TIME SERIES FORECASTING ON SOLAR RADIATION USING DEEP LEARNING

(GÜNEŞ IŞINIMI ÜZERİNDE DERİN ÖĞRENME KULLANARAK ZAMAN SERİLERİ TAHMİNİ)

by

Murat Cihan SORKUN, B.S.

Thesis

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

in the

INSTITUTE OF SCIENCE AND ENGINEERING

of

GALATASARAY UNIVERSITY

Jan 2018

This is to certify that the thesis entitled

TIME SERIES FORECASTING ON SOLAR RADIATION USING DEEP LEARNING

prepared by **Murat Cihan SORKUN** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering** at the **Galatasaray University** is approved by the

Examining Committee:

Assoc. Prof. Dr. Özlem Durmaz İncel (Supervisor)
Department of Computer Engineering
Galatasaray University ------------------------

Asst. Prof. Dr. Günce Keziban Orman
Department of Computer Engineering
Galatasaray University ------------------------

Prof. Dr. Şule Gündüz Öğüdücü
Department of Computer Engineering
Istanbul Technical University -------------------------

Date: ------------------------

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to the following people:

- my supervisors, Dr. Christophe Paoli and Dr. Özlem Durmaz İncel, for their valuable guidance;
- my colleague, Dr. Esra Çınar, for her encouragement and support;
- the company I work for, Idea Technology Solutions, for academic support;
- the University of Corsica, UMR CNRS 6134 Lab, for providing the data used in this study;
- my wife, Elif, for her unconditional love and understanding.

Jan 2018
Murat Cihan SORKUN

TABLE OF CONTENTS

LIST OF SYMBOLS

ANN	**:** Artificial Neural Network
AR	**:** Auto Regressive
ARIMA	**:** Auto Regressive Integrated Mean Average
BPTT	**:** Backpropagation Through Time
CNN	**:** Convolutional Neural Network
DBN	**:** Deep Belief Network
DL	**:** Deep Learning
FNN	**:** Feed Forward Neural Network
GRU	**:** Gated Recurrent Unit
LSTM	**:** Long Short-Term Memory Unit
MA	**:** Mean Avarage
MAE	**:** Mean Absolute Error
MAD	**:** Mean Absolute Deviation
MFE	**:** Mean Forecast Error
MLP	**:** Multi Layer Perceptron
MSE	**:** Mean Squared Error
NAR	**:** Nonlinear Autoregressive
NRMSE	**:** Normalized Root Mean Squared Error
RBM	**:** Restricted Boltzmann Machine
ReLU	**:** Rectified Linear Unit
RMSE	**:** Root Mean Squared Error
RNN	**:** Recurrent Neural Network
SMSE	**:** Signed Mean Squared Error
SVM	**:** Support Vector Machine

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Electricity can be produced from fossil fuels, from nuclear energy, from bio-fuels or from renewable energy resources. As a matter of fact, energy suppliers and managers face the energy management problem. Concerning electricity generation based on solar radiation, it is very important to know precisely the amount of electricity available for the different sources and at different horizons: minutes, hours and days. Depending on the horizon, two main classes of methods can be used to forecast the solar radiation: statistical time series forecasting methods for short to midterm horizons and numerical weather prediction methods for medium to long-term horizons. In this thesis we focus on statistical time series forecasting methods. The aim of this study is to assess if deep learning can be suitable and competitive for solar radiation data time series forecasting. In this context, Recurrent Neural Network variations, namely Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models are used for time series forecasting on solar radiation data. With an experimental approach, the performance of single layer and two layered LSTM and GRU models are investigated. By hyper-parameter tuning optimal parameters are found in order to construct best model that fits the global solar radiation data. In further experiments, the data are divided into the seasons. Seasonal performance of constructed RNN models and other machine learning methods are compared and a hybrid model is proposed according to the experimental results. Finally, the effect of additional meteorological parameters on solar radiation forecasting is investigated. The results show that the LSTM and GRU models can be suitable and competitive for 1 hour horizon time series forecasting on the solar radiation data. Experiments showed that hybrid approach and additional meteorological parameters improve the performance of model.

ÖZET

Elektrik, fosil yakıtlardan, nükleer enerjiden, biyoyakıtlardan veya yenilenebilir enerji kaynaklarından üretilebilir. Nitekim enerji tedarikçileri ve yöneticileri, enerji yönetimi problemiyle karşılaşmaktadır. Güneş ışığı kaynaklı elektrik üreten sistemlerde, dakika, saat ve gün gibi farklı periyotlarda üretilecek elektriğin miktarını tam olarak bilmek oldukça önemlidir. Güneş ışınımı tahmininde, periyoda bağlı olarak iki ana yöntem kullanılmaktadır. Bu yöntemler kısa ve orta dönemlik periyotlar için istatiksel zaman serileri tahmini ve orta ve uzun dönemlik periyotlar için ise sayısal hava durumu tahminidir. Bu çalışmada istatiksel zaman serileri tahmini yöntemleri odaklanılmıştır. Çalışmanın hedefi, güneş ışınımı verisi üzerinde derin öğrenmeile zaman serileri tahmini yöntemlerinin uygunluğu ve rekabet edebilirliğini araştırmaktır. Bu kapsamda, Yenilemeli Sinir Ağı varyasyonu olan Long Short-Term Memory (LSTM) ve Gated Recurrent Unit (GRU) modelleri kullanılarak güneş ışınımı üzerinde zaman serileri tahmini yapılmıştır. Parametreler optimize edilerek güneş ışınımı verisini en iyi temsil eden modeli oluşturacak değerler bulunmuştur. Kurulan RNN modellerinin ve diğer makine öğrenmesi yöntemlerinin mevsimsel performansları karşılaştırılmış ve deneysel sonuçlara göre bir hibrit model önerilmiştir. Son olarak, güneş ışınımı tahmini üzerine ilave meteorolojik parametrelerin etkisi araştırılmıştır. Sonuçlar, LSTM ve GRU modellerinin güneş ışınımı verileri üzerinde 1 saatlik ufuktaki zaman serileri tahmini için uygun ve rekabet edebilir olduğunu göstermektedir. Deneyler, hibrit yaklaşımın ve ilave meteorolojik parametrelerin modelin performansını iyileştirdiğini göstermiştir.

1. INTRODUCTION

While the energy demand on the earth increases day by day, the resources on the earth are inadequate to meet this demand. The reserves of non-renewable energy sources such as coal, oil and natural gas are steadily declining. That is why; the trend towards renewable energy sources in the global sense has emerged. When it can be used efficiently, solar energy has the highest potential among these resources. The development of technology and the reduction of the infrastructure costs required to make use of solar energy are reinforcing the trend towards this area. Depending on this trend, it is becoming more and more important to forecast the solar radiation precisely.

The handicap of solar energy based systems is that the amount of solar radiation cannot be easily estimated since it depends on many variables. At this point time series forecasting can play a key role in short horizon prediction, such as a couple of hours. There are a number of models being conducted to forecast solar radiation with time series, such as combination of Autoregressive and Dynamical System models (Huang et al., 2013), autoregressive integrated moving average (ARIMA) model (Yang et al., 2012), Nonlinar Autoregresive (NAR) neural network (Benmouiza & Cheknane, 2013) and Multi-layer Perceptrons (MLP) (Mihalakakou et al., 2000; Paoli et al., 2010). In (Reikard, 2009), Regression, ARIMA, neural network and unobserved component models are compared.

The disappearance of the hardware boundaries and the production of large volumes of data have led to the widespread use of deep learning models. In the literature, there are many deep learning models applied for time series forecasting. These include: Deep Belief Networks (DBN) (Kuremoto et. al, 2014; Qiu et. al., 2014; Lv et al., 2015), Stacked Auto Encoders (Lv et al., 2015; Bao et al., 2017) and Long Short-Term Memory (LSTM) units (Fischer & Krauß, 2017; Hsu, 2017; Bao et al., 2017).

In this study, two Recurrent Neural Network (RNN) variations, namely LSTM and Gated Recurrent Unit (GRU) models are used for time series forecasting on solar radiation data. With an experimental approach, the performance of single layer and two layer LSTM and GRU models are investigated. By hyper-parameter tuning optimal parameters are found in order to construct best models that fit the global solar radiation data. In further experiments, the data are divided into the seasons. Seasonal performance of constructed RNN models and other machine learning methods are compared and a hybrid model is proposed according to the experimental results. Finally, the effect of additional meteorological parameters on solar radiation forecasting is investigated.

This thesis exposes the performance of recurrent neural network models on solar radiation forecast. The thesis has 3 main contributions. First, we proposed single and double layered LSTM and GRU models with optimized hyper-parameters. Second, we proposed a hybrid model that combines seasonally predictive models. Finally, we proposed a multivariate forecast model using a combination of different meteorological data.

This study is organized as follows. Section 2 introduces time series forecasting concepts. Different forecasting methods and performance metrics are described in this section. Deep Learning methods are introduced in Section 3. Section 4 contains univariate and multivariate experiments using Deep Learning methods on time series forecasting. Finally, Section 5 concludes the findings and includes some information towards future works.

2. TIME SERIES FORCASTING

2.1 Introduction

Time series are sequential data that are measured at certain intervals with respect to any process. The intervals used in the time series may be of different sizes, provided that they are equally divided. They are usually measured at intervals such as hourly, daily, monthly, and yearly. The annual population, the daily number of passengers on the metro and the hourly exchange rate are examples of time series. In the time series, records must be ordered chronologically. Each record may contain information about one or more features. Univariate term is used for time series data containing single information, and multivariate is used for data containing more than one information.

In this context, time series forecasting can be defined as the prediction of the future data using time series data of the past. Time series data are used for creating model by different methods. The process of creating the model by formulating the data is called time series analysis. Forecasting is carried out through these models. In this section, concepts of time series, prediction models and performance measurement methods are explained.

2.2 Time Series Concepts

In order to better understand the characteristics of the time series, some concepts are used. By analyzing the behavior of the time series through these concepts, it is possible to select appropriate preprocessing methods and the model. This section examines the concepts of time series.

2.2.1 Trend

The long-term increase or decrease in the time series is called trend. Increasing trends are termed uptrend, decreasing trends are termed downtrend. These changes can occur linearly or non-linearly. According to the trend tendency, different analyzes are made and trend-related models are formed (Zhang, 2001; Wu & Chang, 2012). Figure 2.1 shows a downtrend example.

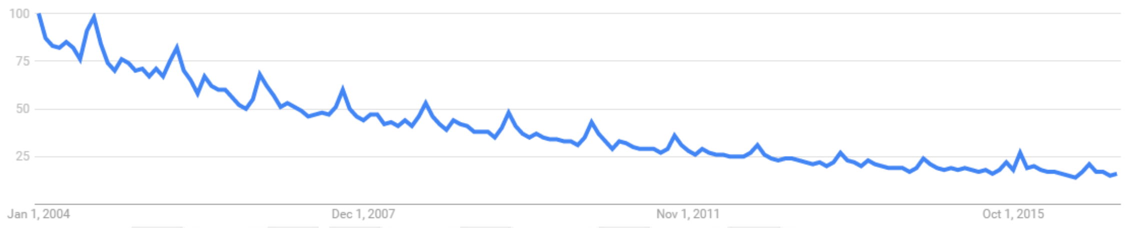

Figure 2.1: Downtrend Example

2.2.2 Seasonality

In the time series, as a result of seasonal factors, the short repeated and incremental increases and decreases are called seasonality. Although it is generally used for four seasons, which are quarters of the year, seasonality can be found in different periods such as annual, monthly and daily. In the seasonal time series, the length of the period is fixed. Figure 2.2 shows a seasonal time series example.

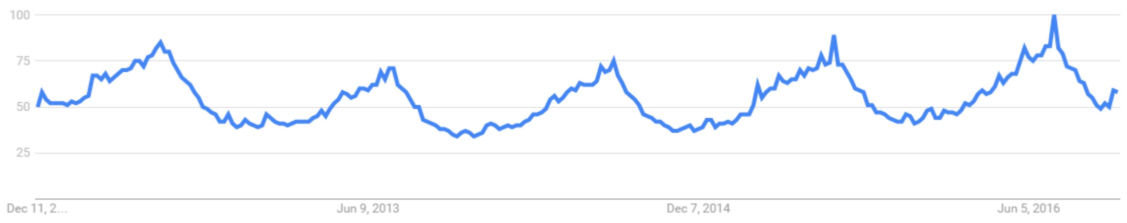

Figure 2.2: Seasonality Example

2.2.3 Stationarity

It is important to know that the time series is stationary in order to estimate a stable future forecasting. Time series are considered stationary if they have equal probability distributions at every point in time (Jeffrey, 2015). This means that statistical data such as mean and variance are not time dependent. Figure 2.3 shows a stationary time series example.

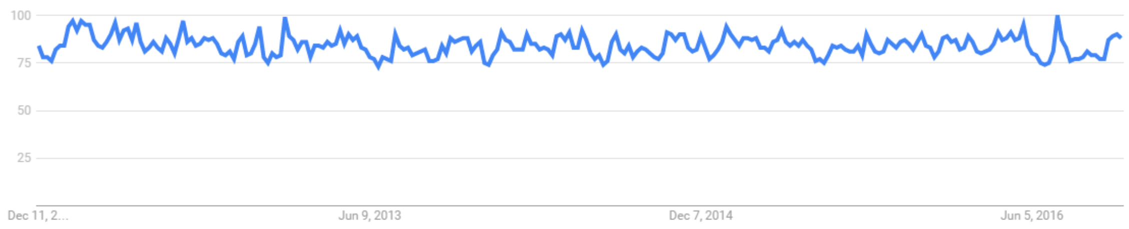

Figure 2.3: Stationarity Example

2.3 Forecasting Methods

A lot of research has been done on the time series forecasting and many studies are still ongoing. In these studies, many different methods have been used for forecasting. Different methods for time series in different characteristics can be successful. In this section, commonly used forecasting methods in the literature are introduced.

2.3.1 Naïve Method

The simplest and cost-effective method that can be used in time series is the naive approach. This method uses the last observed data as the next prediction. Naïve method just ignores the historical data as it uses only the last instance. Although it can achieve successful results over low variance time series, it does not yield reliable results. Generally, it is used as a reference to compare with other methods (Stergiou & Chirtou, 1996; Balkin & Ord, 2000; Conejo et al., 2002).

2.3.2 Linear Models

Linear models try to predict future data with a linear function using past time series data. The most commonly used linear methods are Auto Regressive (AR), Mean Average (MA), and hybrid methods. In this section, traditional linear time series models using these methods are introduced.

ARMA is a linear model that combines AR and MA models. It is used to estimate stationary and univariate time series. In an AR model the future value of a variable is assumed to be a linear combination of p past observations and a random error together with a constant (Adhikari & Agrawal, 2013). The MA model predicts the future by taking the average of the nearest number of data from the past data.

The ARIMA model is a more generalized version of the ARMA model. The ARIMA model can also be applied on non-stationary time series with differencing method. It is the most widely used linear model and has been used as a comparison reference in many studies (More & Deo, 2003; Vengertsev, 2014).

The Box-Jenkins method is an approach applied to ARMA and ARIMA linear models. The goal is to choose the best fit model for the time series in 3 steps. These 3 stages are model selection, parametric estimation and model checking.

2.3.3 Nonlinear Models

When time series show a linear structure, linear prediction models can be used. However, many time series have a nonlinear structure. Many nonlinear models have been proposed in the literature in order to come up to the limitations of linear models. In this section, nonlinear models used in time series forecasting are examined.

2.3.3.1 Support Vector Machines

Support Vector Machine (SVM) is a supervised learning algorithm that has been successfully implemented on classification, approximation, and time series forecasting problems. It is possible to separate two groups by drawing a border between two groups in the plane. The place where this boundary can be drawn is that the two groups must be the most distant and apart from each other. The SVM determines how this boundary is drawn. SVM models generated for multiple inputs can produce nonlinear solutions. SVM is used in many studies that forecast time series. Qiu et al. (2014) proposed ensemble deep learning method consist of deep belief network and support vector regression for time series forecast on three different electricity load demand datasets. Vengertsev (2014) applied support vector regression to compare their proposed deep neural network model.

2.3.3.2 Artificial Neural Networks

Artificial neural network (ANN) is a model inspired from the architecture of the human brain. ANN provides complex quadratic and polynomial functions to be easily represented. It is one of the most commonly used models for predicting nonlinear time series. Feed forward neural networks (FNN) are commonly used on time series forecasting. The FNN architecture consists of three layers, the input layer, the hidden layer and the output layer. More than one hidden layer can be found. In the input layer, there are as many nodes as the number of input features. In the output layer, there are as many nodes as the number of outputs. In the hidden layer, there can be different number of nodes according to the model. Each node in each layer contains the connection called weight to the nodes in the next layer. These weights are calculated by backpropagation method. There are many studies in the literature on time series forecasting using ANN. Allende et al. (2002) reviewed ANN implementations on time series forecasting and applied ANN on two different datasets. Zhang and Qi (2005) applied ANN to forecast seasonal and trend time series.

