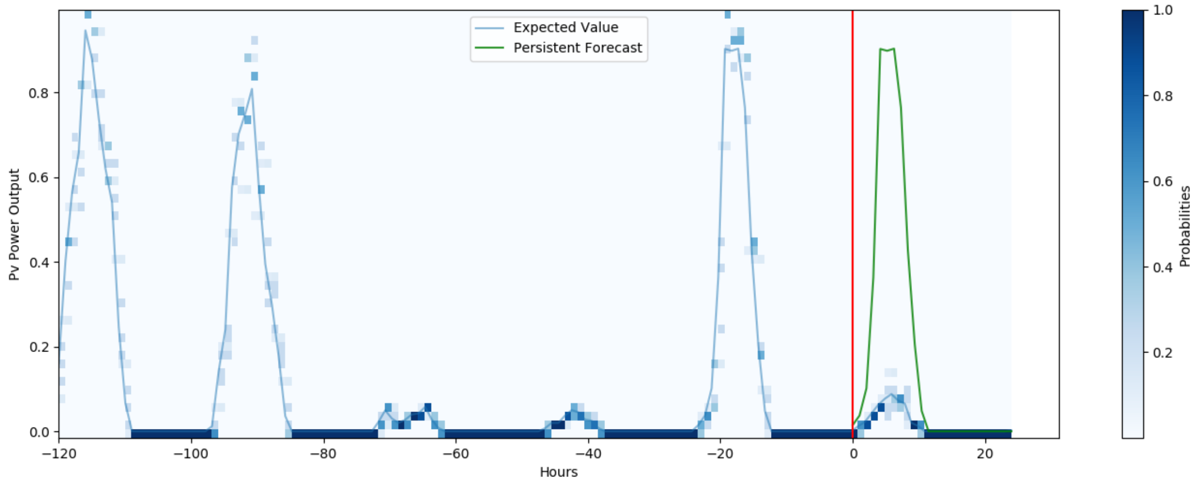

Figure 5.1: Actual PV Generation PDF Signal.

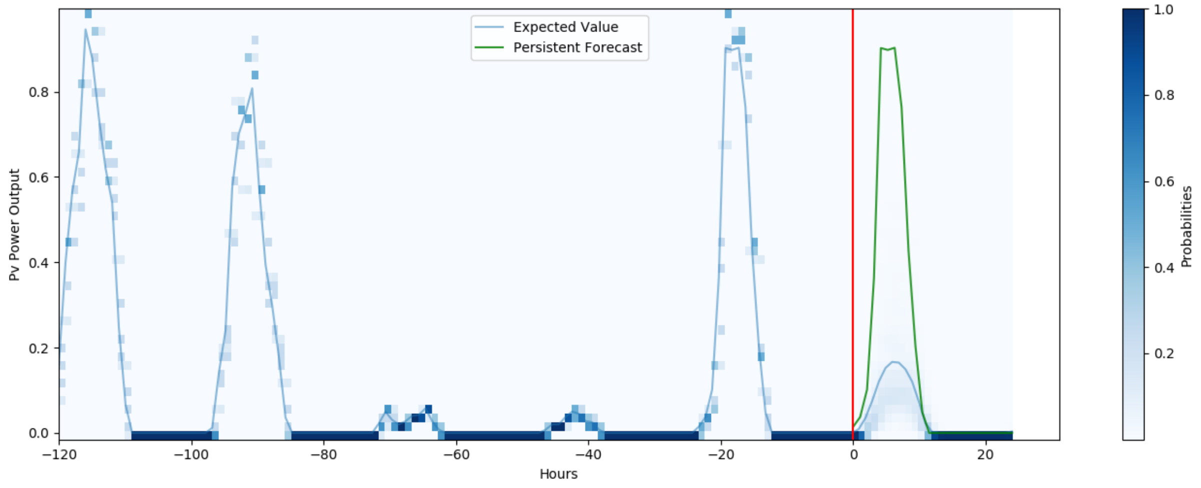

Figure 5.2: PV Generation History and Forecast PDF.

| | nRMSE | | nME | | CRPS | | $S_{\text{nRMSE}}$ | | $S_{\text{CRPS}}$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | *Val* | *Test* | *Val* | *Test* | *Val* | *Test* | *Val* | *Test* | *Val* | *Test* |
| *S2S-Attn-Luong* | 0.092 | 0.088 | 0.052 | 0.049 | **1.320** | 1.242 | 0.300 | 0.347 | **0.491** | 0.537 |
| *S2S-Attn* | 0.092 | **0.086** | 0.052 | **0.048** | 1.333 | **1.228** | 0.303 | **0.362** | 0.487 | **0.543** |

Table 5.4: Attention Model Comparison.

and demonstrated in Table 5.4. The input to the encoder is the shifted NWP signal, and both models are trained with teacher forcing. The model designed according to Luong et al. [25] is referenced as *S2S-Attn-Luong*.

