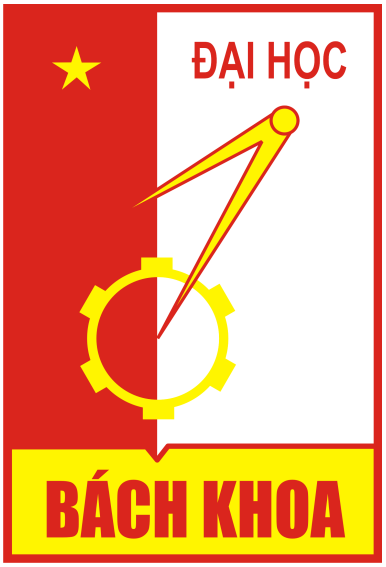
Hanoi University of Science and Technology

The School of Information and Communication Technology



Capstone Project

Deep Learning and Its Applications

Topic: Panama Load Balancing Forecasting

Supervisors:	Assoc. Prof. Nguyễn Thị Kim Anh	
	Prof. Trần Việt Trung	
Students:	Tạ Hữu Bình	20190094
	Trần Trọng Hiệp	20190051
	Nguyễn Văn Trung	20190071
	Hà Đức Tuấn	20190072

Class: 131402

Module: IT4490

Hà Nội, February 3rd 2023

Contents

Introduction

The capacity to estimate Panama's electric load is a crucial issue that has significant effects on the stability and dependability of the power system. Accurate forecasts of future power consumption can aid utilities and governmental organizations in improved resource planning and allocation, ensuring that the supply of electricity matches the demand.

In recent years, deep learning has emerged as a powerful tool for solving complex prediction problems, including time series prediction. Deep learning techniques, such recurrent neural networks (RNNs) and convolutional neural networks (CNNs), are particularly suited for time series prediction issues because they can learn intricate patterns and connections in the data.

Deep learning may be used to forecast future power usage for Panama's electrical load based on existing data. In comparison to conventional time series approaches, it is feasible to increase accuracy and dependability by utilizing the power of deep learning.

The motivation for using deep learning for the prediction of Panama's electric load is to improve the stability and reliability of the power system by providing accurate and reliable forecasts of future electricity consumption. Deep learning makes it feasible to identify intricate connections and patterns in the data, which enhances performance above conventional time series techniques.

In conclusion, using deep learning to the forecast of Panama's electric demand is a promising strategy that has the potential to raise the system's accuracy, dependability, and efficiency. Power providers in Panama are able to guarantee a consistent and long-lasting supply of electricity for the people and businesses of the nation by utilizing the capabilities of deep learning algorithms.

1 Problem

Out report will proposing different models to forecast the Panama load balancing, formulated as a multivariate time series problem [1]. Some taxonomies need to be clarified in this problem

1. Inputs vs. Outputs

   A prediction problem involves using past observations to predict or forecast one or more possible future observations. The goal is to guess about what might happen in the future. The input of our problems is information in some last hours before each prediction, which is called "history time steps". On the other hand, the output is the forecast value of national electricity load in some next hours, which is called "future time steps".

2. Endogenous vs. Exogenous

   An input variable is endogenous if it is affected by other variables in the system and the output variable depends on it. In our problem, we modified that the information of school day and holiday is exogenous since it does not depends on other attributes. Holiday remains the same in every years. All other attributes are endogenous.

3. Regression vs. Classification

   The domain of national electricity load is continuous and our problem is the regression one. We forecast a numerical quantity.

4. Univariate vs. Multivariate

   Multiple variables measured over time and used to forecast, thus, our problem is multivariate.

5. Single-step vs. Multi-step

   We forecast more than one future time steps. Our problem, therefore, is a multi-step time-series problem.

6. Contiguous vs. Discontiguous

   A time series where the observations are not uniform over time is described as discontiguous. The lack of uniformity of the observations may be caused by missing or corrupt values. Our time series problem has uniform observations over time, all measurement is taken after one hour, and no data is loss. Therefore, it is contiguous.

2 Data

We use the data 'Panama Electricity Load Forecasting' [2] to observe the performance of different multivariate time series forecasting models. This data set contains the official Panama electricity load information with reference to Weather Parameters and Special Days from weekly pre-dispatch reports.

2.1 Description

The original data sources provide the post-dispatch Panama electricity load in individual Excel files on a daily basis and weekly pre-dispatch electricity load forecast data in individual Excel files on a weekly basis, both with hourly granularity. Holidays and school periods is sparse data. Weather data is available on daily NetCDF files.

Data sources provide hourly records. The data composition is the following:

- Historical electricity load is available on daily post-dispatch reports, from the grid operator (CND).
- Historical weekly forecasts are available on weekly pre-dispatch reports, both from CND.
- Calendar information related to school periods, taken from Panama's Ministery of Education.
- Calendar information related to holidays from "When on Earth?" website.
- Weather variables, such as temperature, relative humidity, precipitation, and wind speed for three main cities in Panama is taken from Earthdata.

The final dataset is pre-processed by merging all data sources on the date-time index. Loading of the Panama in 'nat_demand' column is the targets to forecast. The remainder includes 12 features of numerical continuous values referring to weather parameters and 2 features providing information of whether the corresponding days is holiday or schoolday or not. Meaning of each column are:

- 'datetime': Date-time index corresponding to Panama time-zone UTC-05:00 (index)
- 'nat_demand': National electricity load (Targetor Dependent variable) (MWh)
- 'T2M_toc': Temperature at 2 meters in Tocumen, Panama city (^{o}C)
- 'QV2M_toc': Relative humidity at 2 meters in Tocumen, Panama city (%)

- 'TQL_toc': Liquid precipitation in Tocumen, Panama cityliter (s/m2)
- 'W2M_toc': Wind Speed at 2 meters in Tocumen, Panama city (m/s)
- 'T2M_san': Temperature at 2 meters in Santiago city (ºC)
- 'QV2M_san': Relative humidity at 2 meters in Santiago city (%)
- 'TQL_san': Liquid precipitation in Santiago city (l/m2)
- 'W2M_san': Wind Speed at 2 meters in Santiago city (m/s)
- 'T2M_dav': Temperature at 2 meters in David city (ºC)
- 'QV2M_dav': Relative humidity at 2 meters in David city (%)
- 'TQL_dav': Liquid precipitation in David city (l/m2)
- 'W2M_dav': Wind Speed at 2 meters in David city (m/s)
- 'holiday': Holiday binary indicator (1 = holiday, 0 = regular day)
- 'school': School period binary indicator

2.2 Data visualization

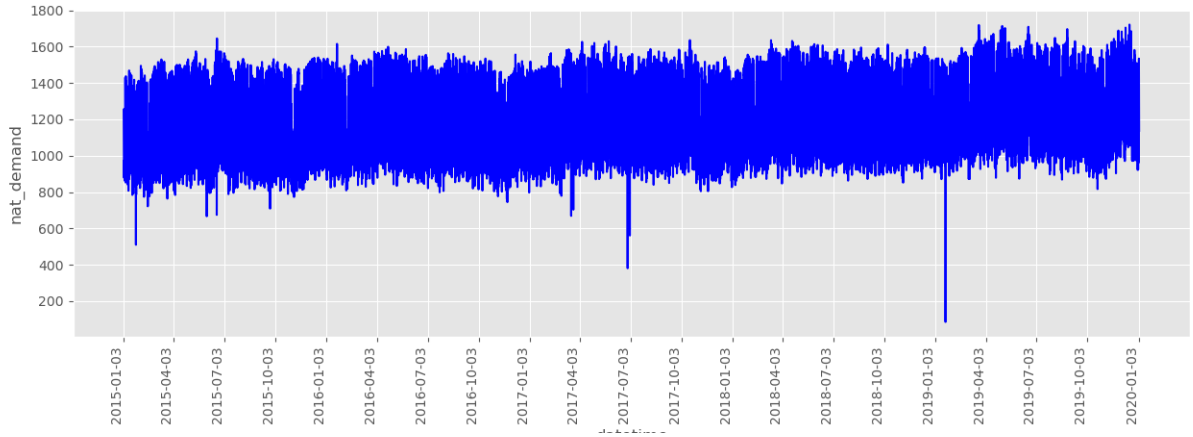


Fig. 1: National electricity load through time.