2.3.3.2 Random Forest

Another structure that is often used in forecasting time series is decision trees. As a result of the training of decision trees, the learned information is modeled on a tree. Random Forest is an ensemble decision tree method in which multiple prediction trees are constructed with different random variables (Breiman, 2001). Random Forest is a fast and extremely resistant method to over fitting problem. The system can be created using as many trees as desired. There are many time series forecasting studies in the literature using Random Forest. Juban et al. (2007) applied Random Forest model to forecast short-term wind power. Dudek (2015) implemented Random Forest model for short-term electricity load forecasting.

2.3.4 Conclusion

In the literature, many different methods have been proposed for estimating time series. Linear or nonlinear models can be used depending on the time series used. In addition, studies using linear and nonlinear models as hybrids have been carried out (Zhang, 2001).

2.4 Performance Measurement

Performance measurement metrics are used to measure and compare the success of the predictions made. There are many performance measures suggested in the literature. These measurements have advantages and disadvantages. This section describes the most commonly used performance measures.

2.4.1 Mean Absolute Error

Mean absolute error (MAE) is one of the most frequently used performance measures in time series forecasting. The measurement, also termed as Mean absolute deviation (MAD), calculates the absolute mean deviation between the estimated value and the actual data. Calculation of MAE is as shown below. MAE does not distinguish between negative and positive error. Extreme errors are not penalized. The MAE value should be kept as small as possible so that the estimates can be considered successful. The equation used to calculate the MAE is as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |e_i| \tag{1}$$

where n is the number of instances and e is the error of each forecast.

2.4.2 Mean Absolute Percentage Error

Mean absolute percentage error (MAPE) calculates the absolute percent difference between the estimated value and the actual data. Calculation of MAPE is as shown below. MAPE does not distinguish between negative and positive error. Extreme errors are not penalized. The MAPE value should be kept as small as possible so that the estimates can be considered successful. The equation used to calculate the MAPE is as follows:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n} |\frac{e_i}{y_i}| \times 100 \tag{2}$$

where n is the number of instances, y is the actual value of instance and e is the error of each forecast.

2.4.3 Mean Squared Error

Mean squared error (MSE) calculates the average squared deviation between the estimated value and the actual data. Calculation of MSE is as shown below. MSE does not distinguish between negative and positive error. Extreme errors are penalized by MSE. The MSE value should be kept as small as possible so that the estimates can be considered successful. The equation used to calculate the MSE is as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n} {e_i}^2 \tag{3}$$

where n is the number of instances and e is the error of each forecast.

2.4.4 Root Mean Squared Error

Root mean squared error (RMSE) calculates the square root of average squared deviation between the estimated value and the actual data. Calculation of RMSE is as shown below. RMSE does not distinguish between negative and positive error. Extreme errors are penalized by RMSE. The RMSE value should be kept as small as possible so that the estimates can be considered successful. The equation used to calculate the RMSE is as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} {e_i}^2} \tag{4}$$

where n is the number of instances and e is the error of each forecast.

2.4.5 Normalized Root Mean Squared Error

Normalized root mean squared error (NRMSE) is a normalized version of RMSE. NRMSE can be calculated in several different ways. The most common of these is dividing the result obtained by RMSE by the average observed value. Calculation of NRMSE is as shown below. NRMSE does not distinguish between negative and positive error. Extreme errors are penalized by NRMSE. The NRMSE value should be kept as small as possible so that the estimates can be considered successful. The equation used to calculate the NRMSE is as follows:

$$NRMSE = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n} {e_i}^2}}{\frac{1}{n}\sum_{i=1}^{n} y_i} \tag{5}$$

where n is the number of instances, y is the actual value of instance and e is the error of each forecast.

2.4.6 Conclusion

Performance metrics used in the time series have different characteristics. Knowing these characteristics allows understanding how accurate forecasting performance is. When performance measures are compared, it is seen that MSE, RMSE and NRMSE penalize, while MAE and MAPE do not penalize extreme errors. None of the metrics provide information about the direction of the error and none make a positive or negative difference. Mean forecast error (MFE) and Signed mean squared error (SMSE) performance measures can be used when the sign and direction information is important.

3. DEEP LEARNING

3.1 Introduction

Deep Learning (DL) is a machine learning method using multi-layer deep ANN architectures. As specified in Section 2.3.3.2, ANN consists of three layers, the input layer, the hidden layer and the output layer. DL is the ANN model with multiple hidden layers. The first study involving the DL structure was carried out by Ivakhnenko and Lapa (1965). Even if the DL concept was based on the 1960s, the rise took place in recent years. This is because there is no processor power to train the deep architects, and there is no sufficient amount of data. Nowadays, with the increase of processor power and reaching enormous dimensions of data generated by digitization in many areas has provided the necessary infrastructure for the DL. These developments have enabled DL to become widely used in the field, such as computer vision, text processing, translation, time series prediction.

DL structures have been shaped according to different needs and different models have been put forward. In this section, commonly used DL models and application areas will be explained.

3.2 Convolutional Neural Networks

The basis of Convolutional Neural Networks (CNN) has been investigated on the functional architecture of cat's visual cortex (Hubel & Wiesel, 1962). This work has inspired an entire image to be divided into small pieces and to apply filters on these pieces. LeNet architecture was one of the first CNN architects and was proposed by Yann LeCun (1998). Until now, this architecture is being developed and new approaches are proposed. CNN are mainly used for object recognition and classification, but other areas such as voice recognition and natural language processing are also being studied.

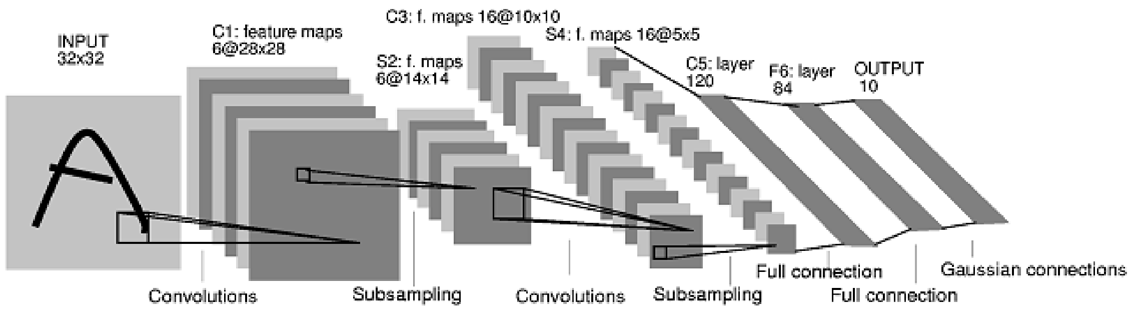

Figure 3.1: Convolutional Neural Network Architecture (LeCun, 1998)

CNN are an FNN architecture composed of many different layers. It will split the image into small pieces. CNN try to find patterns with different filters and distinguishes these patterns.. The LeNet architecture, which includes the layers that make up CNN, is shown in figure-x. The first layer of CNN is the Convolution Layer. In the Convolution layer, the image is scanned by separating the sub-regions of a certain size. Each sub-region is scanned by more than one filter. Weights that create the filters is used as shared. Each filter looks for a different pattern. At the end of each filter a feature map is formed. The resulting feature maps are passed through an activation function. The most frequently used activation function is Rectified Linear Unit (ReLU) function. The characteristic of the ReLU function is that it only passes positive values, shown in Figure 3.2. Other nonlinear activation functions such as sigmoid and tanh can also be used, but studies have shown that ReLU is better on the vanishing gradient problem (Mishkin et al., 2016).

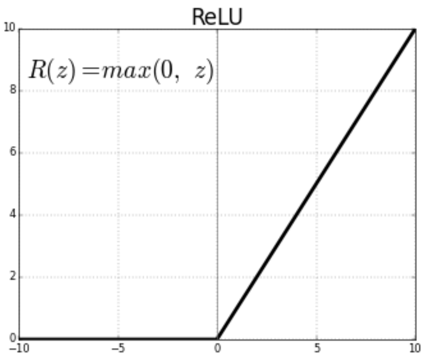

Figure 3.2: Rectified Linear Unit Function

The second layer of CNN is the Pooling Layer. In this layer, dimensional reduction is performed on feature maps passing through the activation function. After the convolution process, a large number of possible patterns arise. In the pooling layer, the most relevant ones of these patterns are selected and continued. The most commonly used pooling method is max-pooling, which is a non-linear down-sampling method. Max-pooling feature maps separate square windows of a certain size. For example, for a pooling of 2x2 window sizes, it takes the largest of the 4 values in the cells. This is done for windows in all feature maps. Convolution and pooling layers can be repeated many times.

The last layer is the fully connected layer. Fully connected layer consists of a multi-layer perceptron that uses the softmax activation function. Each element of the matrices that occur after the repeated operations in the first two layers is connected to this layer as an input. The classification operation takes place via this layer. CNN are trained with backpropagation algorithm. The most common problem in the training process is the vanishing gradient problem. As CNN are used in supervised learning, it needs a very large amount of labeled data to get effective results.

3.3 Restricted Boltzmann Machines

Recurrent Restricted Boltzmann Machines (RBM) are neural networks that represent probabilistic distributions. Initial work on RBMs with original name Harmonium was made in 1986 (Smolensky et al.). As shown in Fig. 3.3, RBMs are composed of two layers, visible layer and hidden layer. It is called Restricted because there is no connection between the neurons in the layers. RBMs are trained by trying to recreate the entered inputs so they can operate as unsupervised. In addition to being a successful feature extraction engine, it is also used in classification and regression processes.

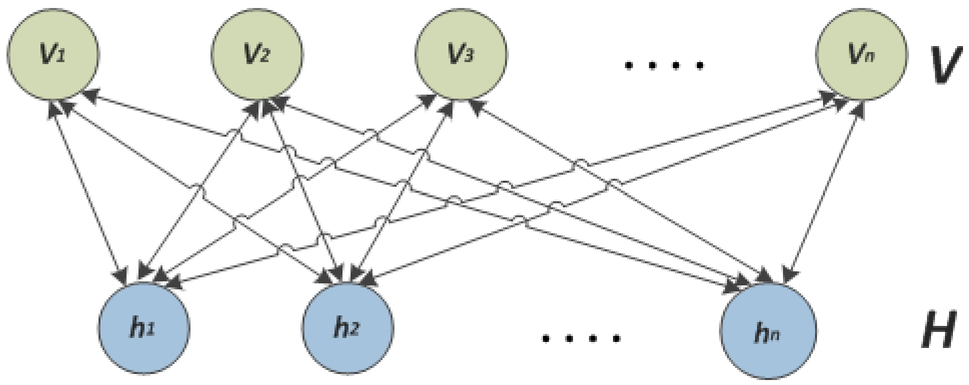

Figure 3.3: Restricted Boltzmann Machine

The RBM training process consists of 3 phases; forward pass, backward pass, and contrastive divergence. In the forward pass phase, the output from the visible layer is multiplied by the stochastic weights, and reaches the nodes in the hidden layer, where the outputs are generated by passing through the activation function. A logistic function is used as the activation function. In the backward pass, which is the second phase, the generated outputs are sent to the visible layer in the reverse direction. Although shared weights are used in the forward pass and backward pass processes, different bias values are used. In the last stage, the input values and the backward pass results are compared. To minimize the difference, weights are updated and this process is repeated many times. The Contrastive Divergence algorithm is used in the training process of RBMs (Hinton, 2002).

Consecutive stacking of multiple RBMs results in Deep Belief Networks (DBN). A sample DBN model is shown in Figure 3.4. In this network, the output of the previous RBM forms the input values of the next. Each RBM that forms the DBN is trained independently. There are studies in the literature that use RBM and DBN for time series forecasting. Qiu et al. (2014) proposed ensemble deep learning method consist of deep belief network and support vector regression for time series forecast on three different electricity load demand datasets. Kuremoto et al. (2014) used a 3-layer deep network of RBM to capture feature of input space of time series data. Terren-Serrano (2016) constructed various RBM structures to forecast global solar radiation.

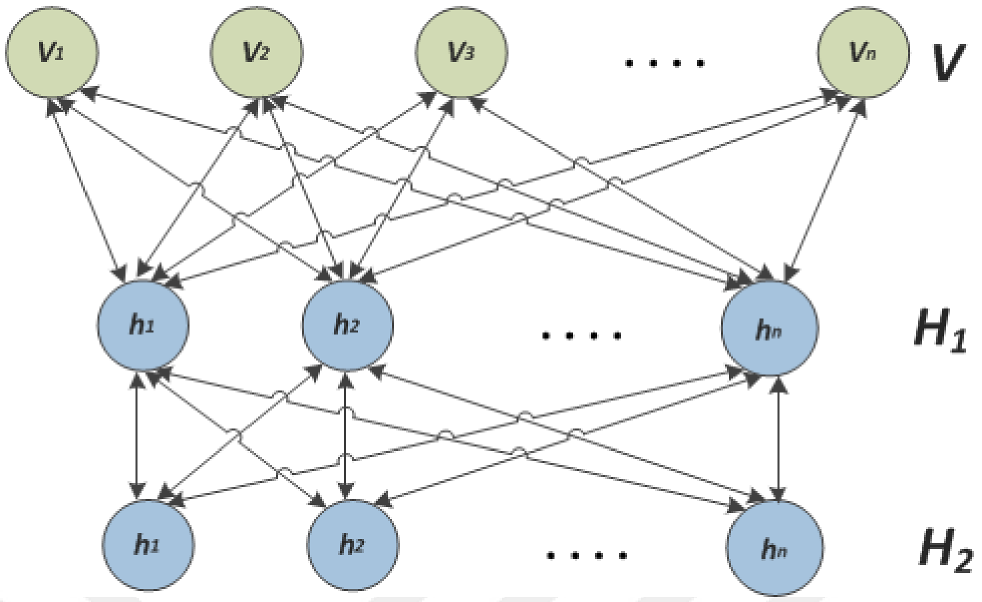

Figure 3.4: Deep Belief Network

3.4 Recurrent Neural Networks

Recurrent Neural Network (RNN) is a special kind of neural network developed to process sequential data. First studies were conducted in the 1980s (Hopfield, 1984; Rumelhart et al., 1986). In traditional structures, each sample is trained independently of each other, but this training is not a sufficient method for text, sound, image, and other data related to time. Independent training is not enough to preserve this knowledge because sequence information is also contained within sequence data. RNN offers this probing solution by taking inputs sequentially.

Unlike other FNNs, the RNN has feedback connections in the hidden layer units. With this feature, it can perform temporal processing and learn sequences. The hidden layer acts as a memory and can store sequential information. The RNN architecture can be transformed into an FNN structure by spreading over time. On this track, the RNN can be trained with a backpropagation version, a Backpropagation Through Time (BPTT) learning algorithm.

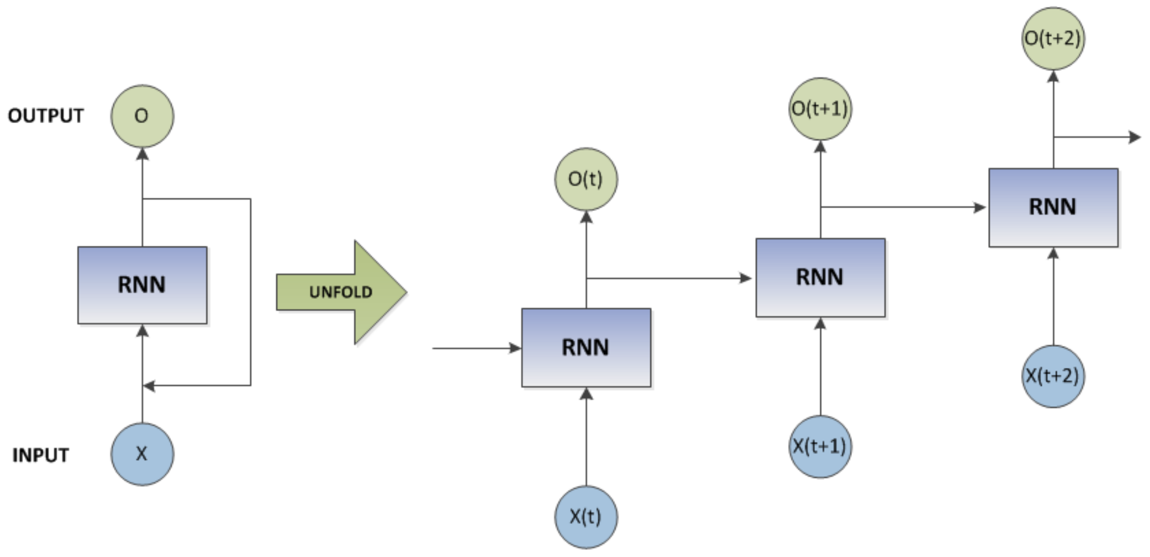

Figure 3.5: Unfolding RNN over Time

In addition to the inputs from the previous layer, each unit in the hidden layer receives as input the output value of the previous phase result. This feedback value is multiplied by its own weight, just like other inputs, and sent to the activation function. The function for the output calculation in the units in the hidden layer is as follows:

$$h(t) = f_H(\, W_{HH} * x(t) + W_{IH} * h(t-1)) \qquad (6)$$

where f_H is the activation function of the hidden layer, $x(t)$ is the input from the previous layer, W_{IH} is the weight of the links from previous layer, $h(t-1)$ is the feedback output calculated in the previous time step and W_{HH} is the weight of this feedback output.