From the results on the test set, the proposed model has superior performance for the specific hyperparameters used in this experiment. However, an interesting observation is that the model based on Luong architecture performs equivalently to the proposed model on the validation set, with some metrics having higher scores. This implies that the proposed model generalizes better. This adds to the conclusion made in the previous experiment, which showed better generalization of the attention model. A subsequent suggestion is that additional attention layers provide further improvement in generalization over a single attention layer by extracting more context from the features.

## 5.4 Experiment 4: GRU

For this study, an application of GRU recurrent units instead of LSTM was considered. The experiment was set up according to the discussion in section 4.5.

According to the results in Table 5.5, the model with LSTM units still performs best regardless of the number of parameters in the GRU based models. However, an interesting note is that the smaller GRU model, *S2S-Attn-GRU-small* with 110 units in each layer, performs marginally better than the larger model, *S2S-Attn-GRU-large* with 128 units and the parameter count equivalent to the LSTM based model. GRU is

| | nRMSE | | nME | | CRPS | | $\mathrm{S}_{\mathrm{nRMSE}}$ | | $S_{\mathrm{CRPS}}$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | *Val* | *Test* | *Val* | *Test* | *Val* | *Test* | *Val* | *Test* | *Val* | *Test* |
| *S2S-Attn-GRU-small* | 0.094 | 0.090 | 0.053 | 0.050 | 1.342 | 1.263 | 0.292 | **0.336** | 0.484 | **0.530** |
| *S2S-Attn-GRU-large* | 0.094 | 0.091 | 0.053 | 0.051 | 1.350 | 1.279 | 0.287 | 0.327 | 0.480 | 0.524 |
| *S2S-Attn-LSTM* | 0.092 | 0.086 | 0.052 | 0.048 | 1.333 | 1.228 | 0.303 | **0.362** | 0.487 | **0.543** |

Table 5.5: Recurrent Unit Comparison.

easier to implement, but it has only two gates within the hidden unit, while LSTM has four. This may affect the memory collection ability of the recurrent unit. Furthermore, LSTM units seem to be more frequently adopted for forecasting [4]. This may attribute for the result above, and support the choice of using LSTM units in the proposed model.

## 5.5 Experiment 5: Model Parameter Tuning

To confirm the parameters chosen for the previous experiment and explore directions for further improvement, a parameter tuning experiment was set up.

As noted from Table 5.6, the best performing architecture is *S2S-Attn-2x256*, which achieves an $\mathrm{S}_{\mathrm{nRMSE}}$ of 41.1% on a particular dataset in this experiment. This indicates that the direction for improvement is to increase width and not depth of the network. However, model *S2S-Attn-2x128* also exhibits good results being the second best model on the test set and achieving $\mathrm{S}_{\mathrm{nRMSE}}$ of 40.2%. Therefore, a choice of the original parameters being 2 layers and 110 units is somewhat justified. It is notable that the model's performance increases for 2 layers, but then deteriorates for 4 layers, which is most obvious in *S2S-Attn-2x256* and *S2S-Attn-4x256*. This is possibly explained by the deeper model overfitting the dataset. Figures 5.3 and 5.4 illustrate this by plotting training and validation curves of the two architectures. In the figures, $y$ axis represents the nRMSE value, which is used to monitor the validation performance.

| Model | nRMSE | | nME | | CRPS | | $S_{\text{nRMSE}}$ | | $S_{\text{CRPS}}$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| | *Val* | *Test* | *Val* | *Test* | *Val* | *Test* | *Val* | *Test* | *Val* | *Test* |
| *S2S-Attn-1x64* | 0.088 | 0.084 | 0.050 | 0.047 | 1.205 | 1.209 | 0.381 | 0.347 | 0.573 | 0.524 |
| *S2S-Attn-1x128* | 0.086 | 0.079 | 0.049 | 0.044 | 1.181 | 1.108 | 0.395 | 0.379 | 0.582 | 0.564 |
| *S2S-Attn-1x256* | 0.087 | 0.078 | 0.049 | 0.043 | 1.207 | 1.127 | 0.387 | 0.387 | 0.572 | 0.556 |
| *S2S-Attn-2x64* | 0.092 | 0.085 | 0.052 | 0.048 | 1.298 | 1.250 | 0.353 | 0.332 | 0.540 | 0.508 |
| *S2S-Attn-2x128* | 0.089 | 0.077 | 0.050 | 0.043 | 1.241 | 1.063 | 0.373 | 0.402 | 0.560 | 0.581 |
| *S2S-Attn-2x256* | **0.085** | **0.075** | **0.048** | **0.042** | **1.172** | **1.032** | **0.397** | **0.411** | **0.585** | **0.594** |
| *S2S-Attn-4x64* | 0.089 | 0.082 | 0.050 | 0.046 | 1.232 | 1.164 | 0.375 | 0.361 | 0.564 | 0.542 |
| *S2S-Attn-4x128* | 0.089 | 0.083 | 0.050 | 0.046 | 1.210 | 1.143 | 0.373 | 0.349 | 0.571 | 0.550 |
| *S2S-Attn-4x256* | 0.091 | 0.083 | 0.051 | 0.046 | 1.286 | 1.184 | 0.357 | 0.350 | 0.544 | 0.533 |

Table 5.6: Model Parameter Tuning Results.