National electricity load, which is the forecast target, has a slightly up trend and the amount are not too different between days though some data points seems strangely low. All other features has at least yearly cycle. The data in 'holiday' and 'schoolday' columns are exactly the same in the same days of different years.

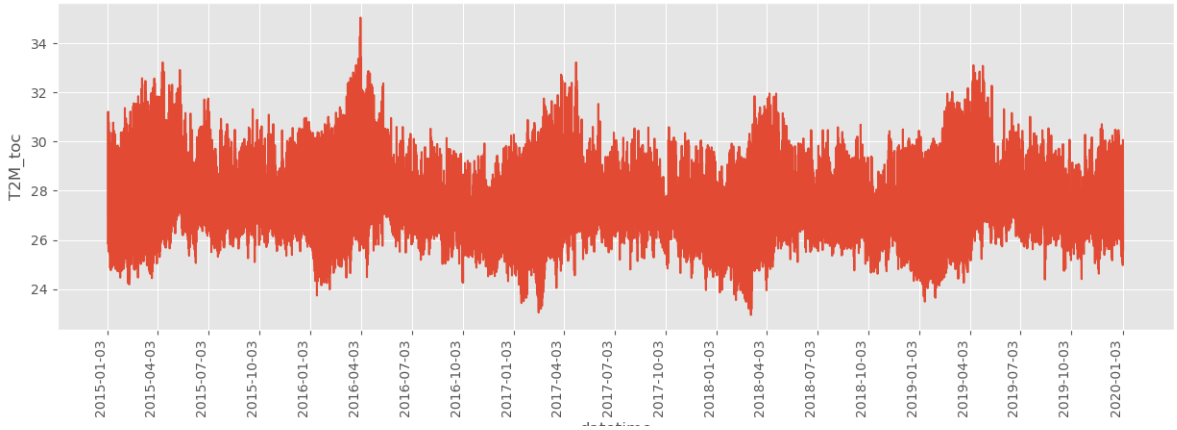


Fig. 2: Temperature at 2 meters in Tocumen through time.

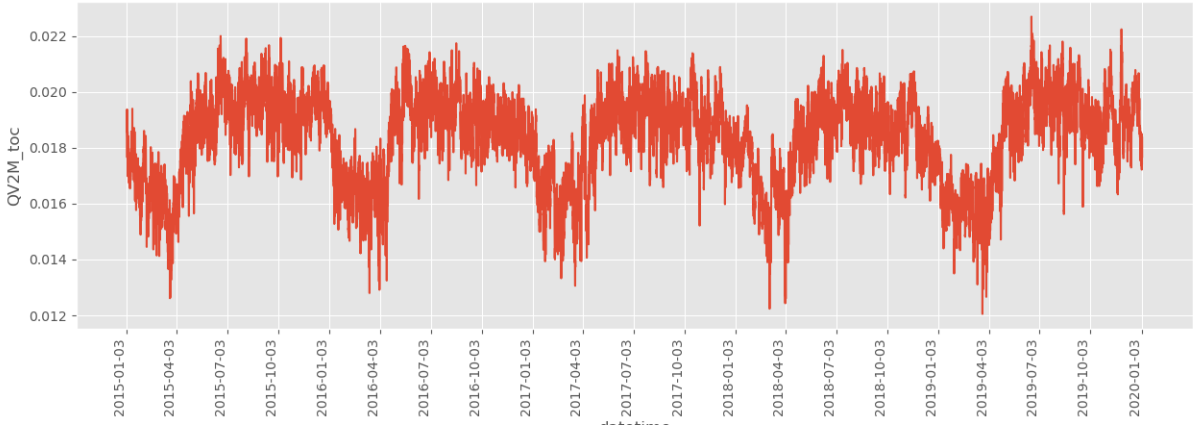


Fig. 3: Relative humidity at 2 meters in Tocumen through time.

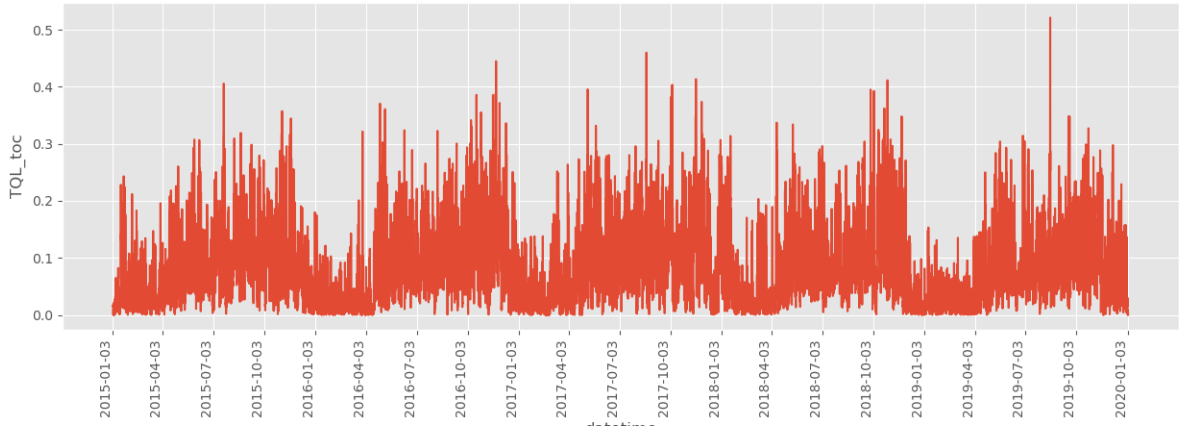


Fig. 4: Liquid precipitation in Tocumen through time.

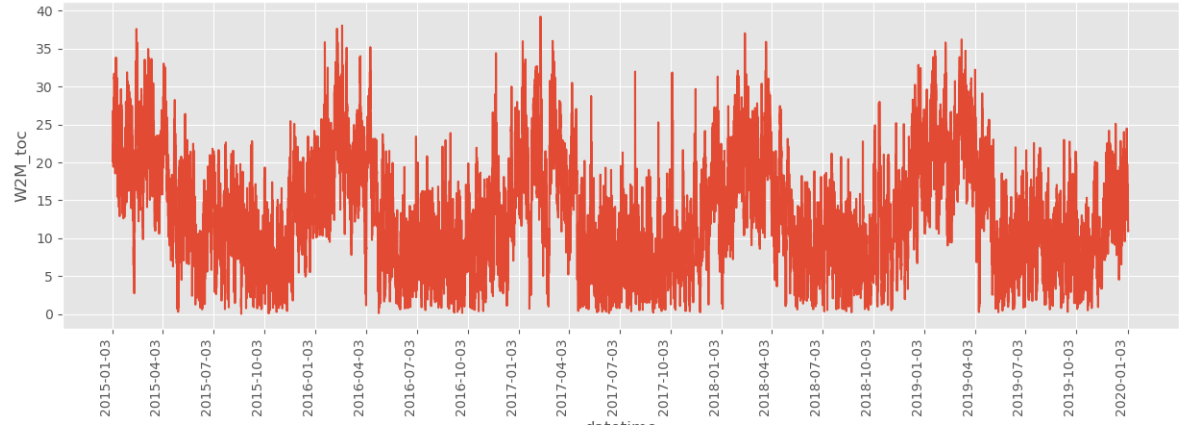


Fig. 5: Wind Speed at 2 meters in Tocumen through time.