Studies have shown that Simple RNN can store limited historical data (Bengio et al., 1994). Some data may be dependent on remote history data, which has led to a long-term dependency problem. To solve this problem, a customized RNN structure has been developed with a Long Short-Term Memory Unit (LSTM) and a Gated Recurrent Unit (GRU).

3.4.1 Long Short-Term Memory Unit

LSTM is a customized RNN model developed by Hochreiter and Schmidhuber (1997). LSTMs can keep track of long-term information through the gates they contain. In an LSTM unit there are basically three gates, which determine what information to store, which are input gate, forget gate and output gate. The input gate specifies which of the incoming input values will be stored in the next state. Forget gate determines which of the previous state information will no longer be stored. The output gate specifies which of the information in the new state will be sent as output. The gates that make up the LSTM unit are shown in Figure 3.6, where *i*, *f* and *o* represent input gate, forget gate and output gate, respectively, *C* represents the unit state, and $\hat{C}$ represents the next candidate state.

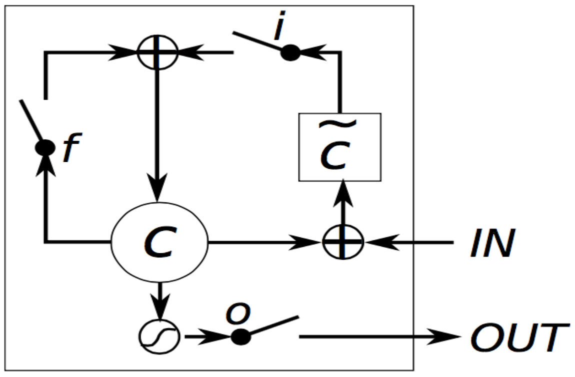

Figure 3.6: LSTM Unit (Chung, 2014)

The equations used to calculate the next output and state values in the LSTM unit are as follows:

$$f_t = \sigma(W_f * [x(t), C(t-1), h(t-1)] + b_f) \quad (7)$$

$$i_t = \sigma(W_i * [x(t), C(t-1), h(t-1)] + b_i) \quad (8)$$

$$o_t = \sigma(W_o * [x(t), C(t), h(t-1)] + b_o) \quad (9)$$

$$C(t) = C(t-1) * f_t + \hat{C} * i_t \quad (10)$$

where σ is the activation function, $x(t)$ is the input, $h(t-1)$ is the previous output, W_i, W_f, W_o and b_i, b_f, b_o are the weights and biases of input gate, forget gate and output gate, respectively.

3.4.2 Gated Recurrent Unit

GRU is another gated recurrent unit that can learn the long-term dependencies proposed by Bahdanau et al. (2014). Unlike the LSTM, as shown in Figure 3.7, there is no memory unit and there are 2 gates instead of 3 gates. Having a simpler architecture requires less computation and can be trained faster. Despite having a less complex structure, studies have shown that performance is comparable to LSTM (Chung et al., 2014).

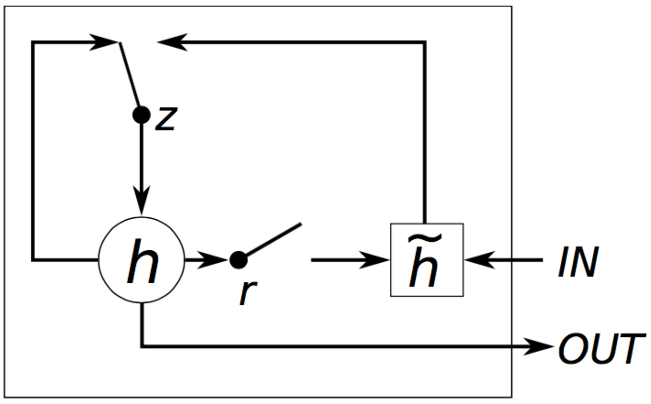

Figure 3.7: Gated Recurrent Unit (Chung, 2014)

GRU unit contains update z and reset r gates. The update gate specifies what information to keep in the next state. The reset gate specifies how information from the previous state and the new input are to be combined. The equations used to calculate the next output and state values in the GRU unit are as follows:

$$z_t = \sigma(W_z * [x(t), h(t-1)]) \tag{11}$$

$$r_t = \sigma(W_r * [x(t), h(t-1)]) \tag{12}$$

$$\hat{h}(t) = \sigma(W_h * [x(t), (r_t * h(t-1))]) \tag{13}$$

$$h(t) = (1 - z_t) * h(t-1) + z_t * \hat{h}(t) \tag{14}$$

where σ is the activation function, $x(t)$ is the input, $h(t-1)$ is the previous output, W_z, W_r and W_h are the weights of update gate, reset gate and candidate output, respectively.

3.4.3 Backpropagation Through Time

Traditional neural networks are trained via the backpropagation algorithm (Rumelhart et al.; 1985). The backpropagation algorithm aims to minimize the cost function by updating the weights between layers. The algorithm consists of two repetitive phases, propagation and weight update. In the first phase, input vector propagated forward through layers and output vector is obtained. In the second stage, the error value is calculated through the cost function by comparing the expected output values with the output value obtained. The error values are propagated backwards and the weights are updated to minimize the cost function. This operation is repeated until the error function reaches the desired small size.

BPTT is a customized backpropagation algorithm used to train RNNs. Since RNNs use sequenced data, it is necessary to unfold them over time as illustrated in Figure 3.5 to train them. After unfolding, each time step that occurs corresponds to a layer. This structure is then trained by backpropagation like traditional FNNs. As the number of time steps used in RNN increases, the training time also increase because the represented network is getting deeper.

4. EXPERIMENTS

4.1 Introduction

The aim of the experiments presented in this section is to investigate the performance of RNN structures introduced in Section 3.4 on forecasting 1 hour horizon for solar radiation. For this purpose, different RNN structures are configured to process both univariate and multivariate data. Results are compared with traditional RNN structure and naïve method.

4.2 Univariate Experiments

In this section, experiments are conducted using univariate data (historical solar radiation). First, effect of depth is investigated on LSTM and GRU models. For this purpose 4 different structures are proposed by optimizing the hyper-parameters. Second, effect of seasonality is investigated and different machine learning models are experimented on seasonal data. According to the results of first and second experiments the hybrid model is proposed.

4.2.1 The Data

In this study, hourly global horizontal solar radiation data is used. The data covers 87600 instances for the 10-year period from January 1998 to December 2007. Measurements of data provided by French meteorological organization (Météo-France) were carried out at the meteorological station of Ajaccio (Corsica Island, France, 41° 55'N, 8° 44'E, 4 m above mean sea level) as demonstrated in Figure 4.1. The station is located between the Mediterranean Sea and the mountains. Sensors used in the station can work in the range of 0 - 90000 J / m^2 and annual maintenance is done regularly.

Location has 'Mediterranean' climate, hot summers with abundant sunshine and mild, dry, clear winters.

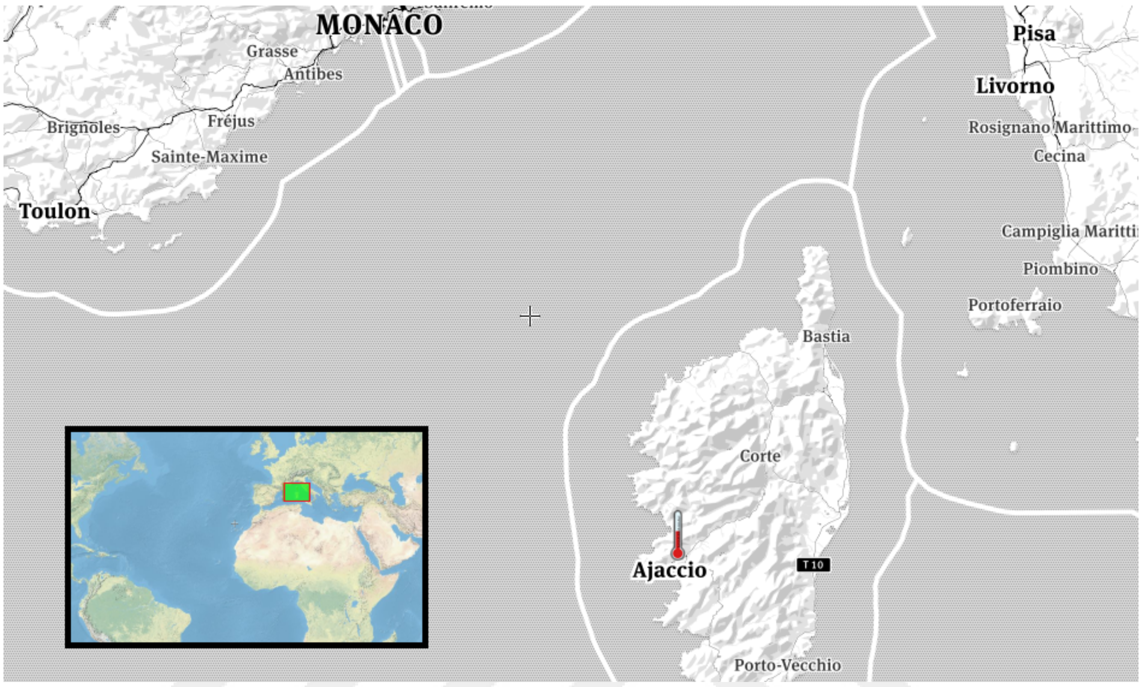

Figure 4.1: Meteorological Station of Ajaccio

The data has a yearly seasonal pattern as shown in Figure 4.2. The daily average horizontal solar radiation per hour is demonstrated in Figure 4.3.

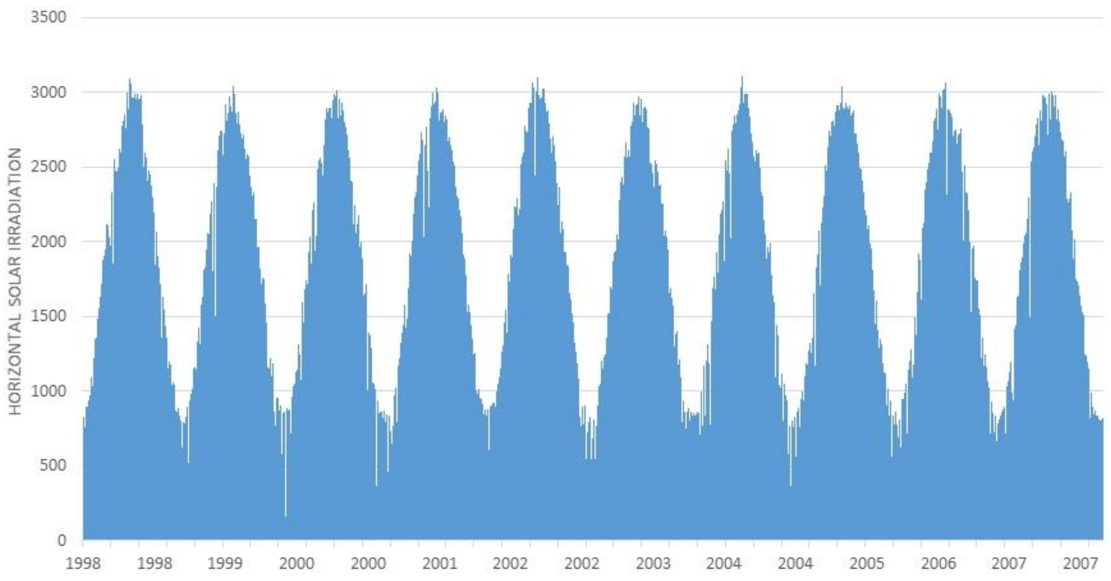

Figure 4.2: Daily Horizontal Solar Radiation

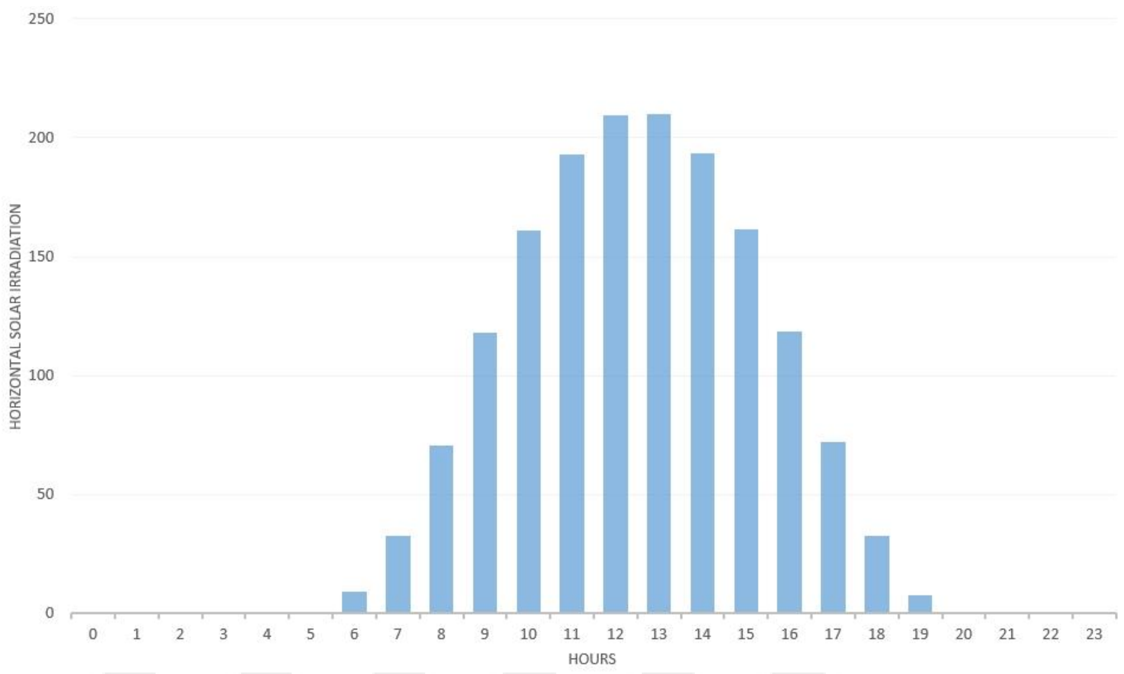

Figure 4.3: Average Horizontal Solar Radiation per Hour

4.2.2 RNN Models

In this section, LSTM and GRU models were used for 1 hour horizon time series forecasting on the data mentioned in Section 4.2. 1. Single layer and two layer structures were formed and the effect of the depth was investigated. Hyper-parameters are tuned to optimize the models. The obtained optimized single layer and two layer models were compared with naive method and simple RNN models.

There is no golden method known to optimize artificial neural networks. In order to optimize the models constructed in this study, the effect of the parameters through the experiments on the model performance was observed. Examined parameters are window size, number of neurons and number of epochs. Each parameter has been increased by doubling until there is no improvement in order to obtain best value that satisfies the most successful result. The experiments were run 6 times for the same configuration in order to avoid local minimums. Best and mean results are obtained. Data are normalized between 0 and 1, then all zero values are removed. RMSProp was used as an optimizer with 0.001 learning rate for all experiments (Hinton et al.; 2012).

4.2.2.1 Single Layer LSTM Structure

This structure consists of 1 hidden LSTM layer with 64 neurons. After the hyper-parameters are optimized, best model is obtained at 40 epochs with batch size of 32. Best window size (sequenced previous instances used as an input) is found as 64. Experimental details for hyper-parameter tuning are demonstrated in Appendix A. Figures that show training and testing losses are given in Appendix B. Figure 4.4 illustrates the single layer LSTM structure.

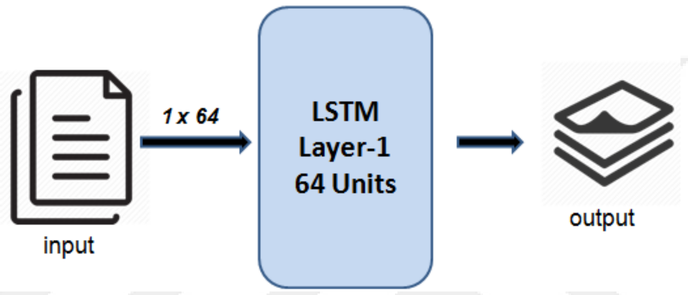

Figure 4.4: Single Layer LSTM Structure

4.2.2.2 Single Layer GRU Structure

This structure is consists of 1 hidden GRU layer with 64 neurons. After the hyper-parameters are optimized, best model obtained at 64 epochs with batch size of 32. Best window size is found as 64. Experimental details for hyper-parameter tuning are demonstrated in Appendix A. Figures that show training and testing losses are given in Appendix B. Figure 4.5 illustrates the single layer GRU structure.