And $x$ axis is the number of epochs the model trained for. Considering that the model in Figure 5.4, *S2S-Attn-4x256*, exhibits earlier overfitting, it can be concluded that the forecasts in this experiment setting will not benefit from deeper architectures.

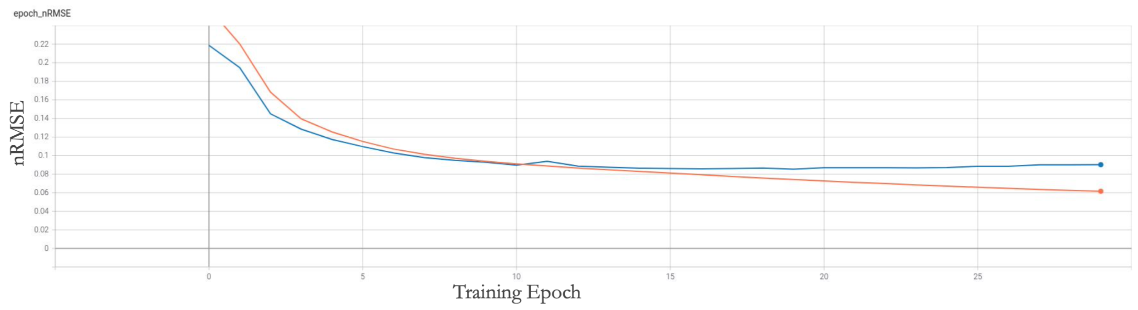

Figure 5.3: Train and Validation Curves for *S2S-Attn-2x256*.

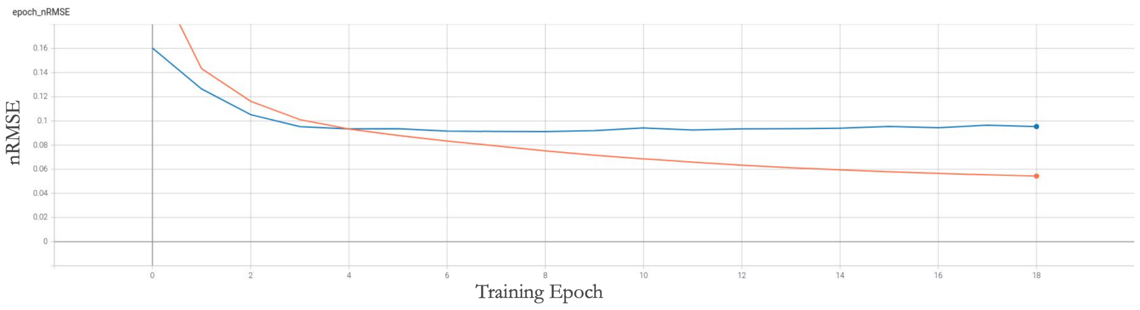

Figure 5.4: Train and Validation Curves for *S2S-Attn-4x256*.

# Chapter 6

# Conclusions & Future Work

## 6.1 Conclusions

This thesis explores the application of sequence to sequence models with attention to multivariate time series forecasting. To analyze the suitability of these techniques, a sequence attention model was proposed, based on the sequence to sequence model by Sutskever [2] and attention mechanism by Bahdanau [3]. Then, a test comparing this model to standard architectures was conducted. Further, the training design choices were inspected. An investigation of whether the model's performance is affected by the dataset was studied. The proposed attention model was compared to the classic sequence to sequence attention for validation. Finally, experiments were conducted on the alternative choice of recurrent units.

The results demonstrated that the sequence models are well suited for multivariate time series forecasting due to the decoupling of the encoder and decoder functions as well as the decoder's recursive nature. Encoder-decoder models appeared to benefit from a high resolution data and probabilistic targets in the form of binned probability distribution for photovoltaic (PV) generation forecasting. Furthermore, providing the next day numerical weather forecast to the models significantly positively affected the performance by providing more than 15% gain in nRMSE skill score. Training the proposed model with teacher forcing further improved the results. The proposed attention model achieved the best performance across the conducted experiments.

On one specific dataset, it achieved an nRMSE skill score of 48.1%. And on a $k$-fold cross-validation experiment, with the datasets generated under different conditions, it achieved an nRMSE skill of 36.2%. The discrepancy between the results is attributed to the differences in the datasets. Overall, it was concluded that sequence to sequence architectures with attention can gain improvements in multivariate time series forecasting.