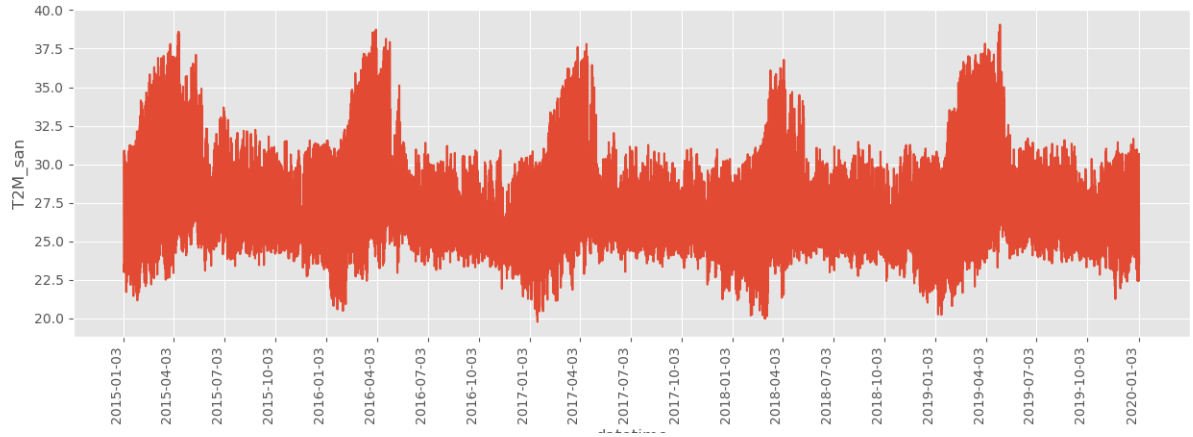


Fig. 6: Temperature at 2 meters in Santiago through time.

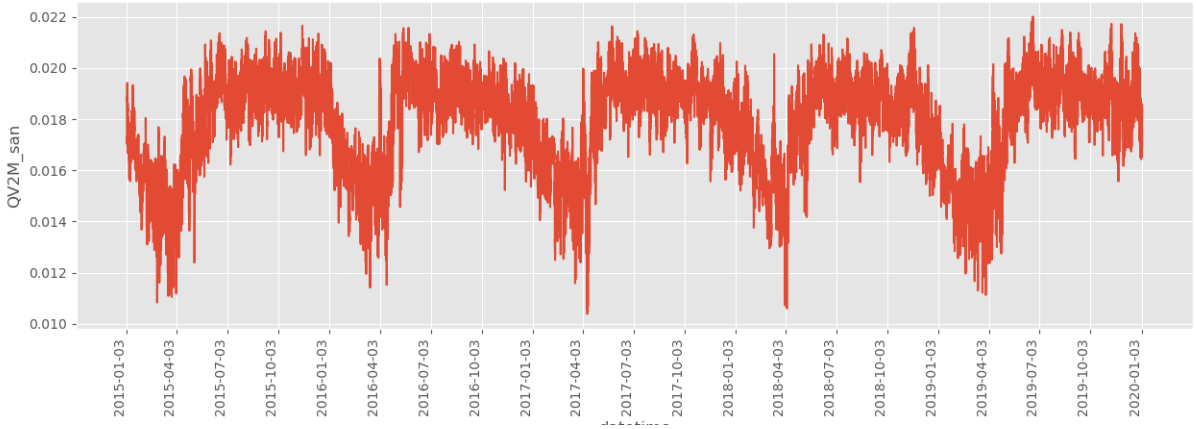


Fig. 7: Relative humidity at 2 meters in Santiago through time.

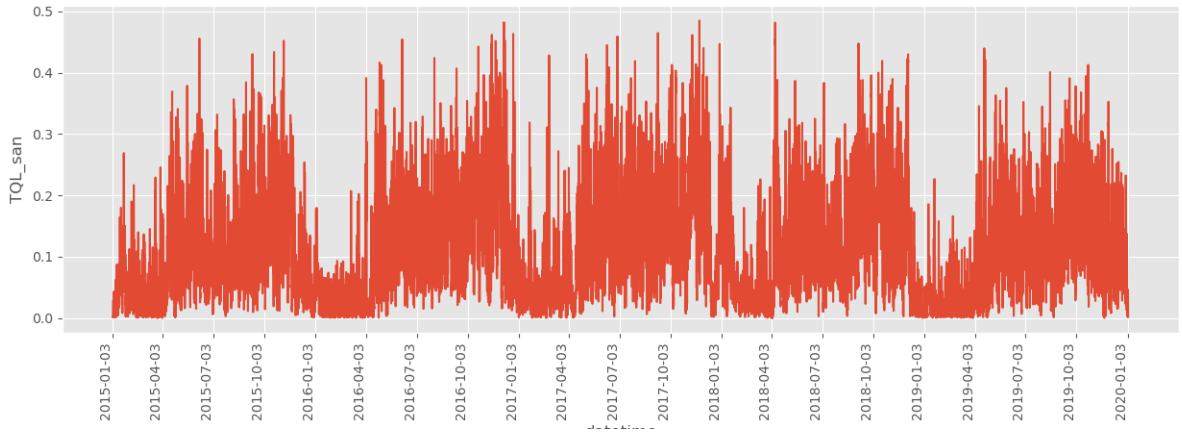


Fig. 8: Liquid precipitation in Santiago through time.

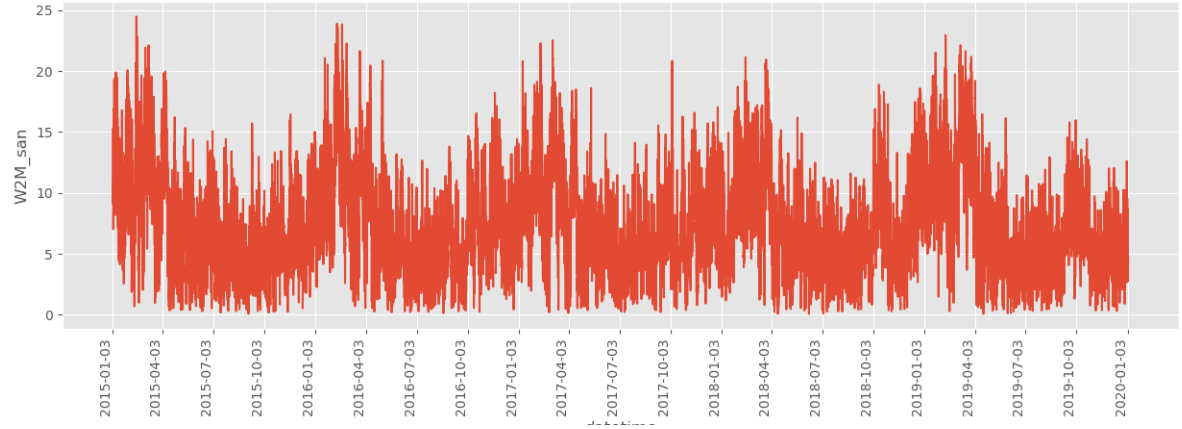


Fig. 9: Wind Speed at 2 meters in Santiago through time.

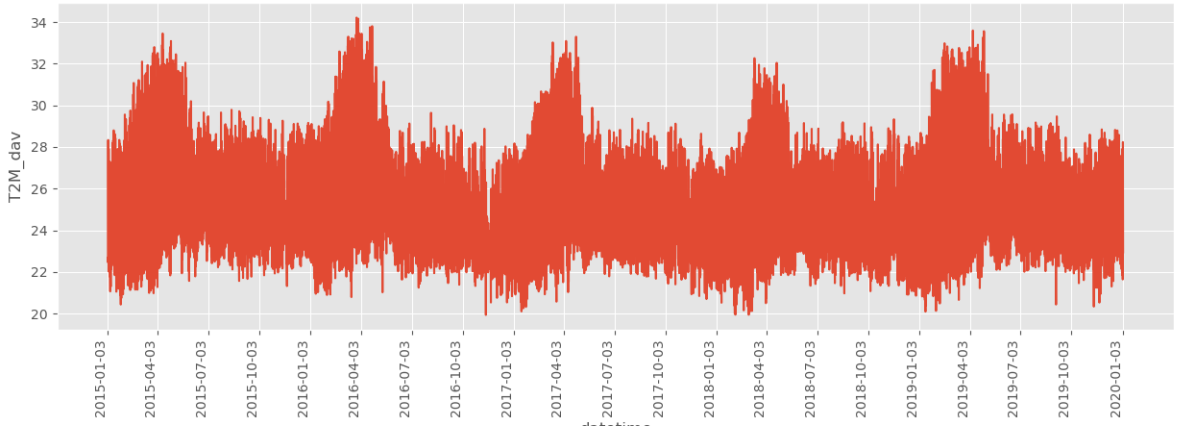


Fig. 10: Temperature at 2 meters in David through time.