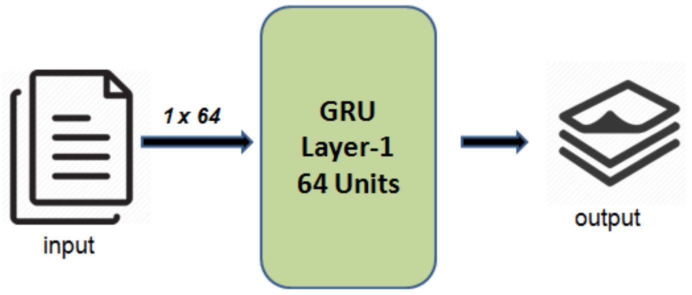

Figure 4.5: Single Layer GRU Structure

4.2.2.3 Two Layer LSTM Structure

This structure consists of 2 hidden LSTM layers with 128 and 4 neurons respectively. After the hyper-parameters are optimized, best model obtained at 32 epochs with batch size of 32. Best window size is found as 64. Experimental details for hyper-parameter tuning are demonstrated in Appendix A. Figures that show training and testing losses are given in Appendix B. Figure 4.6 illustrates the two layer LSTM structure.

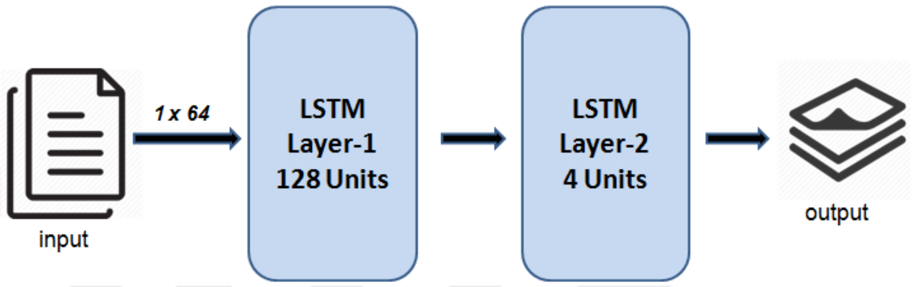

Figure 4.6: Two Layer LSTM Structure

4.2.2.4 Two Layer GRU Structure

This structure consists of 2 hidden LSTM layers with 128 and 8 neurons respectively. After the hyper-parameters are optimized, best model obtained at 32 epochs with batch size of 32. Best window size is found as 64. Experimental details for hyper-parameter tuning are demonstrated in Appendix A. Figures that show train and test losses are given in Appendix B. Figure 4.7 illustrates the two layer GRU structure.

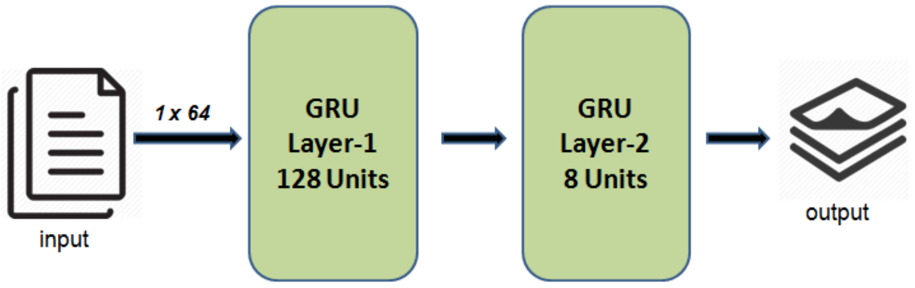

Figure 4.7: Two Layer GRU Structure

4.2.2.5 Results

Optimization results are compared with Naive method and simple RNN model in Table 4.1. According to the results, LSTM and GRU models are not superior to each other. We observed that LSTM and GRU models are more successful than conventional simple RNN architectures. Two layer architectures have shown slight success compared to single layer architectures.

Table 4.1: Comparison of RNN Models

Model	Number of Layers	Result (NRMSE)
Naïve Method	-	0.37
Simple RNN	1	0.222499
Simple RNN	2	0.219836
LSTM	1	0.213652
LSTM	2	0.211575
GRU	1	0.214065
GRU	2	0.211165

4.2.3 Hybrid Model

Solar radiation is strongly dependent on clouds in the sky. Cloud density and speed on the sky may change depending on the season. For example, at the summer season, sky is very clear and radiation shows linear pattern, however at the other seasons cloud density on the sky is higher and radiation shows nonlinear pattern.

In this section, seasonal behavior of the solar radiation data is investigated. For this purpose we divided the data into four seasons (Table 4.2). Hybrid model is constructed according to the result of experiments using various models on seasonal data.

Table 4.2: Seasonal Data

Season	Start Date	End Date	Number of Instances	Standard Deviation
Winter	December 1	February 28	9316	60.14
Spring	March 1	May 31	12750	104.49
Summer	June 1	August 31	14104	111.77
Fall	September 1	November 30	10626	83.13

4.2.3.1 Hybrid Structure

As the solar radiation data shows different patterns depending on the season, using hybrid models may increase the forecast accuracy. Various hybrid approaches and algorithms have been applied in the literature on solar radiation forecast. Benmouiza and Cheknane (2013) combined k-means and NAR neural network models to forecast hourly solar radiation in their study. Bhardwaj et al. (2013) proposed a hybrid technique by combining HMM and Fuzzy models. Voyant et al. (2012) present a hybrid model (ARMA and ANN) on their study. Ji and Chee (2011) are used ARMA and Time Delay Neural Network to build hybrid model. Chen et al. (2013) hybridized fuzzy and neural network methods on their study. Finally, Reikard (2009) compared time series forecast methods on solar radiation, and found that hybrid approach shows better results compared to the singular models especially on high resolution time series.

As shown in the Table 4.2 the data are divided into 4 seasons, then are reshaped using previous 10 hours as the input values. Training set consists of 80% of data and remaining is used as the test set. We applied different linear and non-linear models for each seasonal data to find suitable model that fits the seasonal pattern. In addition to the RNN models presented in Section 4.2.2, Random Forest, SVM, MLP, Decision Tree, Linear Regression methods are applied. Each experiment is conducted 6 times and best values are used on comparison. Errors are calculated as NRMSE. The results of the experiments for each season are demonstrated in Table 4.3.

Table 4.3: Seasonal Experiments

Model	Winter	Spring	Summer	Fall
RandomForest	**0.293523**	**0.213966**	**0.149608**	0.233139
SVM	0.363425	0.258853	0.185519	0.274522
MLP	0.323508	0.235358	0.157793	0.252952
Decision Tree	0.305215	0.225275	0.155196	0.245034
Linear Regression	0.349531	0.248440	0.178668	0.268115
LSTM (2-layer)	0.320697	0.220746	0.170390	0.217725
GRU (2-layer)	0.321263	0.224668	0.173381	**0.215438**

Hybrid model is constructed according to seasonal experimental results. Random Forest method is used for winter, spring and summer models and 2 layered GRU is used for fall model. Combined structure is illustrated in Figure 4.8.

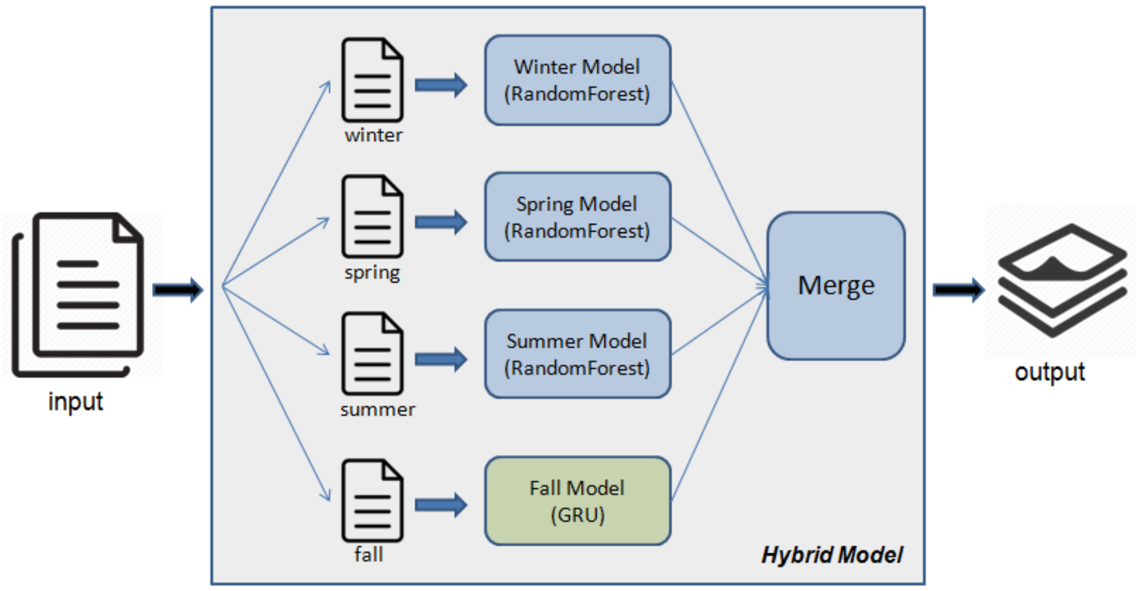
input
winter
spring
summer
fall
Winter Model (RandomForest)
Spring Model (RandomForest)
Summer Model (RandomForest)
Fall Model (GRU)
Merge
output
Hybrid Model

Figure 4.8: Hybrid Model

Error rate of hybrid model is calculated by combining the error of each seasonal model. Single models were calculated in NRMSE. To combine these errors, first each error rate is converted to MSE metric then total error value is calculated with multiplying by weights (sizes). The calculated values are summed and divided by the total size. The equations used to calculate the next output and state values in the LSTM unit are as follows:

$$MSE_s = \left(\frac{1}{n_s}\sum_{i=1}^{n_s} y_i * NRMSE_s\right)^2 \tag{15}$$

$$NRMSE_t = \frac{\sqrt{\frac{1}{n_t}\sum_{i=1}^{n} MSE_i * n_i}}{\frac{1}{n_t}\sum_{i=1}^{n_t} y_i} \tag{16}$$

where MSE_s and $NRMSE_s$ are seasonal errors, n_s is number of instance in the season, $NRMSE_t$ is total error and n_t is total number of instance.

4.2.3.2 Results

According the Seasonal experiments, Random Forest method outperform in all seasons except Fall. LSTM and GRU models have better results on Fall data. Hybrid structure is constructed using Random Forest and GRU models as shown in the Figure 4.8. Hybrid model shows significantly better performance compared to the experiments in Section 4.2.2. Table 4.4 shows the comparison of Hybrid model and RNN models.

Table 4.4: Comparison of Hybrid and RNN Models

Model	Result (NRMSE)
Naïve Method	0.37
Hybrid Model	0.199626
LSTM (1 Layer)	0.213652
LSTM (2 Layers)	0.211575
GRU (1 Layer)	0.214065
GRU (2 Layers)	0.211165

4.3 Multivariate Experiments

Magnitude of solar radiation can be affected from other meteorological variables. Many studies are conducted to reveal relation between solar radiation and other meteorological variables. Most of them found that there is a strong dependency. Sfetsos and Coonick (2000) compared univariate and multivariate approach on solar radiation forecasting on their study. They used temperature, pressure, wind speed, and wind direction as additional parameters in their models. They reported that extra parameters improve the performance of adaptive neuro-fuzzy inference and ANN models. Yadav et al. (2014) investigated the most relevant parameters for solar radiation prediction models. They identified temperature, altitude and sunshine hours as the effective parameters. Chen and Li (2014) used sunshine duration, temperature, humidity, and pressure as the attributes in their SVM model. They observed that sunshine duration and temperature significantly improves the model. Sun et al. (2015) examined the relationships between solar radiation and meteorological variables in their study and they found strong correlation between them. They also observed higher correlation between solar radiation and sunshine duration than temperature. Finally, Voyant et al. (2011) conducted multivariate experiments using the same dataset used in our study for forecasting daily solar radiation with an ANN model. They observed that using multivariable decrease the error rate between 0.5% and 1% (NRMSE).

In this section, additional meteorological variables (given in Table 4.5) are used in addition to solar radiation data to train models. Experiments are conducted using one layer LSTM structure. First, effect of each parameter is investigated. For this purpose 7 different structures (for each parameter) are proposed and their hyper-parameters are optimized. Second, most effective parameters are selected and used to train new model. Finally, all parameters are used to train the obtained optimized model from previous experiments. Results are compared with univariate and hybrid models.

4.3.1 The Data

In addition to the solar radiation (GLO) data given in section 4.2.1, 8 extra meteorological parameters collected from the same meteorological station at the same period are used. Table 4.5 shows the explanations, units, and abbreviations of the parameters.

Table 4.5: Explanation of Meteorological Parameters

Parameter	Explanation	Measurement Unit	Abbreviation
Temperature	Temperature under shelter	°C	TEMP
Pressure	Atmospheric pressure	Pa	PRES
Humidity	Relative humidity	%	HUM
Nebulosity	Total nebulosity	Octas	NEB
Wind Speed	Mean wind speed at 10 meters	m/s	WS
Wind Direction	Wind direction at 10 meters	360°	WD
Insolation	Sunshine duration (120 W.m^2)	min	INS
Rain	The height of rain	mm	RAIN

In the preprocessing step, missing data are filled with mean of previous and next value except pressure (between 1-30 instances). Pressure data are filled with global mean (750 instances). To understand the behavior of parameters and correlations between them data are analyzed. Table 4.6 and Table 4.7 show the analysis results.

Table 4.6: Analysis of Meteorological Parameters

Parameter	Minimum	Maximum	Mean	Standard Deviation
TEMP	-4.2	38.4	15.597	6.88
PRES	981.5	1039	1014.881	6.567
HUM	8	99	73.543	14.122
NEB	0	9	3.63	2.825
WS	0	21	3.426	1.766
WD	0	360	131.298	95.251
INS	0	60	19.125	26.084
RAIN	0	30	0.065	0.534
GLO	0	369	66.258	96.349

Table 4.7: Correlation Matrix of Meteorological Parameters

	TEMP	PRES	HUM	NEB	WS	WD	INS	RAIN	GLO
TEMP	1	-0.14	-0.45	-0.12	0.11	0.47	0.53	-0.04	0.58
PRES	-0.14	1	0.02	-0.3	-0.13	-0.15	0.09	-0.15	0.01
HUM	-0.45	0.02	1	0.13	-0.19	-0.47	-0.55	0.14	-0.52
NEB	-0.12	-0.3	0.13	1	-0.05	0.06	-0.27	0.17	-0.15
WS	0.11	-0.13	-0.19	-0.05	1	0.31	0.18	0.03	0.35
WD	0.47	-0.15	-0.47	0.06	0.31	1	0.58	0.01	0.64
INS	0.53	0.09	-0.55	-0.27	0.18	0.58	1	-0.09	0.85
RAIN	-0.04	-0.15	0.14	0.17	0.03	0.01	-0.09	1	-0.07
GLO	0.58	0.01	-0.52	-0.15	0.35	0.64	0.85	-0.07	1

According to correlation matrix results global solar radiation has; strong positive correlation with insolation, intermediate positive correlation with temperature, wind speed and wind direction, intermediate negative correlation with humidity and weak correlation with pressure, nebulosity pressure and rain. Figure 4.9, Figure 4.10, and Figure 4.11 illustrate pattern of each parameters for 10 days, 1 year, and 10 years, respectively.