In conclusion, the following contributions were made in the experimental process. Objective 1, which comprised an analysis of applying sequence models to forecasting, was met by applying several well-known sequence to sequence models to photovoltaic power generation forecasting. The second objective was to add an exploration of an attention mechanism. To complete the goals of objective 2, an analysis of a proposed attention architecture as well as a review of a classic attention model were studied. This showed that multivariate time series forecasting can potentially improve by applying these models.

For further analysis of the training design choices, as specified in objective 3, a $k$-fold cross-validation experiment was conducted, studying the effects of teacher forcing and the input sliding window data. As per objective 4, to validate the proposed model's performance, the model was tested against a classic sequence to sequence attention architecture. The use of a different recurrent unit was analyzed to potentially reduce the computational resources without harming the performance. A model parameter tuning experiment helped in understanding of architecture dynamics and determined a direction of potential performance improvement. In summary, the objectives set for this work were met. The results of this research prove that sequence models with attention are applicable to multivariate time series forecasting in the energy sector.

## 6.2 Future Work

Overall, this thesis demonstrated that deep learning natural language (NLP) techniques are transferable to multivariate time series forecasting. Moreover, attention was proven to aid feature extraction. Therefore, to further explore NLP methods, a self-attention based architecture will be applied to multivariate time series forecasting. This is a part of an ongoing project, which is focused on taking inspiration from the Transformer and other advanced NLP models. Along the same direction, it is suggested to employ pre-training and fine-tuning techniques to reduce the computational capacity requirements. Currently, one model is trained per dataset. However, the model can be pre-trained on one large dataset, and afterward fine-tuned on a previously unseen dataset from the same domain, to forecast for different individual scenarios. Finally, this project can extend to other domains, such as load forecasting.

# Bibliography

[1] D. van der Meer, J. Widén, and J. Munkhammar, "Review on probabilistic forecasting of photovoltaic power production and electricity consumption," *Renewable and Sustainable Energy Reviews*, vol. 81, pp. 1484–1512, 2018, ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2017.05.212. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032117308523.

[2] I. Sutskever, O. Vinyals, and Q. V. Le. (2014). Sequence to sequence learning with neural networks. arXiv: 1409.3215.

[3] D. Bahdanau, K. Cho, and Y. Bengio. (2014). Neural machine translation by jointly learning to align and translate. arXiv: 1409.0473.

[4] K. Benidis, S. S. Rangapuram, V. Flunkert, B. Wang, D. Maddix, C. Turkmen, J. Gasthaus, M. Bohlke-Schneider, D. Salinas, L. Stella, L. Callot, and T. Januschowski. (2020). Neural forecasting: Introduction and literature overview. arXiv: 2004.10240.

[5] A. Gasparin, S. Lukovic, and C. Alippi. (2019). Deep learning for time series forecasting: The electric load case. arXiv: 1907.09207.

[6] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*, 3rd ed. OTexts: Melbourne, Australia, 2021. [Online]. Available: https://otexts.com/fpp3/.

[7] J. G. De Gooijer and R. J. Hyndman, "25 years of time series forecasting," *International Journal of Forecasting*, vol. 22, no. 3, pp. 443–473, 2006, Twenty five years of forecasting, ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2006.01.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207006000021.

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016, http://www.deeplearningbook.org.

[9] J. Antonanzas, N. Osorio, R. Escobar, R. Urraca, F. M. de Pison, and F. Antonanzas-Torres, "Review of photovoltaic power forecasting," *Solar Energy*, vol. 136, pp. 78–111, 2016, ISSN: 0038-092X. DOI: https://doi.org/10.1016/j.solener.2016.06.069. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0038092X1630250X.

[10] J. Dumas and B. Cornélusse. (2018). Classification of load forecasting studies by forecasting problem to select load forecasting techniques and methodologies. arXiv: 1901.05052.

[11] T. Hong and S. Fan, "Probabilistic electric load forecasting: A tutorial review," *International Journal of Forecasting*, vol. 32, no. 3, pp. 914–938, 2016, ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2015.11.011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207015001508.

[12] U. K. Das, K. S. Tey, M. Seyedmahmoudian, S. Mekhilef, M. Y. I. Idris, W. Van Deventer, B. Horan, and A. Stojcevski, "Forecasting of photovoltaic power generation and model optimization: A review," *Renewable and Sustainable Energy Reviews*, vol. 81, pp. 912–928, 2018, ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2017.08.017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032117311620.

[13] C. Voyant, G. Notton, S. Kalogirou, M.-L. Nivet, C. Paoli, F. Motte, and A. Fouilloy, "Machine learning methods for solar radiation forecasting: A review," *Renewable Energy*, vol. 105, pp. 569–582, 2017, ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2016.12.095. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960148116311648.