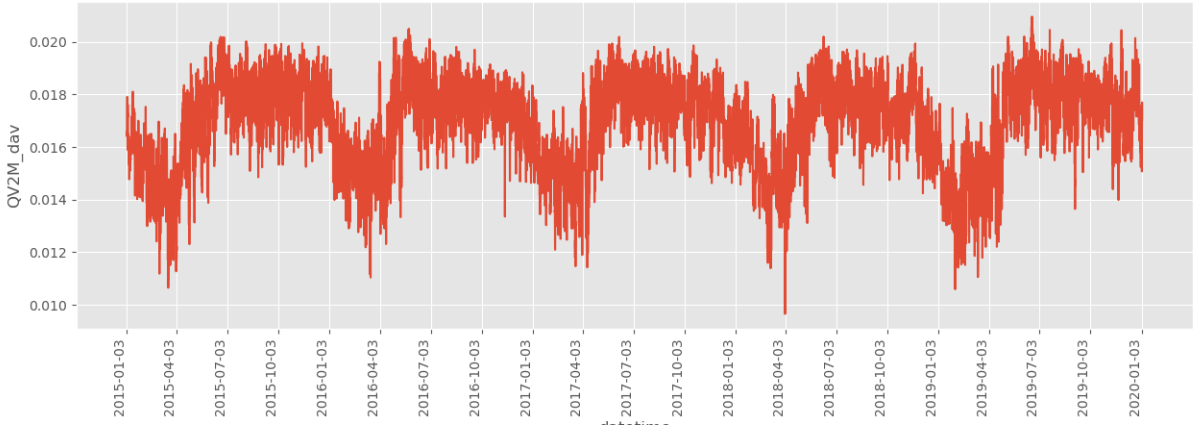


Fig. 11: Relative humidity at 2 meters in David through time.

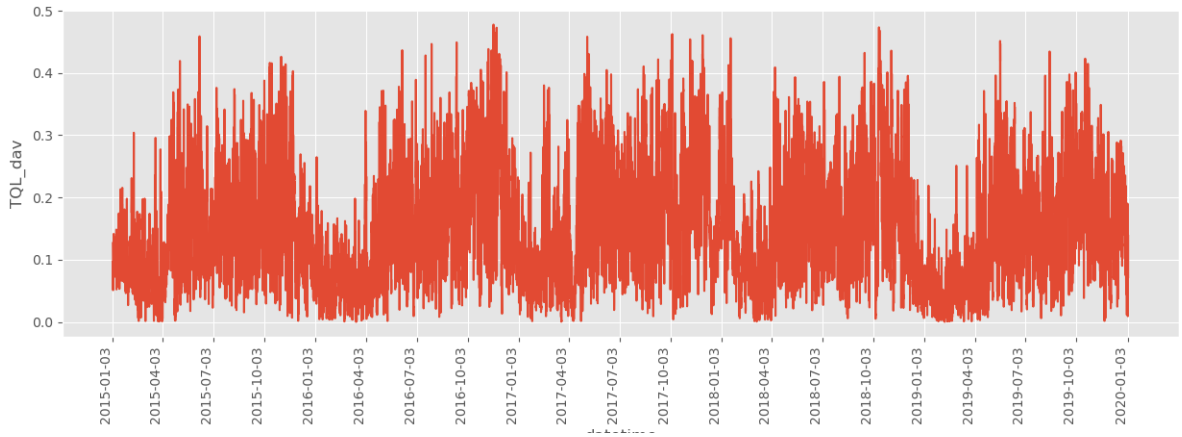


Fig. 12: Liquid precipitation in David through time.

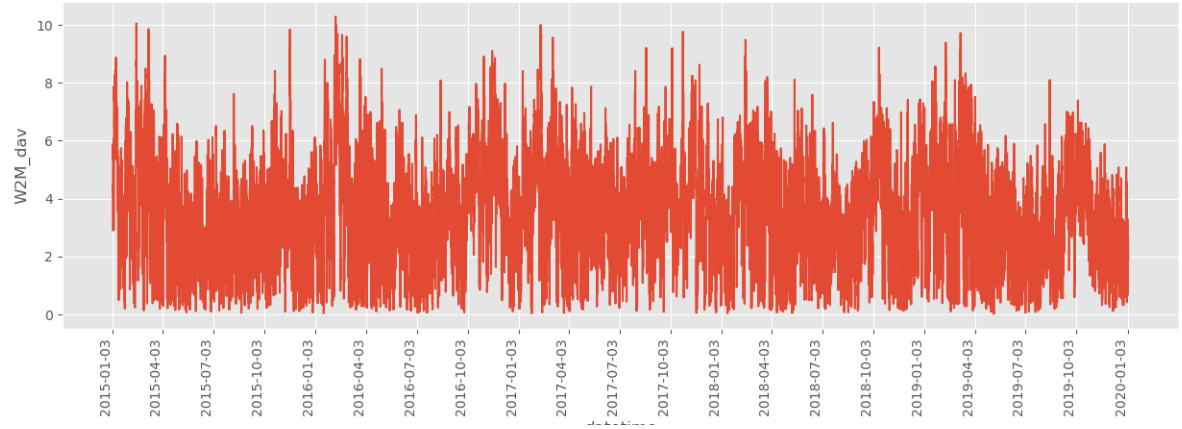


Fig. 13: Wind Speed at 2 meters in David through time.

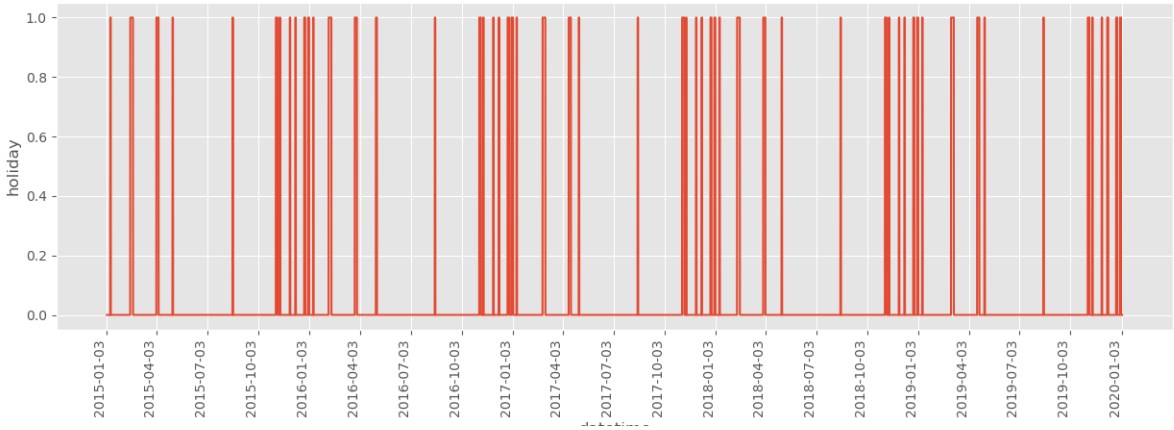


Fig. 14: Holiday through time.

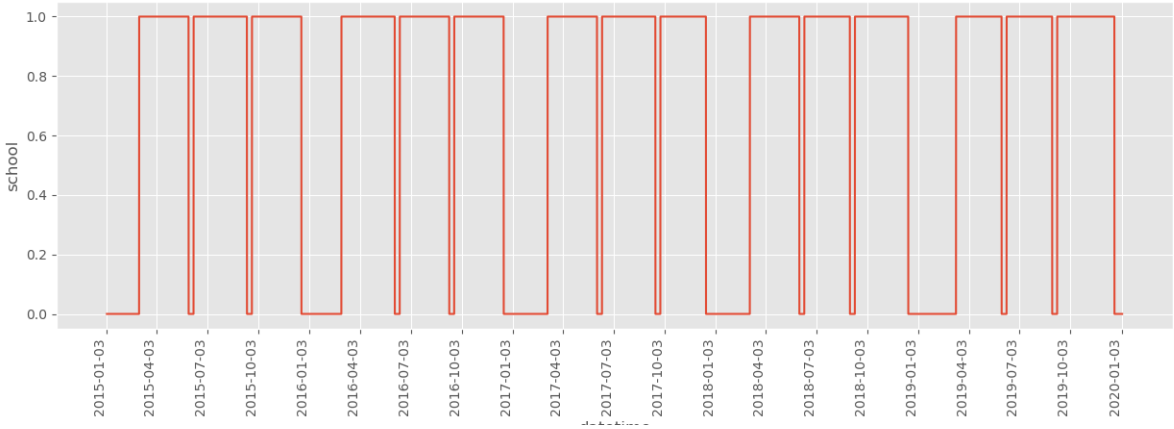


Fig. 15: Schoolday through time.