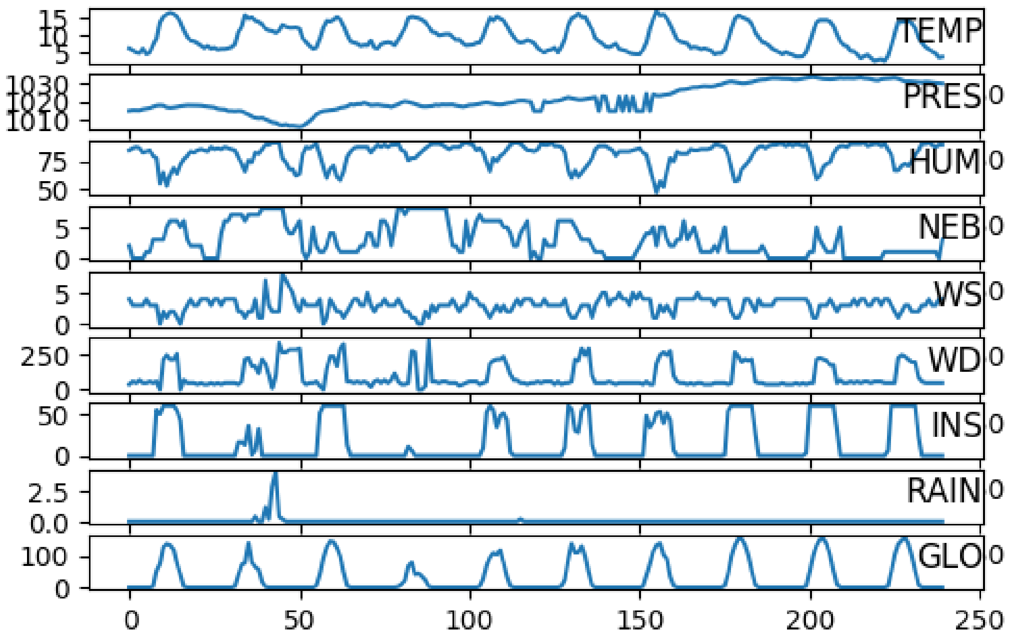

Figure 4.9: Patterns of Meteorological Parameters for 10 days

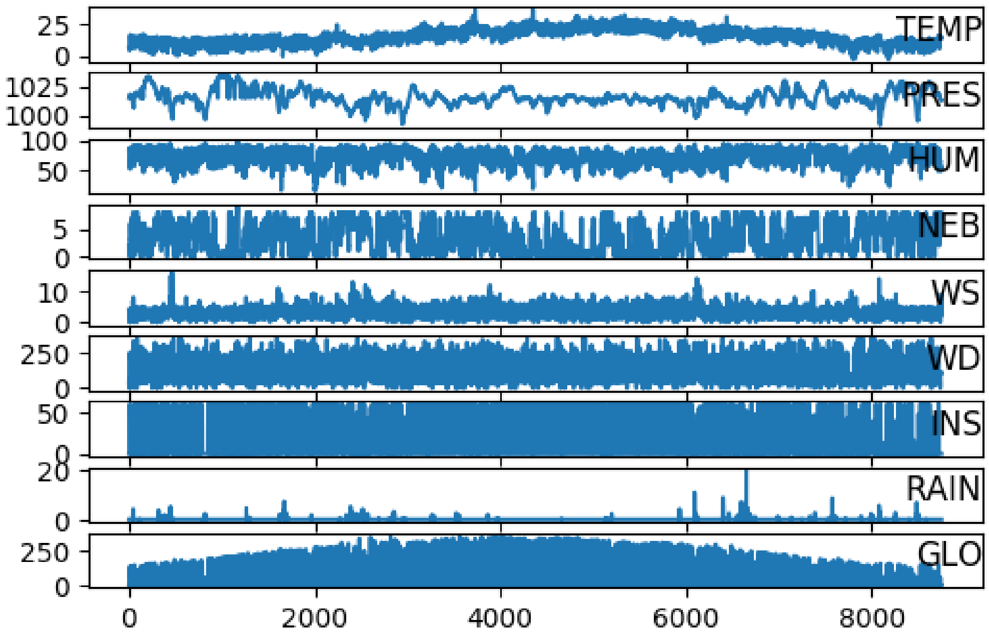

Figure 4.10: Patterns of Meteorological Parameters for 1 year

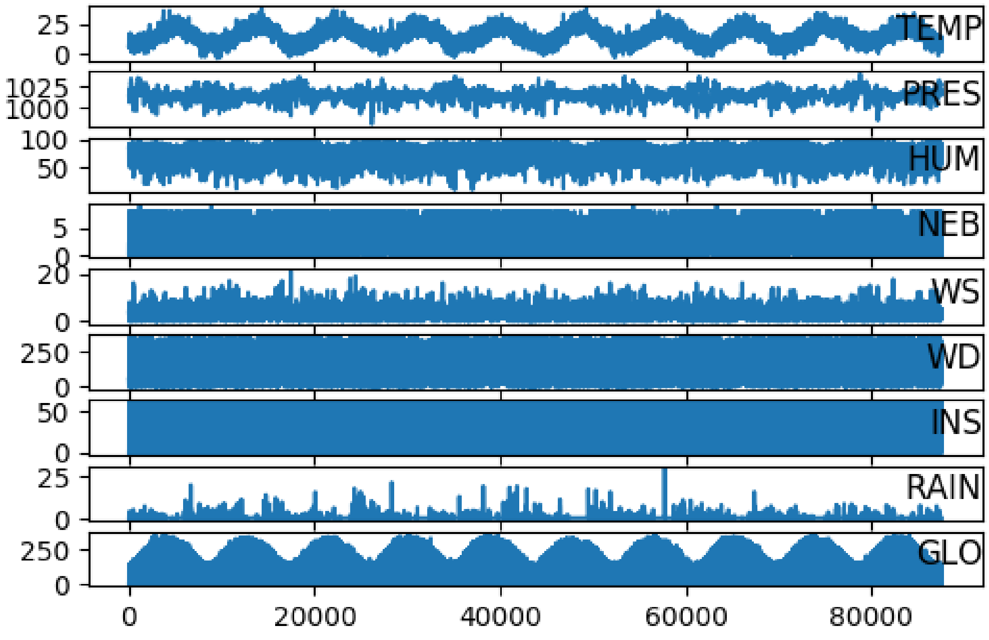

Figure 4.11: Patterns of Meteorological Parameters for 10 years

4.3.2 LSTM Structure

First set of experiments are conducted to investigate effect of each parameter. For this purpose 7 different models are trained using each couple, for example solar radiation and temperature. Only wind speed and wind direction data are used together. Each model is optimized through hyper-parameters tuning. Single layer LSTM is used for experiments. Best structure is obtained with 64 neurons and 64 window size at 32 epochs with batch size of 32 for all models. Results of experiments are shown in Table 4.8. Experimental details for hyper-parameter tuning are demonstrated in Appendix A. Figures that show training and testing losses are given in Appendix B.

Table 4.8: Effect of each Meteorological Parameter

Input Parameters	Best (NRMSE)	Mean (NRMSE)
GLO, TEMP	0.180151	0.182558
GLO, PRES	0.204260	0.206099
GLO, HUM	0.200067	0.201434
GLO, NEB	0.179581	0.179787
GLO, WS, WD	0.205194	0.206534
GLO, INS	0.201077	0.201685
GLO, RAIN	0.202923	0.203737

After effect of each parameter is found, new set of experiments are conducted using most effective parameters. As shown in Table 4.8 TEMP and NEB are the most effective parameters and HUM and INS are the secondary effective parameters. First, effect of TEMP and NEB parameters together is experimented. Second, secondary effective parameters are added individually to the most effective parameters. Finally, all parameters are used to train the model. Results of the experiments are shown in Table 4.9. The structure used in the experiments consists of 1 hidden layer with 128 GRU neurons. Models are trained with 64 epochs at 64 as window size and 32 as batch size. Figure 4.12 illustrates the best configured single layer LSTM structure on multivariate experiments. Figures that show training and testing losses are given in Appendix B.

Table 4.9: Effect of Multivariate Inputs

Input Parameters	Best (NRMSE)	Mean (NRMSE)
GLO, TEMP, NEB	0.160376	0.162538
GLO, TEMP, NEB, HUM	0.159261	0.161425
GLO, TEMP, NEB, INS	0.162142	0.163072
GLO, TEMP, NEB, HUM, INS	0.160638	0.163214
ALL	0.160397	0.160789

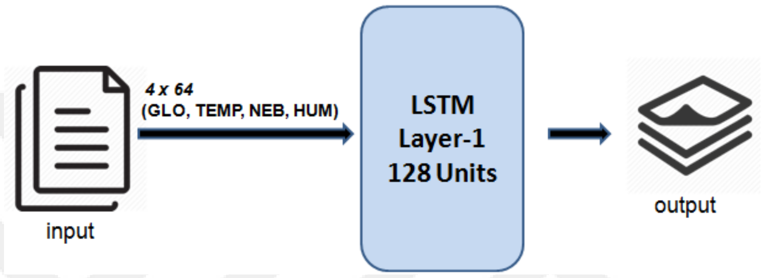

Figure 4.12: Multivariate Single Layer LSTM Structure

4.3.3 Results

According the results of multivariate experiments, TEMP and NEB parameters significantly improve the performance of the model. Other parameters slightly contribute to the performance. We obtain the best performance with using GLO, TEMP, NEB, and HUM parameters. Table 4.10 shows the comparison of multivariate model with the previous experiments.

Table 4.10: Comparison of Multivariate Model and Previous Models

Model	Result (NRMSE)
Naïve Method	0.37
Hybrid Model	0.199626
Single Layer LSTM (univariate)	0.213652
Single Layer LSTM (multivariate)	0.159261

5. CONCLUSION AND PERSPECTIVES

In this study, we applied RNN models for time series forecasting on solar radiation data for 1 hour horizon. Experiments are conducted for univariate and multivariate models.

In univariate experiments, LSTM and GRU models are constructed and the effect of the depth is investigated and hyper-parameters are tuned. According to the results, LSTM and GRU models are not superior to one another. We observed that LSTM and GRU models are more successful than conventional simple RNN architectures. Two layer architectures have shown slightly better performance compared to single layer architectures. After the best structure is found, the seasonal performance of model is investigated and compared with other machine learning methods. According to seasonal experimental results, a new hybrid model is proposed using a two layer GRU model and random forest model. This hybrid model showed better performance compared to the LSTM and GRU models.

In multivariate experiments, the effect of additional meteorological parameters is investigated. According the results of multivariate experiments, Temperature and Nebulosity parameters significantly improved the performance of the model. Other parameters contribute very slightly. We obtain the best performance using Global Solar Radiation, Temperature, Nebulosity, and Humidity parameters.

Since the training of the multivariate models takes a very long time, the experiments were carried out only for the LSTM model. As a result of this limitation the effect of additional parameters on the GRU model and two layer architectures have not been explored Also the seasonal performances of multivariate models have not been studied. Because of the longevity of the experiments, in hyper-parameter tuning, the number of neurons, the number of epochs and the window size are optimized, the optimization of other parameters such as; learning rate, batch size, and activation function are considered as future work.

The results show that the LSTM and GRU models can be suitable and competitive for 1 hour horizon time series forecasting on the solar radiation data. Seasonal experiments showed that hybrid approaches improve the performance. Multivariate experiments proved that solar radiation is strongly dependent on other meteorological parameters.

This study can be extended with possible future work:

- Construction of deep architectures using multivariate data;
- Construction of hybrid architectures using multivariate data;
- Implementation of Hidden Markov Models for comparison;
- Investigation to the effect of different activation functions and optimizers;
- Optimization of parameters (learning rate and batch size);
- Testing proposed model with the data from different locations.

REFERENCES

Adhikari, R. and Agrawal, R. K. (2013). An introductory study on time series modeling and forecasting. Comp. Res. Repository. 1302.6613: 1–67.

Allende, H., Moraga, C., Salas R. (2002). Artificial neural networks in time series forecasting: a comparative analysis. *Kybernetika* 38(6): 685–707.

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

Balkin, S. D. and Ord, J. K. (2000). Automatic neural network modeling for univariate time series. *International Journal of Forecasting* 16: 509–515

Bao, W., Yue, J., Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS one*, 12(7): e0180944.lh

Bengio, Y., Simard, P., Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2): 157-166.

Benmouiza, K. and Cheknane, A. (2013). Forecasting hourly global solar radiation using hybrid k-means and nonlinear autoregressive neural network models. *Energy Conversion and Management*, 75: 561-569.

Bhardwaj, S., Sharma, V., Srivastava, S., Sastry, O. S., Bandyopadhyay, B., Chandel, S. S., Gupta, J. R. P. (2013). Estimation of solar radiation using a combination of Hidden Markov Model and generalized Fuzzy model. *Solar Energy*, 93: 43-54.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1): 5-32.

Chen, J. L., and Li, G. S. (2014). Evaluation of support vector machine for estimation of solar radiation from measured meteorological variables. *Theoretical and applied climatology*, 115(3-4): 627-638.

Chen, S. X., Gooi, H. B., Wang, M. Q. (2013). Solar radiation forecast based on fuzzy logic and neural networks. *Renewable energy*, 60: 195-201.

Chung, J., Gulcehre, C., Cho, K., Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555.*

Conejo, A.J., Palazas, M.A., Espimola, R., Molina, A.B. (2002). Day-ahead electricity price forecasting using the wavelet transform and ARIMA models. *IEEE transactıons on power systems*, 20(2): 1035–1042.

Dudek, G. (2015). Short-term load forecasting using random forests. *Advances in Intelligent Systems and Computing,*323: 821-828.

Fischer, T. and Krauß, C. (2017). Deep learning with long short-term memory networks for financial market predictions, *FAU Discussion Papers in Economics*.

Hinton, G. E. (2002). Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*. 14 (8): 1771–1800.

Hinton, G., Srivastava N., SwerskyK. (2012) RMSProp: Divide the gradient by a running average of its recent magnitude. *Neural networks for machine learning, Coursera lecture 6e*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8): 1735-1780.

Hopfield, J. J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the national academy of sciences*, 81(10): 3088-3092.

Hsu, D. (2017). Time Series Forecasting Based on Augmented Long Short-Term Memory, *arXiv preprint arXiv: 1707.00666, 2017*.

Huang, J., Korolkiewicz, M., Agrawal, M., Boland, J. (2013). Forecasting solar radiation on an hourly time scale using a Coupled AutoRegressive and Dynamical System (CARDS) model, *Solar Energy*, 87: 136-149.

Hubel, D. H. and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1): 106-154.

Ivakhnenko, A. G. and Lapa, V. G. (1965). Cybernetic Predicting Devices. *CCM Information Corporation*.

Jeffrey, P. (2015). Fundamental concepts of time-series econometrics [online]. URL:http://www.reed.edu/economics/parker/312/tschapters/S13_Ch_1.pdf [accessed Jan 9, 2017].

Ji, W. and Chee, K. C. (2011). Prediction of hourly solar radiation using a novel hybrid model of ARMA and TDNN. *Solar Energy*, 85(5): 808-817.

Juban, J., Fugon, L., Kariniotakis, G. (2007). Probabilistic short-term wind power forecasting based on kernel density estimators. *European Wind Energy Conference and Exhibition*.

Kuremoto, T., Kimura, S., Kobayashi, K., Obayashi, M. (2014). Time series forecasting using a deep belief network with restricted Boltzmann machines. *Neurocomputing*, 137: 47-56.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278-2324.

Lv, Y., Duan, Y., Kang, W., Li, Z., Wang, F. Y. (2015). Traffic flow prediction with big data: a deep learning approach, *IEEE Transactions on Intelligent Transportation Systems*, 16(2): 865-873.

Merkel, G., Povinelli, R. J., Brown, R. H. (2017). Deep Neural Network Regression for Short-Term Load Forecasting of Natural Gas, *In 37th Annual International Symposium on Forecasting*.

Mihalakakou, G., Santamouris, M., Asimakopoulos, D. N. (2000). The total solar radiation time series simulation in Athens, using neural networks, *Theoretical and Applied Climatology*, 66(3): 185-197.

Mishkin, D., Sergievskiy, N., Matas, J. (2016). Systematic evaluation of CNN advances on the ImageNet. *arXiv:1606.02228*.

More, A. and Deo, M.C. (2003). Forecasting wind with neural networks. *Marine Structures* 16(1): 35–49.

Paoli, C., Voyant, C., Muselli, M., Nivet, M. L. (2010). Forecasting of preprocessed daily solar radiation time series using neural networks, *Solar Energy*, 84(12): 2146-2160.

Qiu, X., Zhang, L., Ren, Y., Suganthan, P.N. (2014). Ensemble deep learning for regression and time series forecasting. *IEEE Symposium Series on Computational Intelligence*.

Reikard, G. (2009). Predicting solar radiation at high resolutions: A comparison of time series forecasts, *Solar Energy*, 83(3): 342-349.

Rumelhart, D. E., Hinton, G. E., Williams, R. J. (1985). *Learning internal representations by error propagation*. No. ICS-8506. California Univ San Diego La Jolla Inst for Cognitive Science

Rumelhart, D. E., Hinton, G. E., Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088): 533-538.

Sfetsos, A. and Coonick, A. H. (2000). Univariate and multivariate forecasting of hourly solar radiation with artificial intelligence techniques. *Solar Energy*, 68(2): 169-178.

Smolensky, P (1986). Chapter 6: Information Processing in Dynamical Systems: Foundations of Harmony Theory. In *Rumelhart, David E.; McLelland, James L. Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*. MIT Press. pp. 194–281.

Stergiou, K.I. and Chirtou, E.D. (1996). Modelling and forecasting annual fisheries catches: comparison of regression, univariate and multivariate time series methods. Vol. 50 of *Fisheries Research*, pp. 1035–1042.

Sun, H., Zhao, N., Zeng, X., Yan, D. (2015). Study of solar radiation prediction and modeling of relationships between solar radiation and meteorological variables. *Energy Conversion and Management*, 105: 880-890.

Terren-Serrano, G. (2016). *Machine Learning Approach to Forecast Global Solar Radiation Time Series*, Master's thesis, University of Zaragoza

Vengertsev, D. (2014). *Deep learning architecture for univariate time series forecasting*. Technical report, Stanford University.

Voyant, C., Muselli, M., Paoli, C., Nivet, M. L. (2011). Optimization of an artificial neural network dedicated to the multivariate forecasting of daily global radiation. *Energy*, 36(1): 348-359.

Voyant, C., Muselli, M., Paoli, C., Nivet, M. L. (2012). Numerical weather prediction (NWP) and hybrid ARMA/ANN model to predict global radiation. *Energy*, 39(1): 341-355.

Wu, J.L. and Chang, P.C. (2012). A trend-based segmentation method and the support vector regression for financial time series forecasting. *Mathematical Problems in Engineering*.

Yadav, A. K., Malik, H., Chandel, S. S. (2014). Selection of most relevant input parameters using WEKA for artificial neural network based solar radiation prediction models. *Renewable and Sustainable Energy Reviews*, 31: 509-519.

Yang, D., Jirutitijaroen, P., Walsh, W. M. (2012). Hourly solar irradiance time series forecasting using cloud cover index, *Solar Energy*, 86(12): 3531-3543.

Zhang G.P. (2001). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing* Vol. 50 pp. 159–175.