[14] S. Sobri, S. Koohi-Kamali, and N. A. Rahim, "Solar photovoltaic generation forecasting methods: A review," *Energy Conversion and Management*, vol. 156, pp. 459–497, 2018, ISSN: 0196-8904. DOI: https://doi.org/10.1016/j.enconman.2017.11.019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0196890417310622.

[15] D. Lee and K. Kim, "Recurrent neural network-based hourly prediction of photovoltaic power output using meteorological information," *Energies*, vol. 12, no. 2, 2019, ISSN: 1996-1073. [Online]. Available: https://www.mdpi.com/1996-1073/12/2/215.

[16] M. Pierro, F. Bucci, M. De Felice, E. Maggioni, D. Moser, A. Perotto, F. Spada, and C. Cornaro, "Multi-model ensemble for day ahead prediction of photovoltaic power generation," *Solar Energy*, vol. 134, pp. 132–146, 2016, ISSN: 0038-092X. DOI: https://doi.org/10.1016/j.solener.2016.04.040. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0038092X16300731.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html.

[18] S. Ioffe and C. Szegedy. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv: 1502.03167.

[19] J. L. Ba, J. R. Kiros, and G. E. Hinton. (2016). Layer normalization. arXiv: 1607.06450.

[20] K. He, X. Zhang, S. Ren, and J. Sun. (2015). Deep residual learning for image recognition. arXiv: 1512.03385.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. (2017). Attention is all you need. arXiv: 1706.03762.

[22] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv: 1406.1078.

[23] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. (2019). Language models are unsupervised multitask learners, [Online]. Available: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, 1735–1780, Nov. 1997, ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735.

[25] M.-T. Luong, H. Pham, and C. D. Manning. (2015). Effective approaches to attention-based neural machine translation. arXiv: 1508.04025.

[26] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv: 1810.04805.

[27] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. (2018). Improving language understanding by generative pre-training, [Online]. Available: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

[28] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever. (2019). Deep double descent: Where bigger models and more data hurt. arXiv: 1912.02292.

[29] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[30] E. Kharlova, D. May, and P. Musilek, "Forecasting photovoltaic power production using a deep learning sequence to sequence model with attention," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–7. DOI: 10.1109/IJCNN48605.2020.9207573.

[31] A. Bracale, G. Carpinelli, P. De Falco, R. Rizzo, and A. Russo, "New advanced method and cost-based indices applied to probabilistic forecasting of photovoltaic generation," *Journal of Renewable and Sustainable Energy*, vol. 8, no. 2, p. 023 505, 2016. DOI: 10.1063/1.4946798.

# Appendix A: Additional tables and figures

## A.1 k-fold Cross-Validaion

Please see Table A.1 for the averaged nRMSE, nME, CRPS, $S_{\mathrm{nRMSE}}$, and $S_{\mathrm{CRPS}}$ for the k-fold cross-validation experiment. The averaged results of the PV historical signal ablation study are in Table A.2.

## A.2 Forecasts

Presented here are additional figures of forecasts produced by the proposed model *S2S-Attn* for the scenario with shifted NWP sample and teacher forcing. Figures A.1 and A.2 illustrate a case where the model failed to grasp the PV distribution well. Figures A.3 and A.4 show an example of PV generation presumably in winter when there is little generation due to low levels of solar irradiance. And figures A.5 and A.6 demonstrate an example of a day when there are high levels of solar irradiance.

| | nRMSE | | nME | | CRPS | | $S_{\mathrm{nRMSE}}$ | | $S_{\mathrm{CRPS}}$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | *Val* | *Test* | *Val* | *Test* | *Val* | *Test* | *Val* | *Test* | *Val* | *Test* |
| *Persistence* | 0.133 | 0.135 | 0.072 | 0.073 | 2.612 | 2.680 | - | - | - | - |
| | SW/no TF | | | | | | | | | |
| *LSTM* | 0.116 | 0.120 | 0.067 | 0.069 | 1.741 | 1.841 | 0.126 | 0.112 | 0.330 | 0.315 |
| *S2S* | 0.112 | 0.114 | 0.065 | 0.066 | 1.729 | 1.756 | 0.154 | 0.158 | 0.336 | 0.343 |
| *S2S-Attn* | 0.112 | 0.112 | 0.065 | 0.065 | 1.689 | 1.662 | 0.156 | 0.171 | 0.351 | 0.381 |
| | SW/TF | | | | | | | | | |
| *LSTM* | - | - | - | - | - | - | - | - | - | - |
| *S2S* | 0.113 | 0.113 | 0.065 | 0.065 | 1.745 | 1.723 | 0.149 | 0.167 | 0.330 | 0.357 |
| *S2S-Attn* | 0.110 | 0.114 | 0.064 | 0.066 | 1.699 | 1.765 | 0.164 | 0.154 | 0.346 | 0.342 |
| | Shifted SW/no TF | | | | | | | | | |
| *LSTM* | 0.094 | 0.095 | 0.053 | 0.054 | 1.322 | 1.379 | 0.289 | 0.299 | 0.491 | 0.489 |
| *S2S* | **0.091** | 0.089 | **0.051** | 0.050 | **1.319** | 1.298 | **0.312** | 0.345 | **0.493** | 0.517 |
| *S2S-Attn* | 0.092 | 0.087 | 0.052 | 0.049 | 1.335 | 1.240 | 0.303 | 0.356 | 0.486 | 0.539 |
| | Shifted SW/TF | | | | | | | | | |
| *LSTM* | - | - | - | - | - | - | - | - | - | - |
| *S2S* | **0.091** | 0.091 | 0.052 | 0.051 | 1.332 | 1.342 | 0.309 | 0.331 | 0.488 | 0.501 |
| *S2S-Attn* | 0.092 | **0.086** | 0.052 | **0.048** | 1.333 | **1.228** | 0.303 | **0.362** | 0.487 | **0.543** |