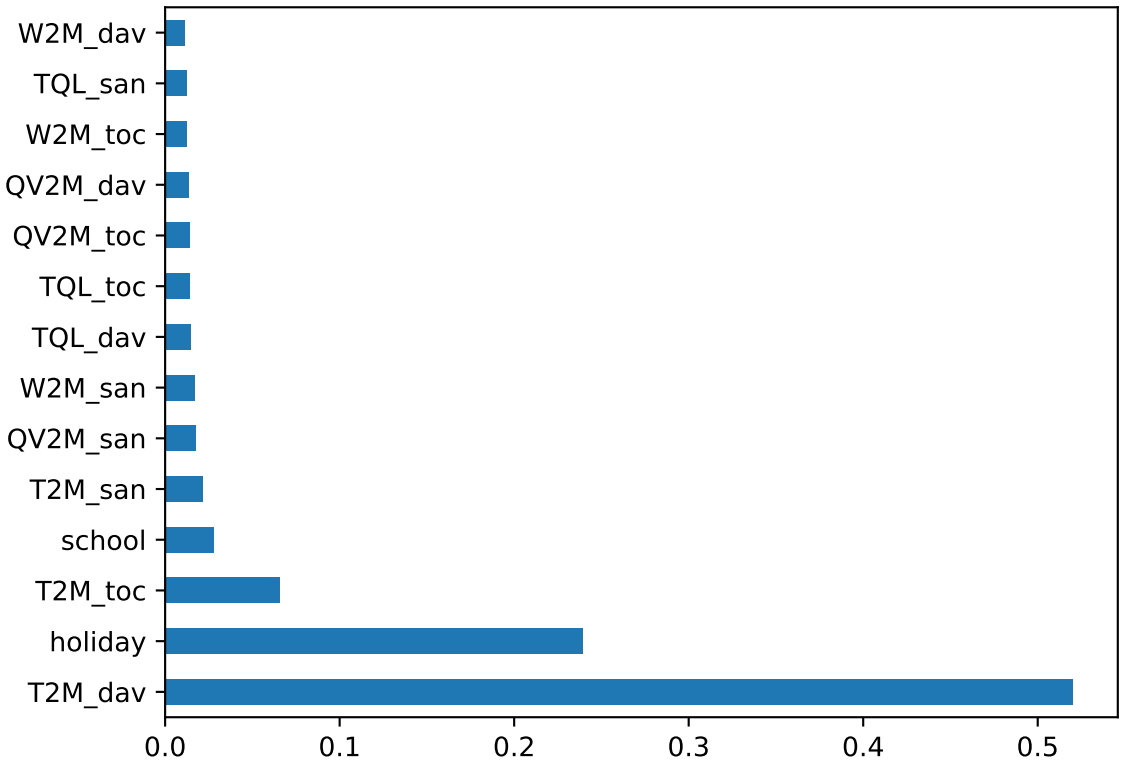


Fig. 16: Feature Importance using XGBoost.

Fig. (16) shows the importance of each features following XGBoost. Temperature at 2 meters plays very essential roles, expecially the ones in Tocumen and David city. All other weather features play the similar roles. Binary and definite stationary data of schoolday and especially holiday affect a lot to the electricity load.

3 Proposed multivariate time series models

3.1 Vector Autoregression

Vector Autoregression (VAR) is a statistical model used to capture the interdependence between multiple time-series variables. The model is widely used in econometrics and finance to analyze and forecast the relationships between variables such as inflation, exchange rates, and stock prices.

VAR models work by assuming that each time-series variable is influenced by its past values and the past values of other variables in the model. The model can then be used to forecast future values of each variable based on its past values and the past values of other variables in the model. This makes VAR models useful for analyzing complex relationships between multiple time-series variables.

One advantage of VAR models is their simplicity and ease of use. The model is straightforward to implement and requires only a minimal amount of prior knowledge about the relationships between the variables. Additionally, VAR models can be easily extended to include more variables or to incorporate exogenous variables, such as economic indicators or demographic data.

However, there are also some limitations to VAR models. One of the main limitations is that VAR models assume that the relationships between variables are linear, which may not always be the case in real-world data. Additionally, VAR models can be sensitive to the choice of lag order, which can affect the results and the accuracy of the model.

3.2 Facebook Prophet

Facebook Prophet (FbProphet) is a time series forecasting library developed by Facebook. It is designed for analyzing time series data that has multiple seasons and trends, making it suitable for a wide range of applications such as predicting sales, website traffic, and stock prices. Prophet uses a decomposable model that separates the trend, seasonality, and holiday components of a time series, allowing it to make accurate predictions even with limited historical data.

The basic math used in the prophet:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

$g(t)$: linear or logistic growth with respect to time series data.

$s(t)$: seasonal component.

$h(t)$: effect of holiday

ϵ_t : error term caused due to unexpected occurrences.

Prophet's core procedure is implemented using Stan (a probabilistic programming language). Stan performs map optimization to find parameters and facilitates estimating parameter uncertainty using the Hamiltonian Monte Carlo algorithm.

Some advantages of Prophet:

- Accurate and fast
- Allows adjustment of parameters, customized seasonality components which may improve the forecasts
- Also handle outliers and handles other data issues

However, Facebook Prophet has some limitations:

- Non-linearity: Prophet is designed to handle linear and additive time series models, but it may not perform well with highly non-linear or complex data.
- Holidays and events: Prophet has built-in support for modeling holidays and special events, but it may not be able to handle more complex or unusual events.
- Seasonality: While Prophet can handle multiple types of seasonality, it assumes a fixed period length and may not work well for data with non-stationary seasonality patterns.
- Outliers: Prophet can be sensitive to outliers and may not be the best choice for datasets with a lot of extreme values.
- Limited Modeling Capabilities: Prophet has limited modeling capabilities compared to more advanced time series models and may not be suitable for more complex time series problems.
- Scalability: Prophet is not optimized for large-scale time series forecasting and may struggle with datasets with many time series or long histories.

3.3 Deep Learning

3.3.1 Recurrent Neural Network

Recurrent Neural Networks (RNNs) are a type of deep learning architecture that is widely used in sequential data processing tasks. RNNs are designed to handle sequential data, such as speech signals, text, and time-series data, which have a continuous structure and depend on previous inputs. The architecture of RNNs is designed to allow information to flow through the network, allowing the network to learn and store information from the input sequence.

RNNs can be depicted as illustrated below:

The application of RNNs is widespread in various fields, such as speech recognition, natural language processing, and robotics. In speech recognition, RNNs are used to identify spoken words from speech signals and translate them into written text. In natural language processing, RNNs are used for text classification, sentiment analysis, and machine translation. In robotics, RNNs are used to control the movement of robots based on their inputs and previous experiences.