Zhang, G. P. and Qi, M. (2005). Neural network forecasting for seasonal and trend time series. *European journal of operational research*, 160(2): 501-514.

APPENDICES

Appendix A.

The optimization experiments for the single-layer LSTM model are shown in Table A.1, Table A.2, and Table A.3, respectively.

Table A.1: Single-layer LSTM window size tuning

Number of Neurons	Number of Epochs	Window Size	Best (NRMSE)	Mean (NRMSE)
30	40	1	0.363870	0.364279
30	40	2	0.276411	0.277801
30	40	4	0.249072	0.247460
30	40	8	0.236184	0.242810
30	40	16	0.224898	0.228282
30	40	32	0.216498	0.218615
30	40	64	**0.214726**	0.218280
30	40	128	0.217557	0.219357

Table A.2: Single-layer LSTM number of neurons tuning

Number of Neurons	Number of Epochs	Window Size	Best (NRMSE)	Mean (NRMSE)
1	40	64	0.236154	0.241564
2	40	64	0.226201	0.227503
4	40	64	0.220388	0.222954
8	40	64	0.217480	0.220124
16	40	64	0.214506	0.217394
40	40	64	0.214726	0.218280
64	40	64	**0.213652**	0.219843
128	40	64	0.222160	0.248149

Table A.3: Single-layer LSTM number of epochs tuning

Number of Neurons	Number of Epochs	Window Size	Best (NRMSE)	Mean (NRMSE)
64	8	64	0.220850	0.229908
64	16	64	0.217642	0.223031
64	32	64	0.217799	0.225571
64	40	64	**0.213652**	0.219843
64	64	64	0.214807	0.216060

The optimization experiments for the single-layer LSTM model are shown in Table A.4, Table A.5, and Table A.6, respectively.

Table A.4: Single-layer GRU window size tuning

Number of Neurons	Number of Epochs	Window Size	Best (NRMSE)	Mean (NRMSE)
30	40	1	0.363895	0.364815
30	40	2	0.275833	0.278855
30	40	4	0.245283	0.246137
30	40	8	0.233821	0.236621
30	40	16	0.225712	0.227785
30	40	32	0.218613	0.219963
30	40	64	**0.217462**	0.221839
30	40	128	0.238937	0.242157

Table A.5: Single-layer GRU number of neurons tuning

Number of Neurons	Number of Epochs	Window Size	Best (NRMSE)	Mean (NRMSE)
1	40	64	0.238937	0.242157
2	40	64	0.233123	0.236277
4	40	64	0.224325	0.229378
8	40	64	0.219325	0.222661
16	40	64	0.217503	0.225403
30	40	64	0.217462	0.221839
64	40	64	**0.214736**	0.218363
128	40	64	0.216583	0.226974

Table A.6: Single-layer GRU number of epochs tuning

Number of Neurons	Number of Epochs	Window Size	Best (NRMSE)	Mean (NRMSE)
64	8	64	0.221328	0.230129
64	16	64	0.217834	0.219597
64	32	64	0.214682	0.218709
64	64	64	**0.214065**	0.218143
64	128	64	0.216983	0.220124

The optimization experiments for the two-layer LSTM model are shown in Table A.7, Table A.8, and Table A.9, respectively.

Table A.7: Two-layer LSTM window size tuning

Num. of Neurons (L1)	Num. of Neurons (L2)	Number of Epochs	Window Size	Best (NRMSE)	Mean (NRMSE)
32	8	32	1	0.363869	0.365237
32	8	32	2	0.257817	0.260344
32	8	32	4	0.243725	0.245013
32	8	32	8	0.233748	0.237458
32	8	32	16	0.224112	0.225946
32	8	32	32	0.217160	0.219650
32	8	32	64	**0.214083**	0.217465
32	8	32	128	0.215939	0.217726

Table A.8: Two-layer LSTM number of neurons tuning

Num. of Neurons (L1)	Num. of Neurons (L2)	Number of Epochs	Window Size	Best (NRMSE)	Mean (NRMSE)
16	8	32	64	0.214445	0.219393
32	8	32	64	0.214083	0.217465
64	8	32	64	0.213144	0.216074
128	8	32	64	0.212346	0.216435
256	8	32	64	0.211936	0.214954
128	4	32	64	**0.211575**	0.214416
128	16	32	64	0.214556	0.217934
256	4	32	64	0.212674	0.218435

Table A.9: Two-layer LSTM number of epochs tuning

Num. of Neurons (L1)	Num. of Neurons (L2)	Number of Epochs	Window Size	Best (NRMSE)	Mean (NRMSE)
128	4	8	64	0.218595	0.221475
128	4	16	64	0.213716	0.216859
128	4	32	64	**0.211575**	0.214416
128	4	64	64	0.214511	0.216623

The optimization experiments for the two-layer GRU model are shown in Table A.10, Table A.11, and Table A.12, respectively.

Table A.10: Two-layer GRU window size tuning

Num. of Neurons (L1)	Num. of Neurons (L2)	Number of Epochs	Window Size	Best (NRMSE)	Mean (NRMSE)
32	8	32	1	0.363925	0.364898
32	8	32	2	0.257033	0.258891
32	8	32	4	0.239896	0.246458
32	8	32	8	0.231609	0.234364
32	8	32	16	0.224348	0.229706
32	8	32	32	0.215713	0.219995
32	8	32	64	**0.213180**	0.217890
32	8	32	128	0.217072	0.223187

Table A.11: Two-layer GRU number of neurons tuning

Num. of Neurons (L1)	Num. of Neurons (L2)	Number of Epochs	Window Size	Best (NRMSE)	Mean (NRMSE)
16	8	32	64	0.215808	0.217407
32	8	32	64	0.213180	0.217890
64	8	32	64	0.214060	0.220389
128	8	32	64	**0.211165**	0.216483
256	8	32	64	0.213687	0.216021
128	4	32	64	0.213095	0.219377
128	16	32	64	0.213457	0.228449
128	32	32	64	0.214144	0.218405
256	16	32	64	0.214766	0.216510

Table A.12: Two-layer GRU number of epochs tuning

Num. of Neurons (L1)	Num. of Neurons (L2)	Number of Epochs	Window Size	Best (NRMSE)	Mean (NRMSE)
128	8	8	64	0.216992	0.218789
128	8	16	64	0.213552	0.217332
128	8	32	64	**0.211165**	0.216483
128	8	64	64	0.216251	0.221794

The optimization experiments for each additional parameter on the single-layer LSTM model are shown in Table A.13, Table A.14, Table A.15, Table A.16, Table A.17, Table A.18, and Table A.19, respectively.

Table A.13: Effect of temperature - Single-layer LSTM hyper-parameters tuning

Number of Neurons	Number of Epochs	Window Size	Best (NRMSE)	Mean (NRMSE)
16	16	1	0.363311	0.364340
16	16	2	0.251451	0.253735
16	16	4	0.225124	0.227863
16	16	8	0.211028	0.214515
16	16	16	0.205737	0.209124
16	16	32	0.203789	0.209114
4	16	32	0.223622	0.224808
8	16	32	0.212922	0.215250
32	16	32	0.192241	0.195718
64	16	32	0.190752	0.191547
64	4	32	0.209618	0.213008
64	8	32	0.201242	0.201866
64	32	32	0.186055	0.188991
64	64	32	0.184857	0.185768
64	128	32	0.188852	0.192043
64	32	64	0.180151	0.182558

Table A.14: Effect of pressure - Single-layer LSTM hyper-parameters tuning

Number of Neurons	Number of Epochs	Window Size	Best (NRMSE)	Mean (NRMSE)
16	16	1	0.363669	0.363968
16	16	2	0.275123	0.276769
16	16	4	0.244368	0.246037
16	16	8	0.232290	0.235710
16	16	16	0.225132	0.225886
16	16	32	0.213965	0.216785
4	16	32	0.231727	0.233618
8	16	32	0.222771	0.225194
32	16	32	0.212845	0.213338
64	16	32	0.209519	0.211350
64	4	32	0.217831	0.219996
64	8	32	0.211022	0.215780
64	32	32	0.206514	0.208251
64	64	32	0.209160	0.210872
64	32	64	0.204260	0.206099

Table A.15: Effect of humidity - Single-layer LSTM hyper-parameters tuning

Number of Neurons	Number of Epochs	Window Size	Best (NRMSE)	Mean (NRMSE)
16	16	1	0.363469	0.364831
16	16	2	0.270261	0.271118
16	16	4	0.240462	0.244287
16	16	8	0.231596	0.231979
16	16	16	0.221716	0.235797
16	16	32	0.211066	0.215217
4	16	32	0.229444	0.234037
8	16	32	0.224230	0.225946
32	16	32	0.207207	0.208249
64	16	32	0.205082	0.207706
64	4	32	0.216737	0.221553
64	8	32	0.208018	0.209430
64	32	32	0.201093	0.201513
64	64	32	0.204859	0.207934
64	32	64	0.200067	0.201434

Table A.16: Effect of nebulosity - Single-layer LSTM hyper-parameters tuning

Number of Neurons	Number of Epochs	Window Size	Best (NRMSE)	Mean (NRMSE)
16	16	1	0.360358	0.360450
16	16	2	0.246786	0.247881
16	16	4	0.225443	0.227574
16	16	8	0.212332	0.213902
16	16	16	0.203267	0.206268
16	16	32	0.193451	0.195393
4	16	32	0.216532	0.218519
8	16	32	0.201421	0.205542
32	16	32	0.188164	0.189882
64	16	32	0.184978	0.187300
64	4	32	0.199013	0.201283
64	8	32	0.189622	0.192185
64	32	32	0.182640	0.187287
64	64	32	0.182544	0.183532
64	32	64	0.179581	0.179787

Table A.17: Effect of insolation - Single-layer LSTM hyper-parameters tuning

Number of Neurons	Number of Epochs	Window Size	Best (NRMSE)	Mean (NRMSE)
16	16	1	0.363367	0.363685
16	16	2	0.232657	0.233004
16	16	4	0.221680	0.222228
16	16	8	0.215752	0.220351
16	16	16	0.210573	0.214115
16	16	32	0.207011	0.209415
4	16	32	0.212666	0.217349
8	16	32	0.211012	0.211930
32	16	32	0.207440	0.209335
16	4	32	0.218466	0.222132
16	8	32	0.212833	0.214291
16	32	32	0.204696	0.206536
16	64	32	0.202771	0.203186
16	64	64	0.201077	0.201685

Table A.18: Effect of wind - Single-layer LSTM hyper-parameters tuning

Number of Neurons	Number of Epochs	Window Size	Best (NRMSE)	Mean (NRMSE)
16	16	1	0.353687	0.354155
16	16	2	0.269787	0.270315
16	16	4	0.244531	0.249973
16	16	8	0.234594	0.237008
16	16	16	0.225131	0.227632
16	16	32	0.215414	0.221502
4	16	32	0.228492	0.232483
8	16	32	0.227567	0.227824
32	16	32	0.212512	0.216422
64	16	32	0.215104	0.209526
32	4	32	0.225321	0.227798
32	8	32	0.219678	0.225127
32	32	32	0.207562	0.211713
32	64	32	0.207213	0.208513
32	32	64	0.205194	0.206534

Table A.19: Effect of rain - Single-layer LSTM hyper-parameters tuning

Number of Neurons	Number of Epochs	Window Size	Best (NRMSE)	Mean (NRMSE)
16	16	1	0.363437	0.363592
16	16	2	0.273091	0.273780
16	16	4	0.243712	0.246456
16	16	8	0.234794	0.235523
16	16	16	0.224182	0.227605
16	16	32	0.214572	0.220914
4	16	32	0.229604	0.233192
8	16	32	0.224452	0.225170
32	16	32	0.210155	0.213180
64	16	32	0.206708	0.209617
64	4	32	0.219430	0.222762
64	8	32	0.212522	0.214641
64	32	32	0.204403	0.206282
64	64	32	0.208317	0.209562
64	32	64	0.202923	0.203737

Appendix B

In this section representation of test and train performances of models are given In figures, blue lines represent the training losses and yellow lines represent the test losses.