Table A.1: Complete k-fold Experiment Results.

| **Model** | nRMSE | | nME | | CRPS | | $S_{\text{nRMSE}}$ | | $S_{\text{CRPS}}$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| | *Val* | *Test* | *Val* | *Test* | *Val* | *Test* | *Val* | *Test* | *Val* | *Test* |
| *Persistence* | 0.133 | 0.135 | 0.072 | 0.073 | 2.612 | 2.680 | - | - | - | - |
| | Shifted SW/no TF | | | | | | | | | |
| *LSTM* | 0.094 | 0.095 | 0.053 | 0.053 | 1.365 | 1.377 | 0.285 | 0.299 | 0.474 | 0.486 |
| *S2S* | **0.093** | 0.093 | 0.053 | 0.053 | **1.363** | 1.373 | **0.292** | 0.313 | **0.475** | 0.490 |
| *S2S-Attn* | 0.096 | 0.092 | 0.054 | 0.051 | 1.431 | 1.356 | 0.274 | 0.317 | 0.449 | 0.495 |
| | Shifted SW/TF | | | | | | | | | |
| *LSTM* | - | - | - | - | - | - | - | - | - | - |
| *S2S* | 0.094 | 0.097 | 0.053 | 0.054 | 1.425 | 1.510 | 0.286 | 0.282 | 0.451 | 0.440 |
| *S2S-Attn* | 0.097 | **0.090** | 0.054 | **0.050** | 1.441 | **1.322** | 0.269 | **0.334** | 0.445 | **0.510** |

Table A.2: Complete ablation Experiment Results.

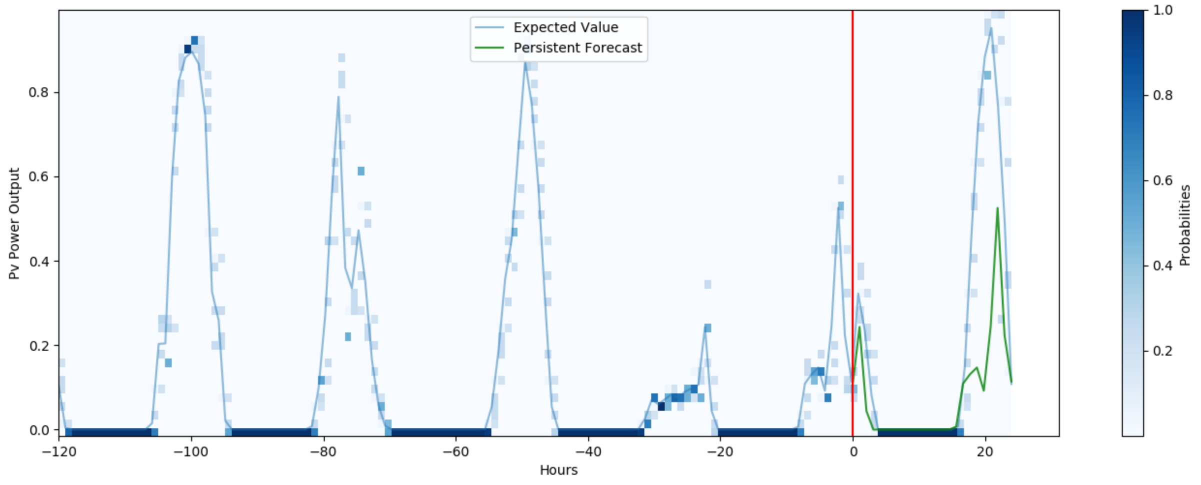

Figure A.1: Actual PV Generation PDF Signal.