The advantages of RNNs include their ability to handle sequential data effectively and their ability to store information for future use. Unlike traditional neural networks, RNNs can

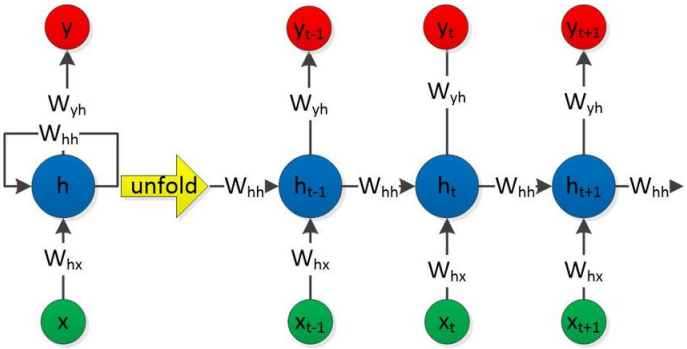


Fig. 17: The architecture of RNN. (source: ResearchGate)

capture long-term dependencies in sequential data and store the information for future use, which helps the network to better understand the context and relationships between inputs. Additionally, RNNs are highly flexible and can be adapted to various tasks and domains, making them a versatile and powerful tool in sequential data processing.

However, there are also some limitations to RNNs. One of the main limitations is the vanishing gradients problem, where the gradients of the loss function become very small as the information is propagated through the network. This can lead to slow convergence or even no convergence of the network. To address this problem, other variants of RNNs have been developed, such as LSTMs and GRUs, which have different structures and mechanisms to capture long-term dependencies in sequential data

3.3.2 Long-Short Term Memory

Long Short-Term Memory is an example of a recurrent neural network (RNN) architecture, and its name is LSTM. The shortcomings of conventional RNNs in addressing long-term dependencies in sequential data were addressed by the development of LSTMs.

The key feature of LSTMs is the memory cell, which allows them to remember information over a long period of time and selectively forget irrelevant information. This makes LSTMs well-suited for a wide range of issues in fields like natural language processing and speech recognition, time series, and the ability to handle sequences of variable durations and capture long-term relationships in the data.

Due to its ability to recognize both short- and long-term relationships in the data, LSTMs are also well suited for time series challenges. In order to enhance predictions of future values based on previous patterns, LSTMs use memory cells to store significant information. This allows them to generate predictions based on patterns in the data.

Although LSTM have many beneficials, LSTMs have several disadvantages that can

limit their practical applications. These include their propensity to overfit on short datasets, which results in subpar generalization performance, and their high computational complexity, which can result in longer training durations and higher processing demands. Due to its abstract internal state and gates, LSTMs can also be challenging to analyze and comprehend, which results in a lack of transparency and explicability. Challenges in some domains might also arise from the large memory requirements of LSTMs and the requirement for meticulous hyperparameter adjustment.

In conclussion, LSTMs are preferred over conventional RNNs for a number of jobs involving sequence data, including time series issues.

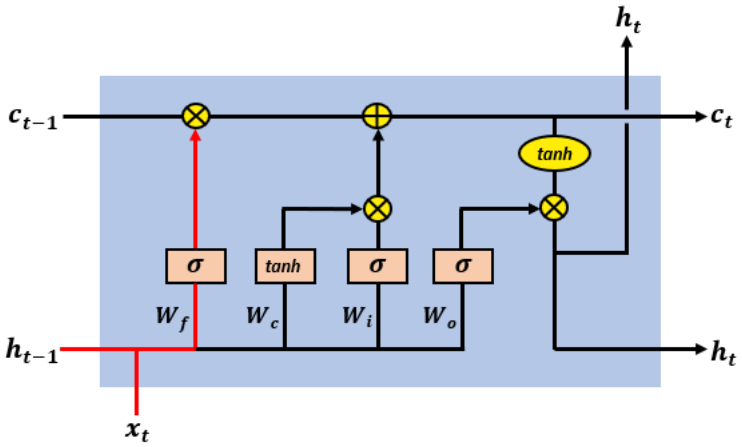


Fig. 18: LSTM (source: medium)

3.3.2.1 Stacked Long-Short Term Memory

A deep recurrent neural network design known as stacked LSTM stacks many LSTM layers on top of one another. In order to capture progressively complicated representations of the input data as it moves through each layer, this type of architecture was developed.

The capacity of Stacked LSTMs to learn hierarchical representations of the input data, with each layer learning a different degree of abstraction, is its essential characteristic. Stacked LSTMs are highly suited for a broad range of issues in fields like natural language processing, speech recognition and time series because of their ability to handle exceedingly big and complicated input sequences as a result.

Stacked LSTMs can effectively capture both short-term and long-term dependencies in the data, making them an excellent choice for time series challenges. Stacked LSTMs can learn more complicated representations of the input sequence by stacking several LSTM

layers, improving predictions of future values based on historical patterns.

Stacked LSTMs, while providing improved performance on certain tasks compared to traditional LSTMs, also have several disadvantages. The number of layers in a stacked LSTM model can greatly increase the amount of computing necessary, which can lead to greater computational complexity and longer training durations. The potential of overfitting on the training data increases as more layers are added, which might be a difficulty. A stacked LSTM's numerous layers and abstract representations make it challenging to comprehend how it functions from the inside out, which is another disadvantage.

In conclussion, stacked LSTMs are a popular option for handling time series issues and other jobs involving sequence data because they have a number of benefits over conventional RNNs and single-layer LSTMs.

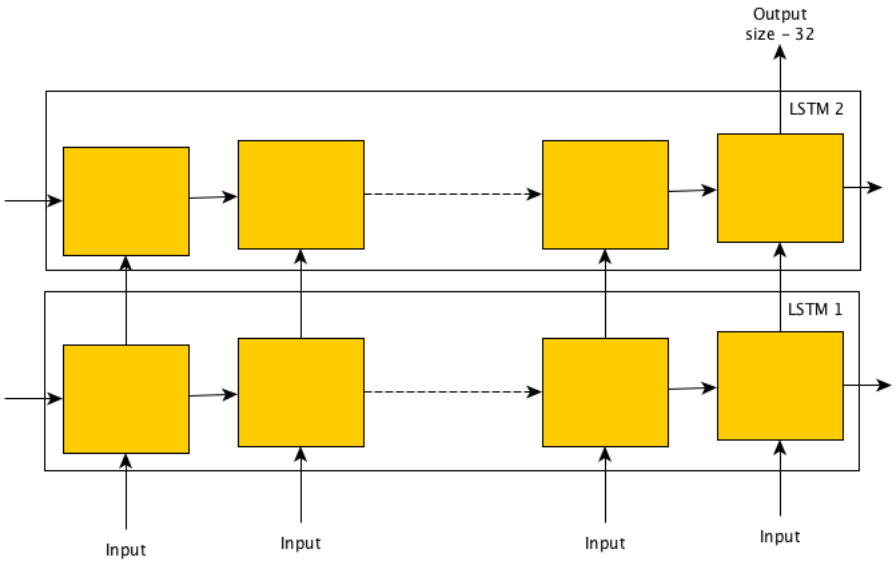


Fig. 19: Stacked LSTM (source: Stackoverflow)

3.3.2.2 Bidirectional Long-Short Term Memory

Bidirectional long-short term memory(Bi-LSTM) is the technique of allowing any neural network to store sequence information in both ways, either backwards (future to past) or forwards (past to future). Unlike standard LSTM, in bi-directional, we can make input flow in both directions to preserve the future and the past information. Nevertheless, normal LSTMs allow input flow in one direction (either forwards or backwards).

BiLSTM is well-suited for solving problems in natural language processing (NLP), speech recognition, and time series prediction. Because it can capture the relationships be-

tween sequenced events in the past as well as the future, it is particularly well suited for time series issues. The ability to forecast future values using historical trends might be valuable. Additionally, BiLSTMs can be stacked to form deep BiLSTM networks, which can learn more complex representations of the input data. As a result, BiLSTMs are an effective tool for handling vast volumes of sequence data and capturing both short- and long-term relationships in the data.

Althought it has many beneficials, it also has several disadvantages. Its divantages are quite similary to Stacked LSTM but it has some differences. It can be computationally expensive, as it needs to store two hidden states for each time step, which can quickly lead to an explosion in the number of parameters and memory requirements. Additionally, because the hidden states are only sent in one direction, BiLSTM could have trouble understanding long-term relationships in the input data. As a result, modeling intricate, time-dependent relationships in the data may be challenging. By stacking many LSTM layers on top of one another, Stacked LSTM overcomes these restrictions and enables the model to learn richer and more complicated representations of the input data. Better performance may result from doing so, especially when working with complicated input data patterns or long-term dependencies.