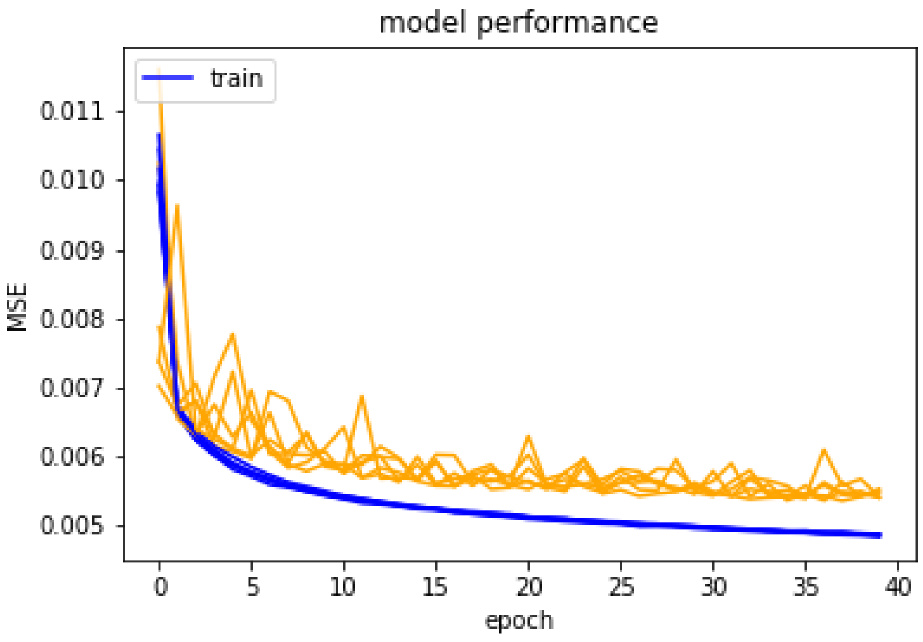

Figure B.1: Train and Test Losses of Single Layer LSTM

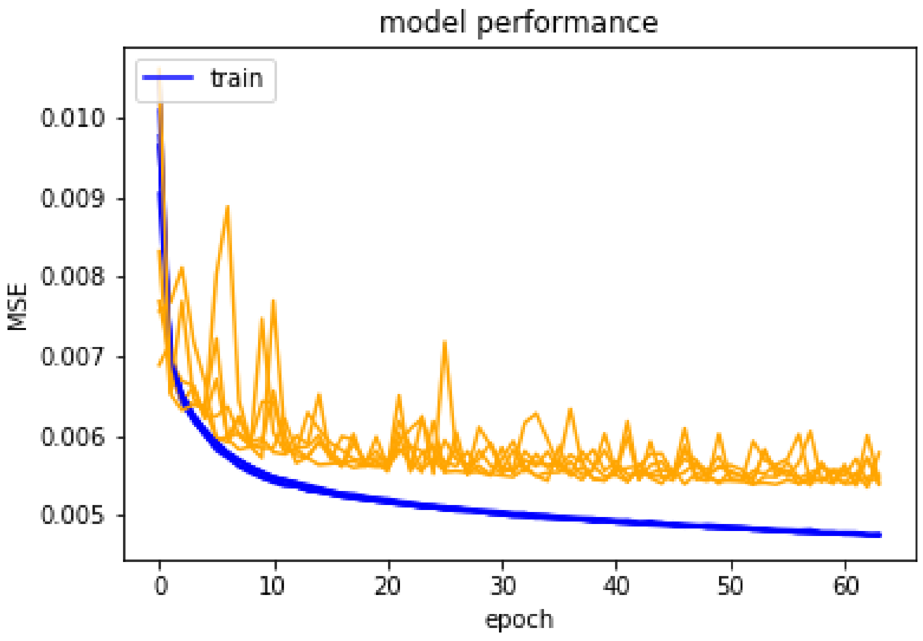

Figure B.2: Train and Test Losses of Single-layer GRU

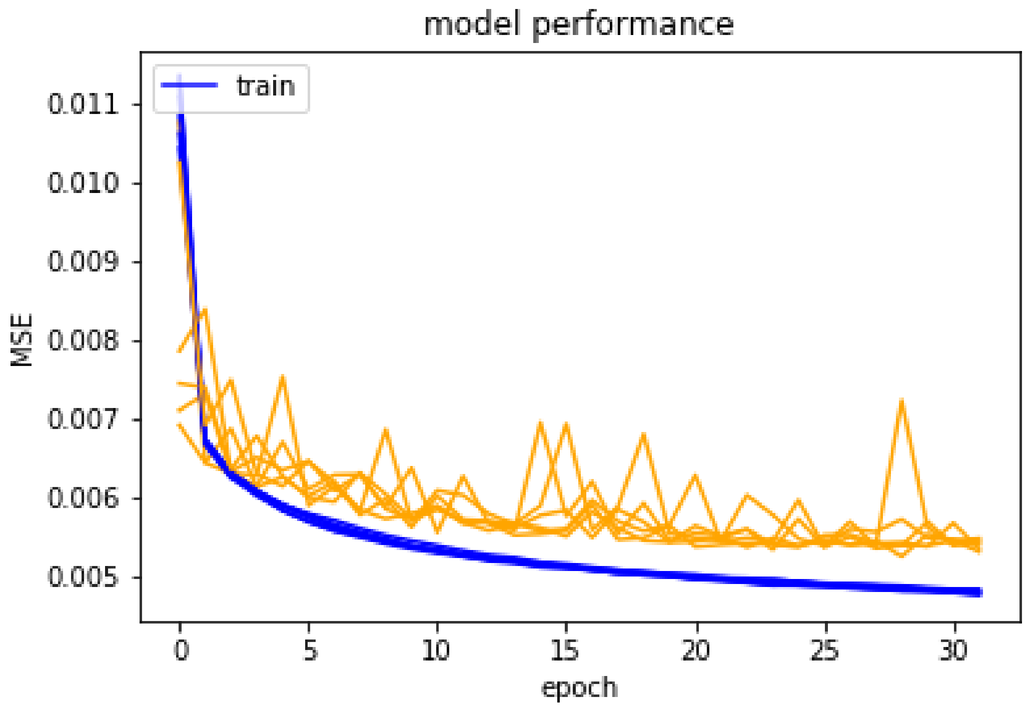

Figure B.3: Train and Test Losses of Two-layer LSTM

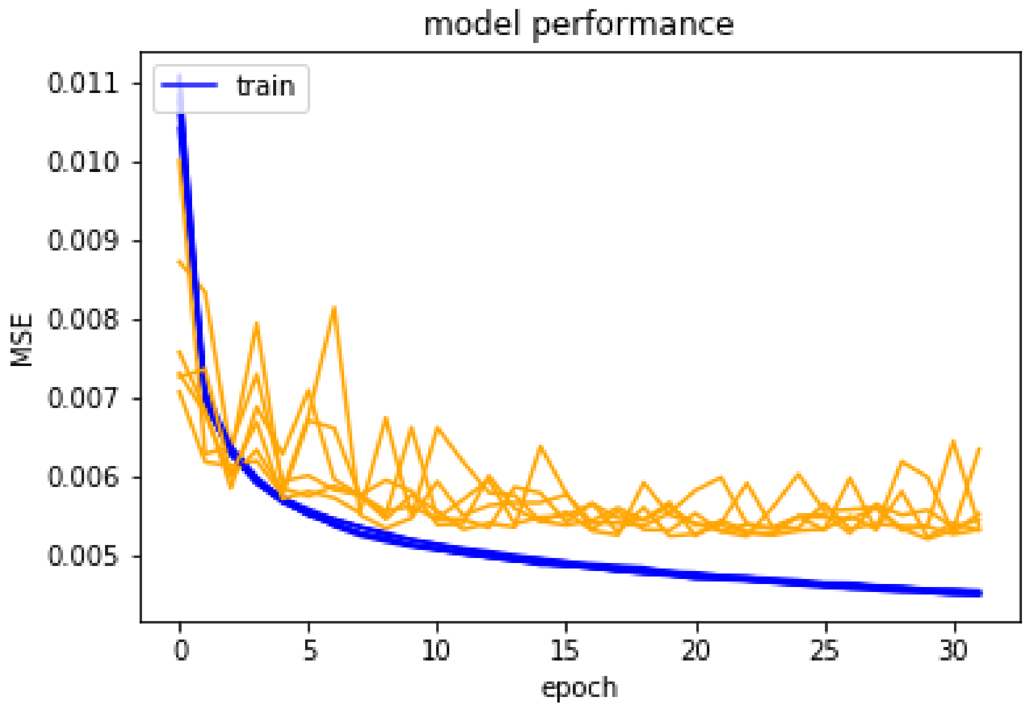

Figure B.4: Train and Test Losses of Two-layer GRU

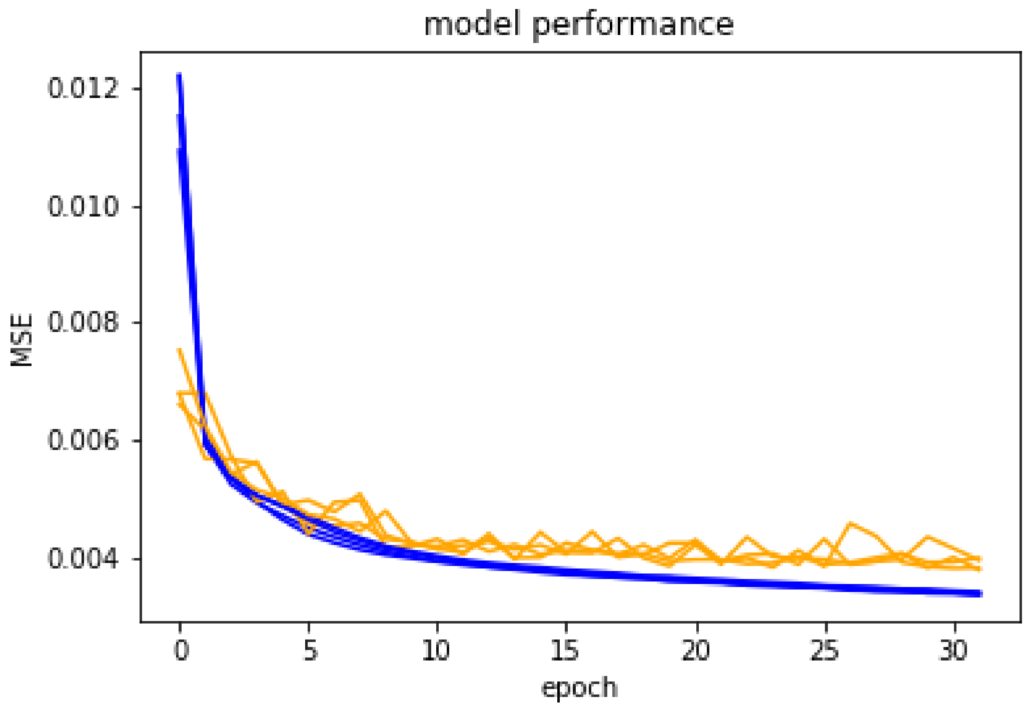

Figure B.5: Train and Test Losses of Model (GLO, TEMP data)

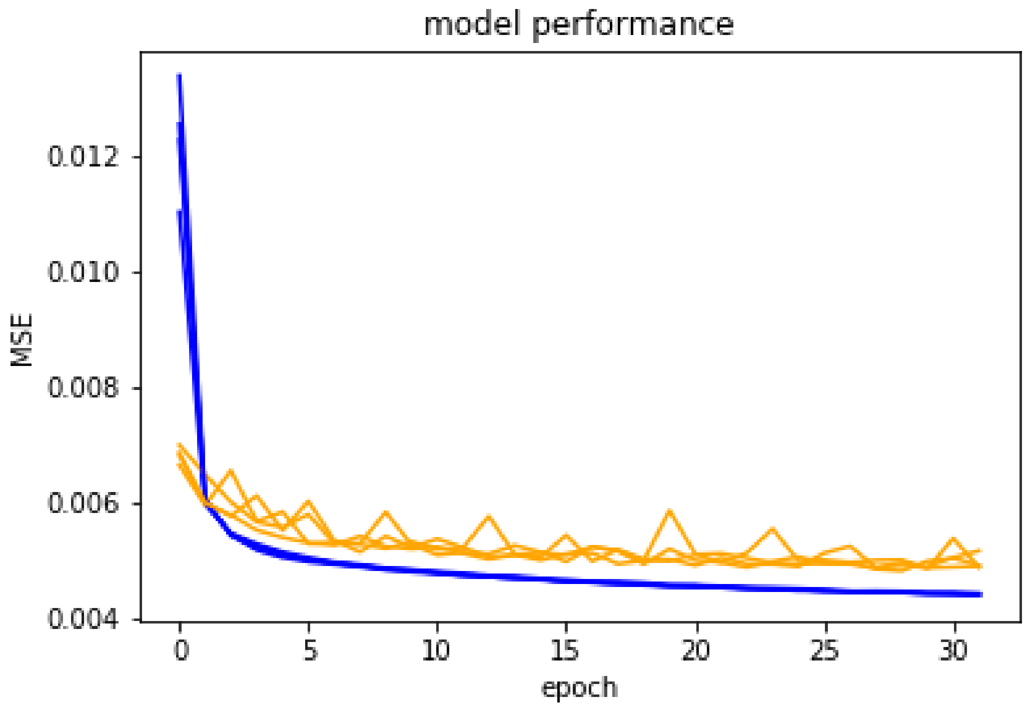

Figure B.6: Train and Test Losses of Model (GLO, PRES data)

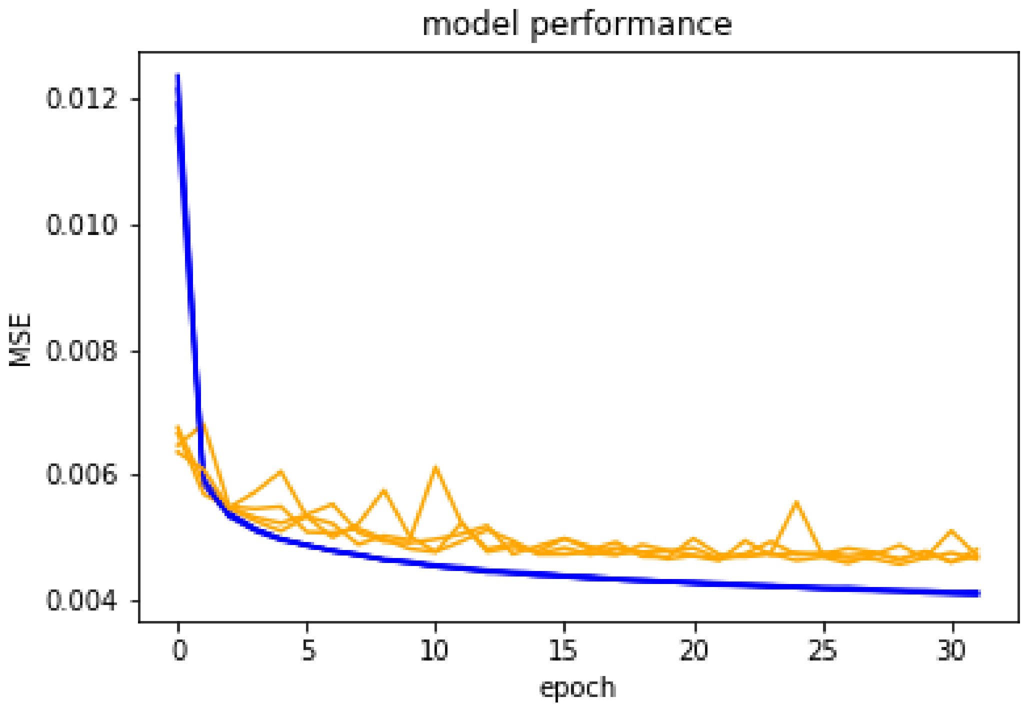

Figure B.7: Train and Test Losses of Model (GLO, HUM data)

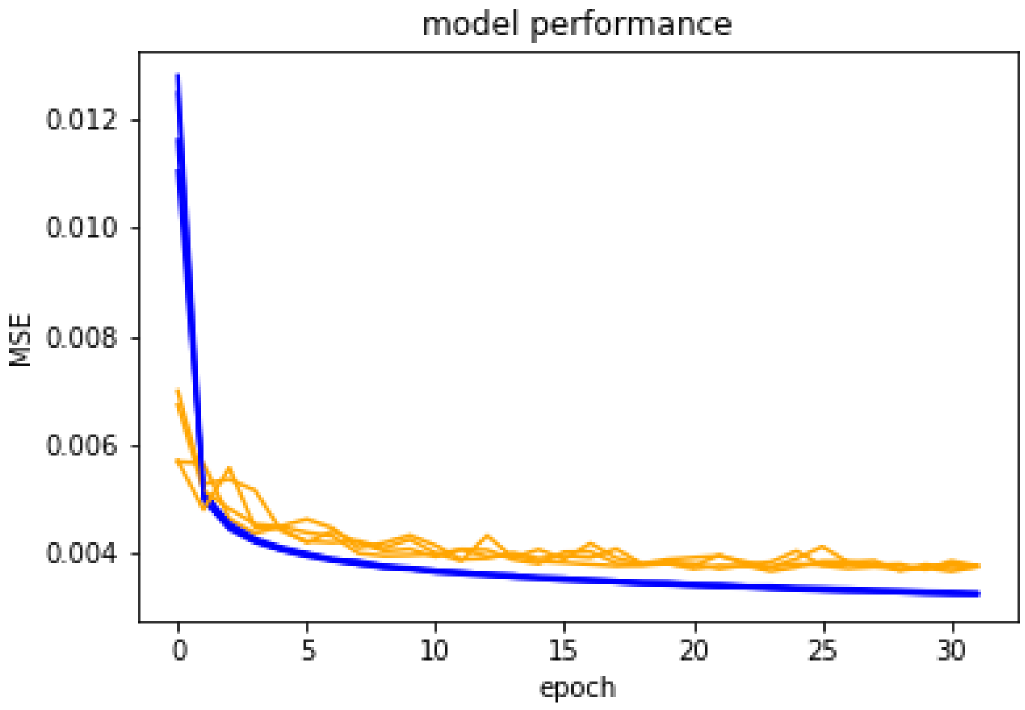

Figure B.8: Train and Test Losses of Model (GLO, NEB data)

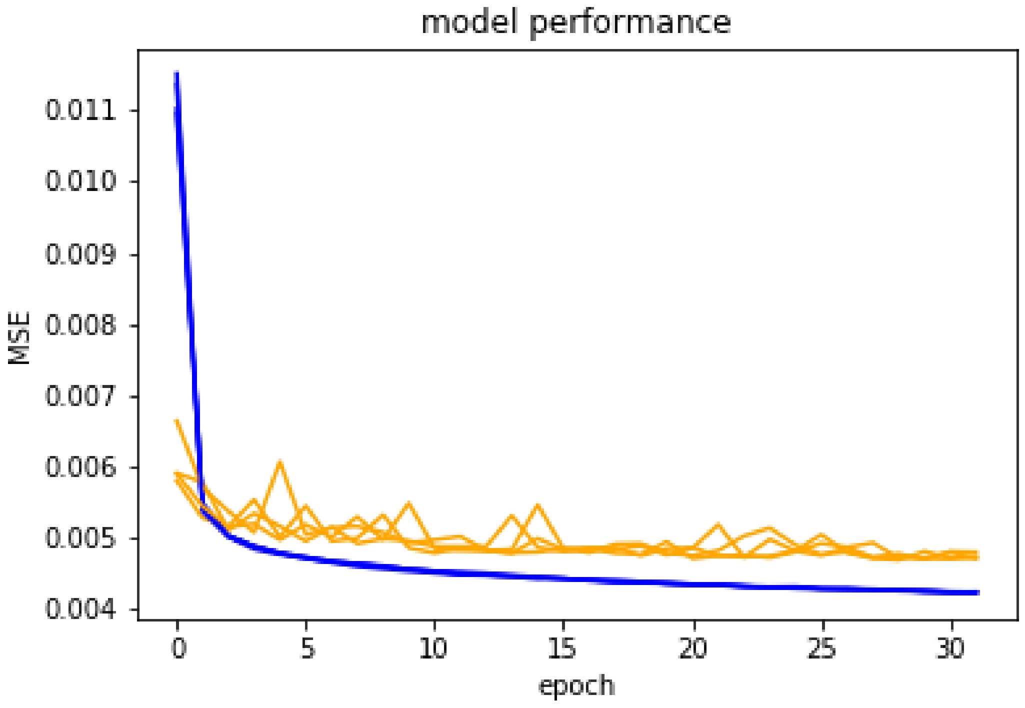

Figure B.9: Train and Test Losses of Model (GLO, INS data)

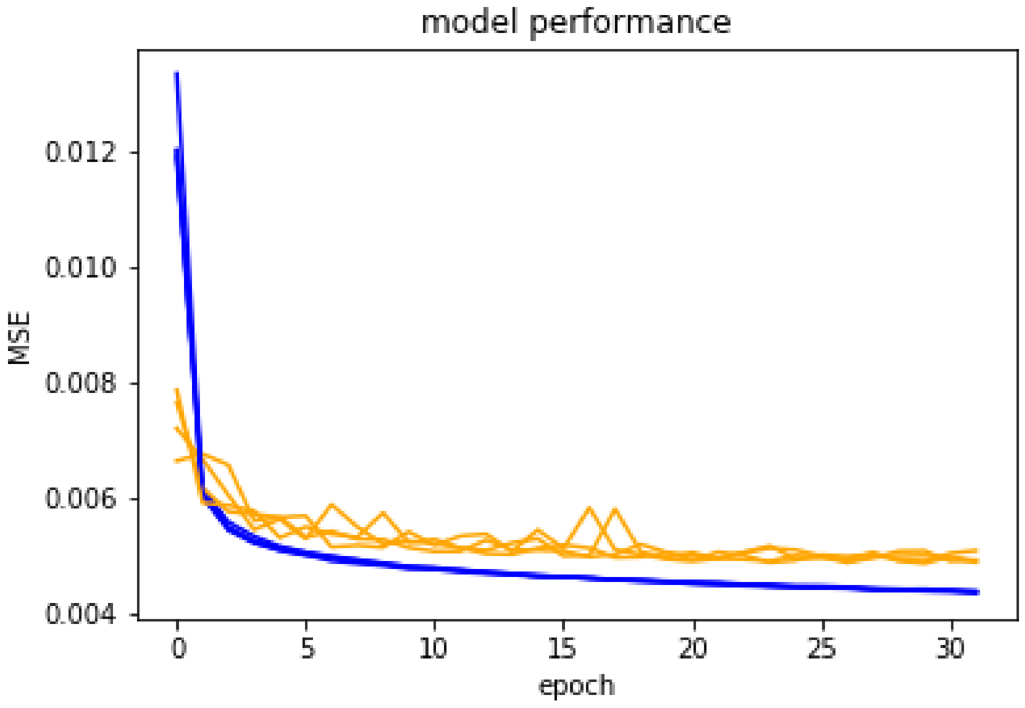

Figure B.10: Train and Test Losses of Model (GLO, WS, WD data)