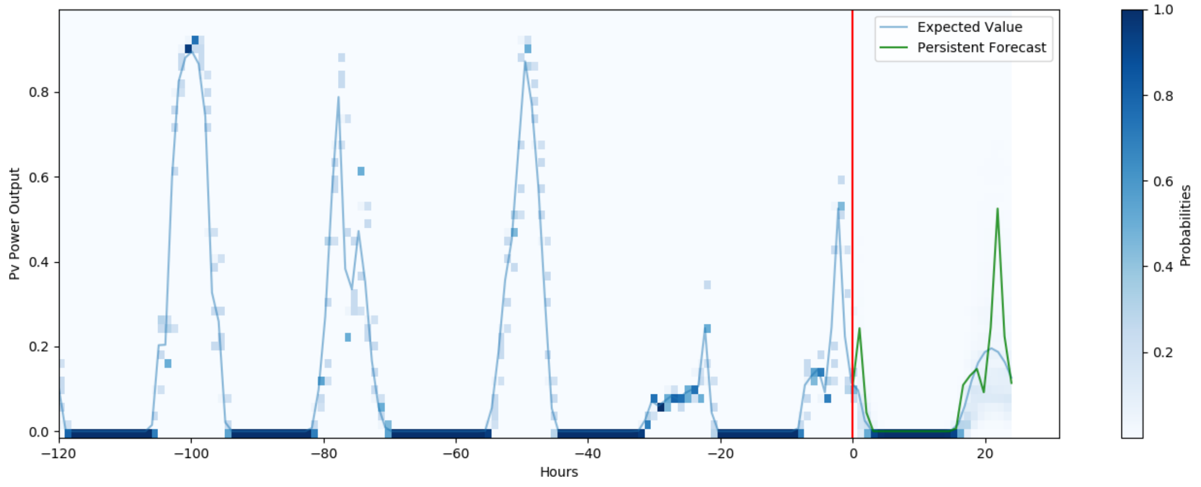

Figure A.2: PV Generation History and Forecast PDF.

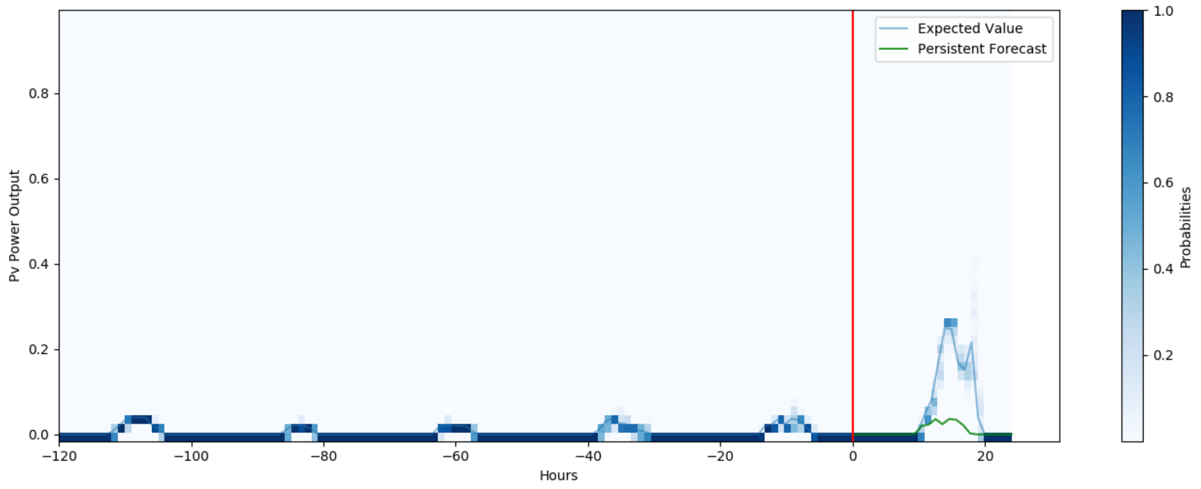

Figure A.3: Actual PV Generation PDF Signal.