In conclussion, BiLSTMs offer a number of advantages over traditional RNNs, making them a popular choice for time series problems and other sequence data tasks.

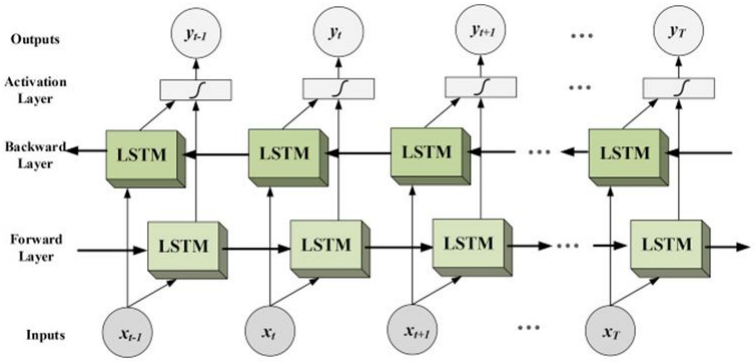


Fig. 20: Bidirectional LSTM (source: medium)

3.3.2.3 Auto-encoder Long-Short Term Memory

Auto-encoder Long-Short Term Memory (Encoder-LSTM) is an implementation of an Auto-Encoder for sequence data using an Encoder-Decoder LSTM architecture.

An encoder-decoder LSTM is trained on a dataset of sequences to read, encode, decode and recreate the input sequence. The success of the model is measured by how accurately

it can recreate the original sequence. Once the desired level of performance is reached, the decoder component can be discarded and only the encoder remains. This encoder model can then be utilized to convert input sequences into a compact fixed-length vector. These vectors can be utilized in a range of applications, including as a condensed representation of the sequence as input to other machine learning models.

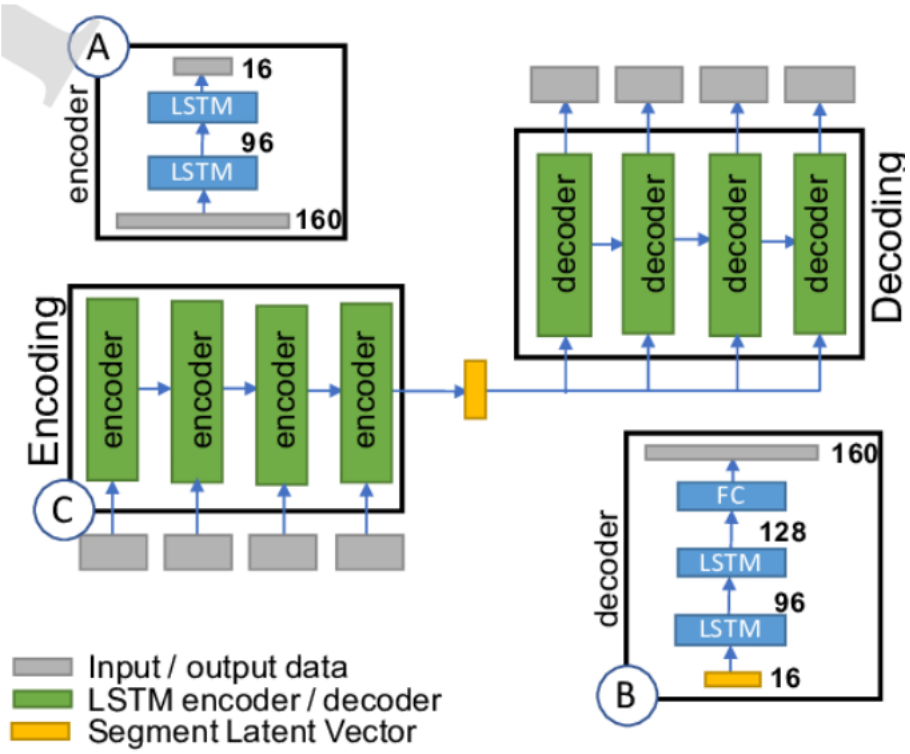


Fig. 21: The architecture of Auto-encoder LSTM.

(A) and (B) are architectures of the encoder and decoder respectively. In (C), each green encoder block represents a time step during encoding phase and gray blocks represent input data for each time step. After encoding, a segment latent vector (the yellow block) is generated.

Auto-Encoder LSTM (Long Short-Term Memory) has several advantages, including:

- Dimensionality reduction: The encoder part of the model can be used to reduce the dimensionality of the input data by encoding it into a lower dimensional space, which can be useful for data compression or visualization.
- Anomaly detection: The autoencoder can be trained to reconstruct normal data patterns and detect anomalies in the input.
- Pre-training for other tasks: The autoencoder can be used as a pre-training step for other machine learning models, allowing for faster convergence and better performance.

- Sequence processing: The LSTM component of the autoencoder allows it to process sequences of data, making it suitable for time series or sequential data processing.
- Generative modeling: The decoder component of the autoencoder can be used for generative modeling, allowing the model to generate new data similar to the input data.

Althought it has many advantages, it also have several disadvantages:

- Complexity: Auto-Encoder LSTMs can be complex to design, train and interpret, requiring a good understanding of deep learning and autoencoder concepts.
- Overfitting: As with any deep learning model, the Auto-Encoder LSTM is susceptible to overfitting, especially if the model is not properly regularized.
- Data requirements: The Auto-Encoder LSTM requires a large amount of labeled data to be trained effectively, which may not always be available.
- Computational costs: The complexity of the model can make training and deploying the Auto-Encoder LSTM computationally expensive, requiring a lot of processing power and memory.

3.3.2.4 Attention mechanism

Attention mechanisms are becoming increasingly popular in deep learning, especially in sequential data processing tasks such as natural language processing. One of the most common architectures used in sequential data processing is Long Short-Term Memory (LSTM). LSTMs are designed to capture the long-term dependencies in sequential data, but they can sometimes struggle with longer sequences. In these cases, the attention mechanism can be added to the LSTM to allow the network to focus on the most important information in the input sequence.

An attention mechanism in LSTMs works by creating a weight vector that determines the importance of each input in the sequence. This weight vector is then used to weigh the inputs, allowing the network to focus on the most relevant information.

The use of attention mechanisms in LSTMs has several benefits. First, it allows the network to handle longer sequences, as the attention mechanism enables the network to focus on the most important information in the sequence. This helps the network to better understand the context and relationships between the inputs. Second, attention mechanisms can also improve the interpretability of the model, as the weight vector created by the attention mechanism can be visualized to understand which inputs the network is focusing on.

There are several types of attention mechanisms that can be used in LSTMs, including

soft attention and hard attention. Soft attention assigns a weight to each input based on its relevance to the output, while hard attention uses a discrete binary value to determine the importance of each input.

In conclusion, the use of attention mechanisms in LSTMs can greatly improve the performance of the model in sequential data processing tasks. By allowing the network to focus on the most important information in the input sequence, the attention mechanism can help the network to better understand the context and relationships between the inputs. Additionally, the use of attention mechanisms can also improve the interpretability of the model. However, it is important to consider the specific requirements of each task before choosing a particular type of attention mechanism to use in an LSTM model.

3.3.3 Gated Recurrent Unit

GRU, also known as Gated Recurrent Unit, is a type of deep learning architecture that is widely used in sequential data processing tasks. It is designed to handle sequential data, such as speech signals, which have a continuous structure and depend on previous inputs. The architecture of GRU consists of two main components, the update gate and the reset gate, which control the information flow through the network. These gates help the network to learn the most important information from the input sequence and store it for future use.