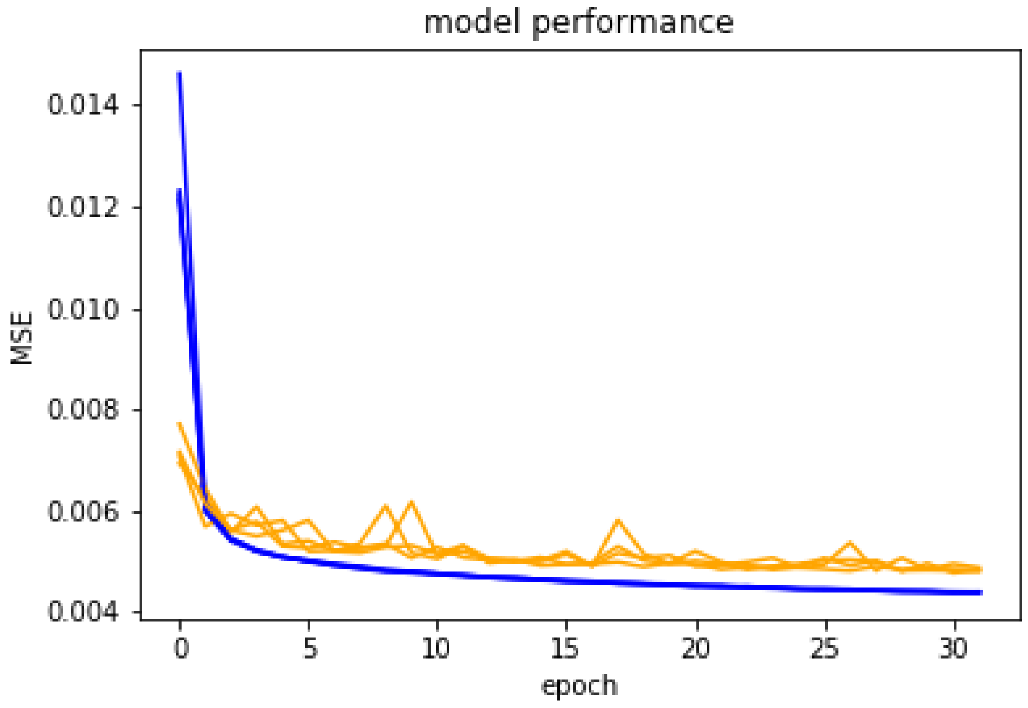

Figure B.11: Train and Test Losses of Model (GLO, RAIN data)

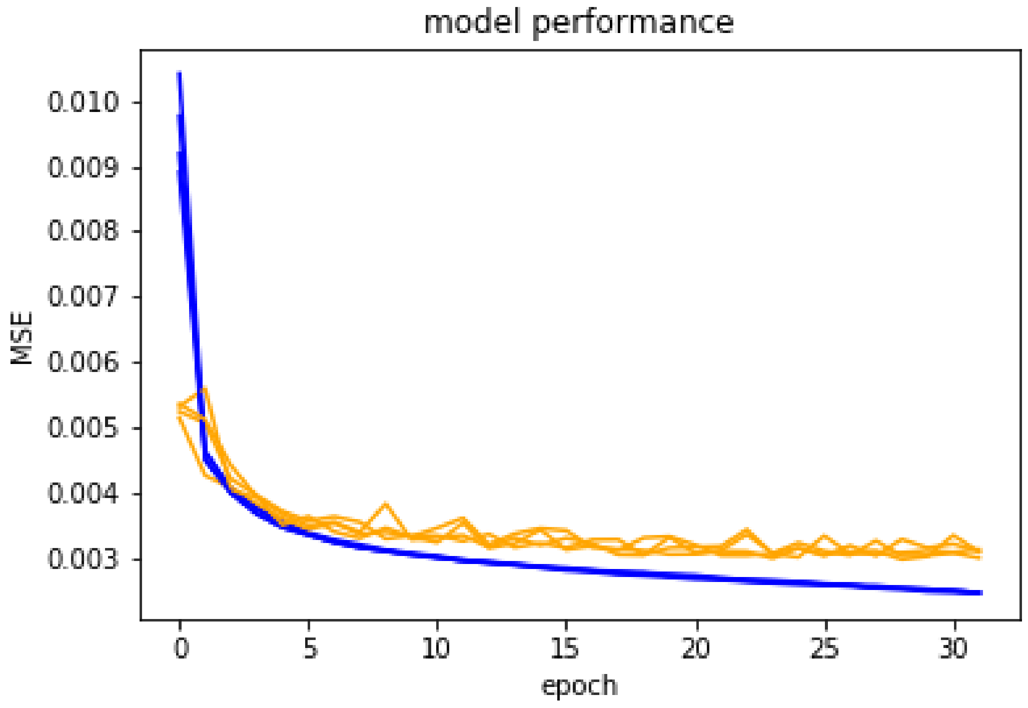

Figure B.12: Train and Test Losses of Model (GLO, TEMP, HUM data)

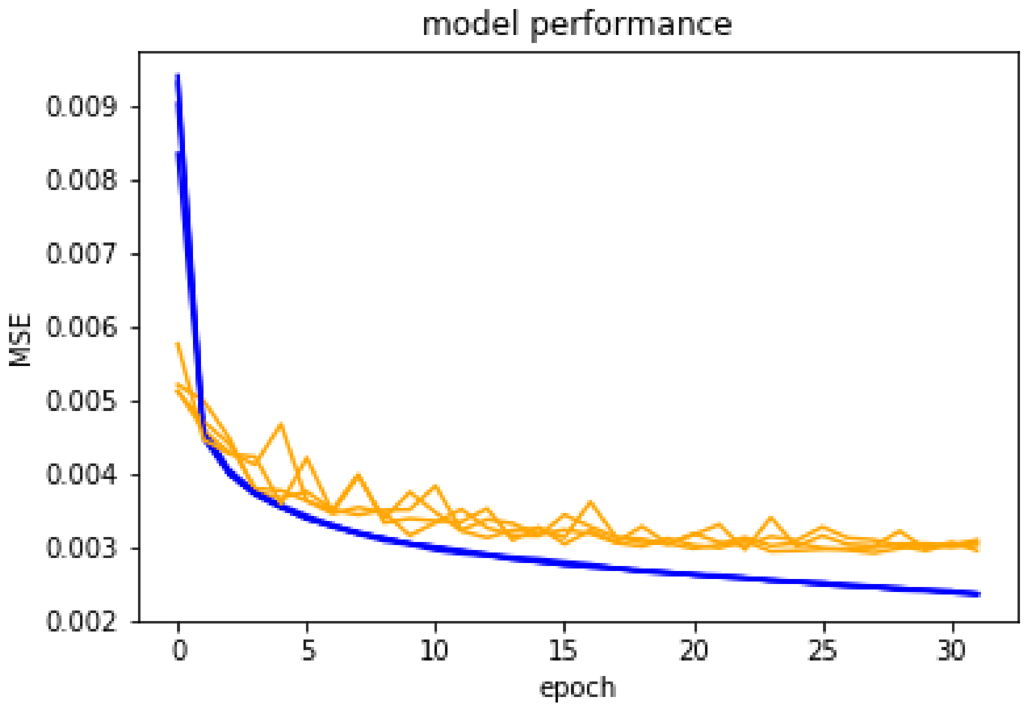

Figure B.13: Train and Test Losses of Model (GLO, TEMP, NEB, HUM data)

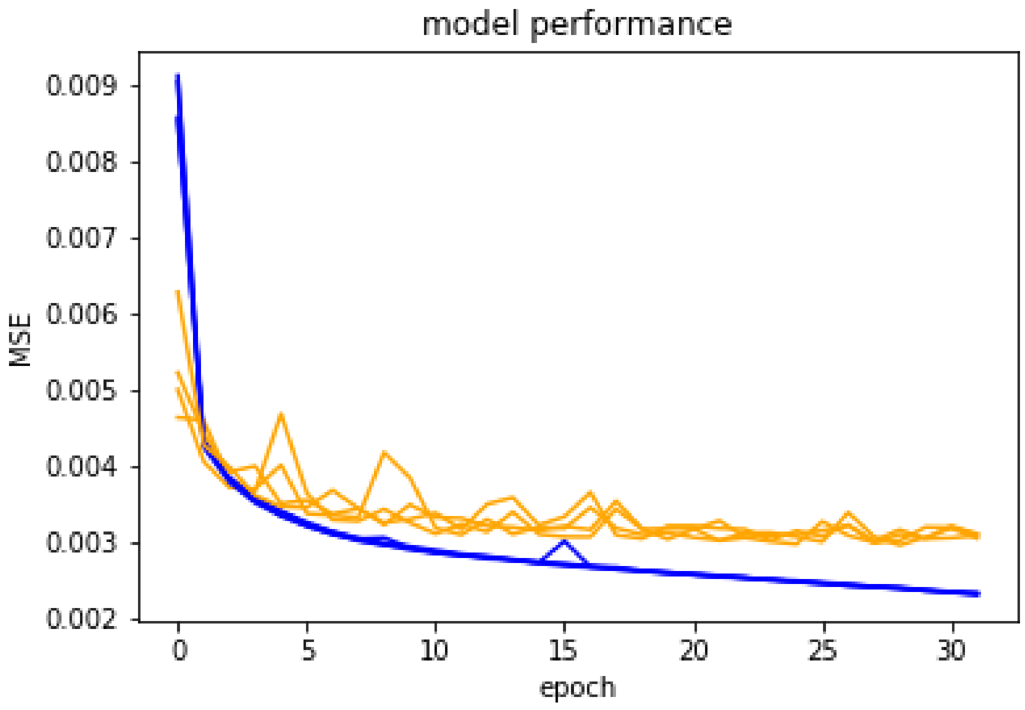

Figure B.14: Train and Test Losses of Model (GLO, TEMP, NEB, INS data)

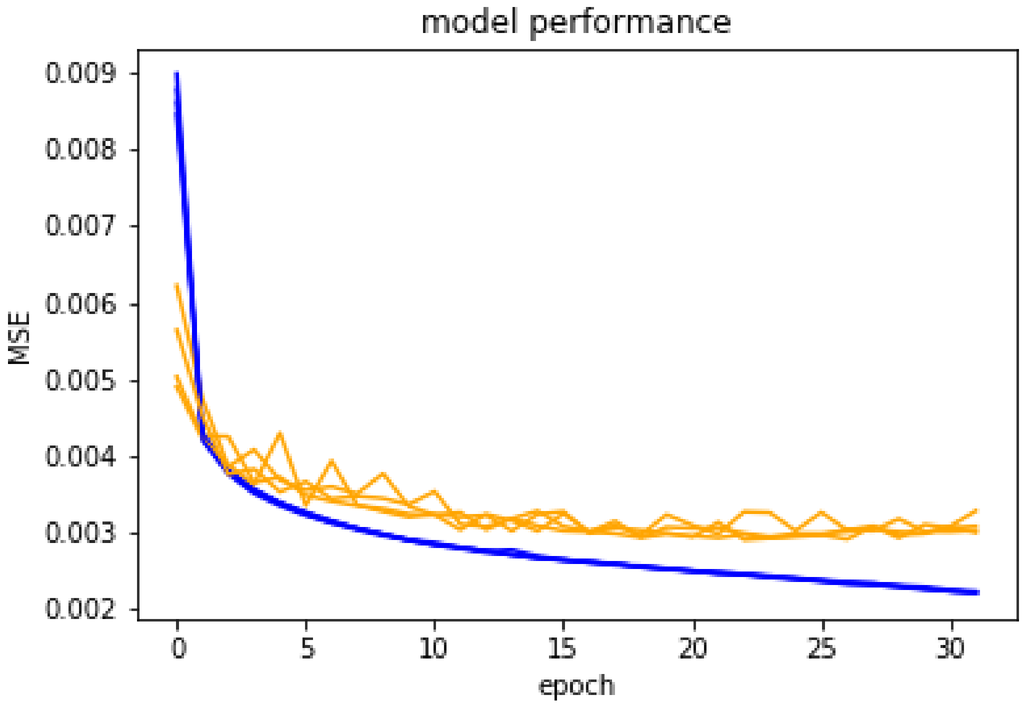

Figure B.15: Train and Test Losses of Model (GLO, TEMP, NEB, HUM, INS data)

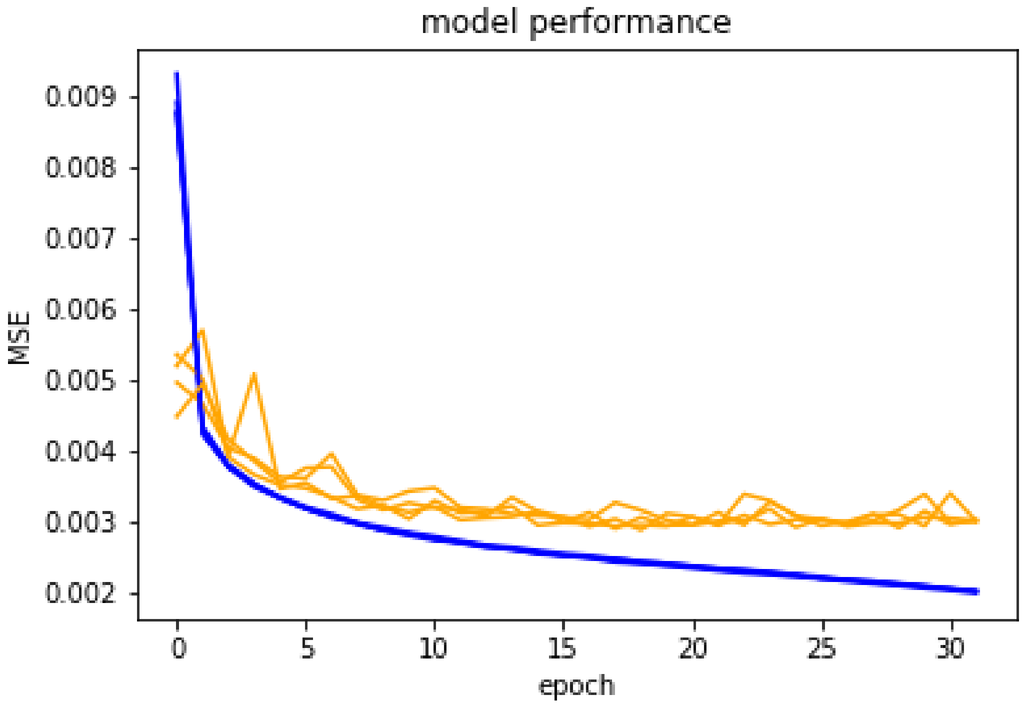

Figure B.16: Train and Test Losses of Model (Full data)

BIOGRAPHICAL SKETCH

Murat Cihan Sorkun was born in Konya in 1988. He studied at Konya Meram Science High School. He received the B.S. degree in Computer Engineering from Istanbul Technical University, in 2012. He has been studying M.S. at Galatasaray University Computer Engineering Department since 2014. He has been working as R&D Engineer at Idea Technology Solutions since August 2015.

PUBLICATIONS

- Sorkun, M. (2016) "Secondhand Car Price Estimation Using Artificial Neural Network" *International Artificial Intelligence and Data Processing Symposium*
- Sorkun, M. and Bayar, S. (2017). “Fault Tolerant Software Architecture applied on Embedded System using Dual Modular Redundancy” *Dokuz Eylul University Faculty of Engineering Journal of Science and Engineering,* vol: 19 no: 55.1 pp. 63-70 Jan 2017
- Sorkun, M.C. (2017) "Customer Behavior Analysis via Electronic Financial Data" *Akademik Bilişim'17*
- Sorkun, M.C., Toraman, T. (2017) "Fraud Detection on Financial Statements Using Data Mining Techniques" *International Journal of Intelligent Systems and Applications in Engineering,* vol: 5 no: 3 pp. 132-134
- Sorkun, M.C., Ozbey, C. (2017) "Compression Experiments on Term-Document Indexes" *2nd International Conference on Computer Science and Engineering*
- Sorkun, M.C., Paoli, C., İncel Durmaz, Ö. (2017). "Time Series Forecasting on Solar Irradiation Using Deep Learning", *International Conference on Electrical and Electronics Engineering.*