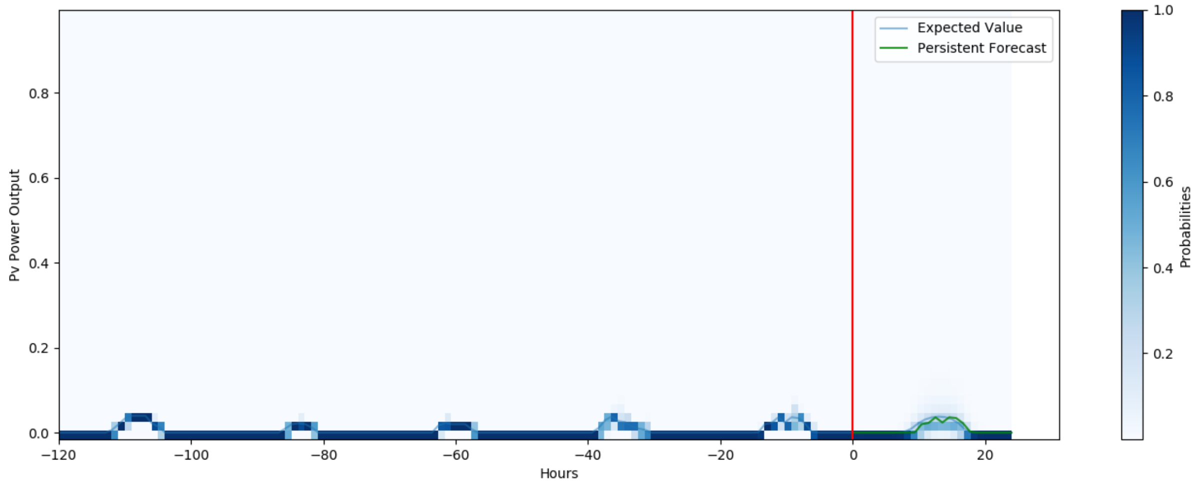

Figure A.4: PV Generation History and Forecast PDF.

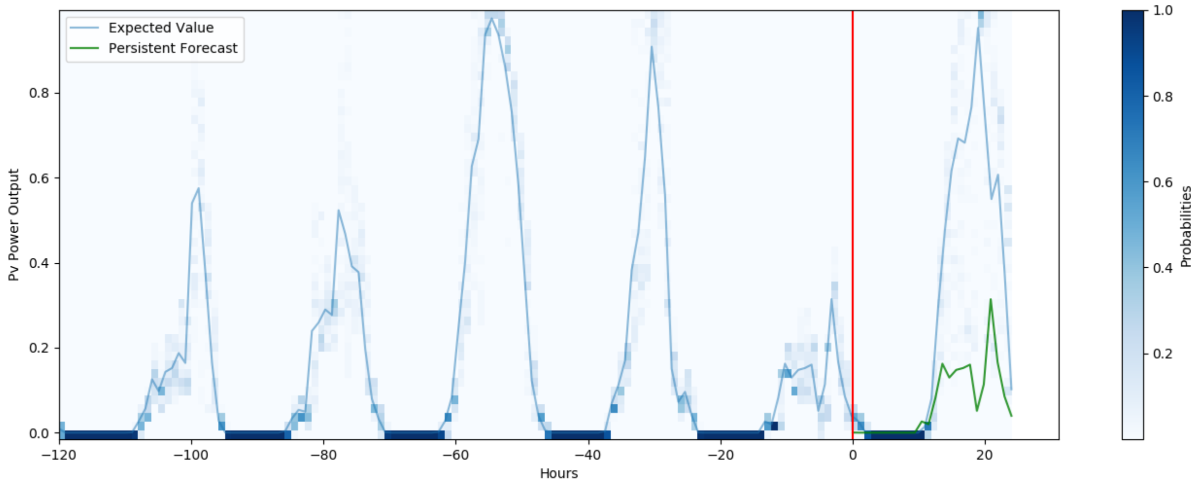

Figure A.5: Actual PV Generation PDF Signal.

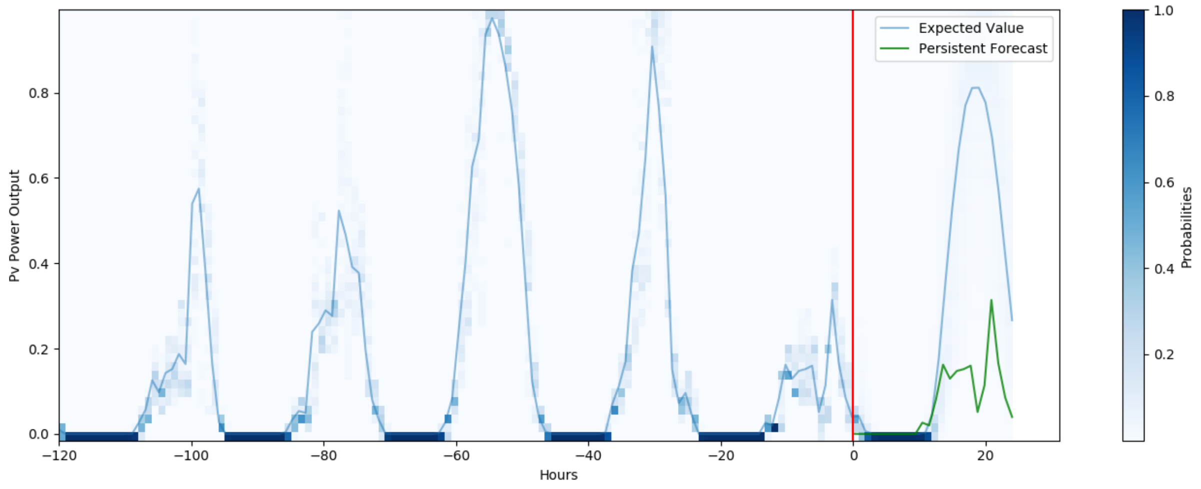

Figure A.6: PV Generation History and Forecast PDF.