GRUs are often depicted as illustrated below:

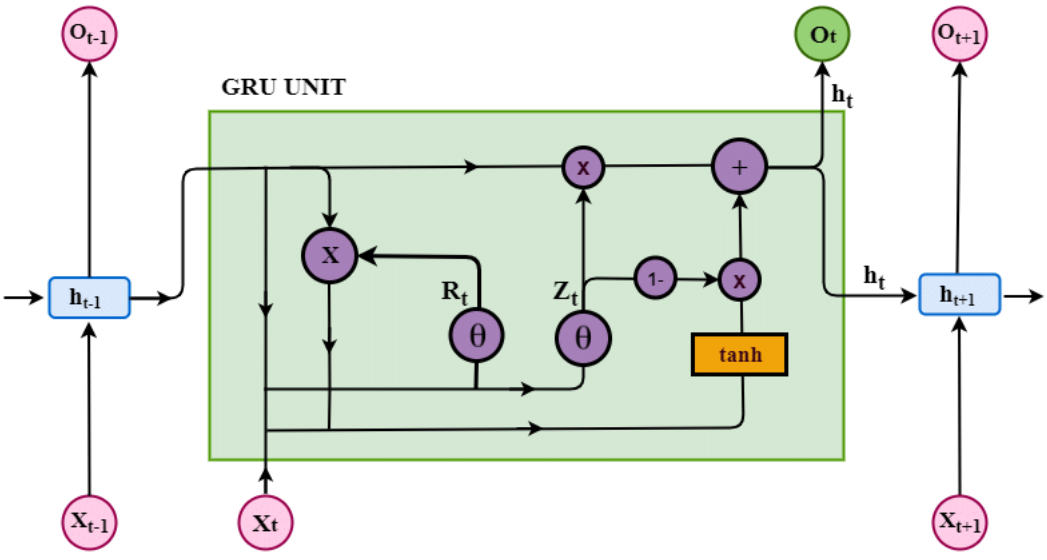


Fig. 22: The architecture of GRU. (source: ResearchGate)

The application of GRUs is widespread in various fields, such as speech recognition, natural language processing, and robotics. In speech recognition, GRUs are used to identify spoken words from speech signals and translate them into written text. In natural language

processing, GRUs are used for text classification, sentiment analysis, and machine translation. In robotics, GRUs are used to control the movement of robots based on their inputs and previous experiences.

The advantages of GRUs include their ability to handle sequential data effectively and their efficiency. Unlike traditional neural networks, GRUs can capture long-term dependencies in sequential data and store the information for future use, which helps the network to better understand the context and relationships between inputs. Additionally, GRUs require less computational power compared to LSTM networks, making them ideal for applications with limited computational resources.

However, like any deep learning architecture, GRUs also have limitations. One of the main drawbacks of GRUs is their limited ability to store information for a long time, which may not be suitable for applications where information needs to be stored for an extended period, such as in robotics or long-term memory tasks. Another limitation is the lack of interpretability of GRUs, as they are considered black box models, making it challenging to understand the decisions made by the network and interpret the results of the model.

In conclusion, GRUs are a powerful deep learning architecture that is well-suited for a range of sequential data processing tasks. Its ability to handle sequential data effectively, combined with its efficiency, makes it a popular choice for speech recognition, natural language processing, and robotics applications. Despite its limitations, GRUs are still a valuable tool for deep learning and provide promising results in a variety of fields.

4 Experimental results

4.1 Experimental setting

To measure the performance of models, we conduct experiences including these scenarios:

- First, measure the performance of two traditional approaches which are not related to deep learning. These are the benchmarks to compare the performance of deep learning models with.
- Second, measure the performance of deep learning models with different number of history time steps used to forecast, then we can know the impact of these important hyperparameters.
- Next, observe the performance of the models using clipping techniques. This result will

be used to answer whether the abnormally low value of the National electricity load should be handled.

- After that, we measure the performance of models when predicting difference number of future time steps. This will show when should we use deep learning models.
- Finally, we test the endurance of each models.

4.1.1 Loss function

In our project, we use three loss function, namely **Mean Absolute Error (MAE)**, **Root-mean-square deviation (RMSE)** and **Mean absolute percentage error (MAPE)**, for measuring forecast accuracy.

4.1.1.1 Mean Absolute Error (MAE)

MAE is average absolute difference between actual and anticipated values. It measures the magnitude of the error, regardless of the direction.

$$\text{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

n is number of data point,

x_i is actual value,

and y_i is predicted value

4.1.1.2 Root-mean-square deviation (RMSE)

Root-mean-square deviation (RMSE) or Root Mean Square Deviation (RMSD) is the average of the squared discrepancies between the actual value and the value anticipated. It is standard deviation of the residuals (prediction error).

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} (y_i - x_i)^2}{n}}$$

n is number of data point,

x_i is actual value,

and y_i is predicted value

4.1.1.3 Mean absolute percentage error (MAPE)

The typical absolute percentage variance between the numbers that were anticipated and those that were actually obtained. It is used to determine how much, in percentage terms, the prediction differs from the actual figure.

$$\text{MAPE} = \frac{1}{n}(\sum_{i=1}^{n} \frac{|y_i - x_i|}{y_i})$$

n is number of data point

y_i is the actual value

x_i is predicted value

4.1.2 Models setting

4.1.2.1 Stacked Long Short Term Memory

In the Stacked LSTM model, we utilized two LSTM layers. The input was first processed through an LSTM layer with 100 outputs, followed by a Dropout layer with a rate of 0.5. Then, the output was further processed through another LSTM layer with 50 outputs, followed by another Dropout layer with a rate of 0.2. Finally, the result was fed into a Neural Network layer to generate the final output with the desired dimension.

4.1.2.2 Bidirectional Long Short Term Memory

The BiLSTM model took the input and processed it through an BiLSTM layer with 150 outputs. The result then was fed into Neural Network layer with 20 outputs, before activating by tanh function. Then, the output ran throught Dropout layer with rate of 0.2, followed by a another Neural Network layer to generate the final output with the desired dimension.

4.1.2.3 Auto-encoder Long Short Term Memory

In Auto-encoder Long Short Term Memory (Auto-encoder LSTM), we utilized three LSTM layers.

The first LSTM layer has 40 units. The output from this layer will be used as input for the next layer in the sequence.

The second LSTM layer has 20 units and also returns sequences, while the third LSTM layer has 15 units and does not return sequences.

A RepeatVector layer is then added. This layer is used to ensure that the output from the encoder can be processed by the decoder.

The last three layers are LSTM layers with 40, 25 units respectively, each of them also returning sequences.

The final layer is a TimeDistributed layer that applies a Dense (fully connected) layer to each time step of the input. The units parameter of the Dense layer is set to the desired dimension.

4.1.2.4 Attention mechanism

The model starts with two Bidirectional LSTM layers, with 100 and 50 units respectively, that analyze the input data and capture its temporal relationships. The LSTM layers are set to return sequences, meaning that their output will be passed on to the next layer for further processing.

After the LSTM layers, the model uses an attention mechanism to weigh the importance of each part of the input data for making predictions. This is done by passing the output of the LSTM layers through a Dense layer with a softmax activation function, which normalizes the values to sum up to 1. The attention scores are then reordered using a Permute layer, before being multiplied with the original input data using a Multiply layer.

The attention-weighted input data is then processed through a series of dense layers, with a tanh activation function, a Dropout layer, and another Dense layer with horizon units. The Dropout layer is used to prevent overfitting, by randomly setting some of the activations to zero during training. The final output of the model is a prediction for a given horizon.

4.1.2.5 Gated Recurrent Unit

The model has two GRU layers, two Dropout layers, and one Dense layer.

The first GRU layer has 100 units and takes in the input shape of the x_train_multi data, which has the shape of (batch size, time steps, features). The return_sequences parameter is set to True, which means that the GRU layer will return a sequence of outputs for each time step. The Dropout layer is used to regularize the model by randomly dropping out a certain percentage of neurons, in this case, 20

The second GRU layer has 50 units and the return_sequences parameter is set to False, which means that it will only return the last output of the sequence. The second Dropout layer is also set to drop out 20

Finally, the Dense layer has a number of units equal to the horizon, which represents the number of future time steps that we want to predict. The model is compiled using the Adam optimizer and the mean squared error (mse) loss function. The mse loss function is a common choice for regression problems, as it measures the difference between the predicted and actual values.

4.2 Vector Autoregression

Bayesian Information Criterion (BIC) is a model selection tool that provides a measure of the relative goodness of fit of different models to a given data set. It is widely used in time series analysis to determine the best model for a specific time series. BIC considers both the accuracy of the model and its complexity, and balances these two factors to provide an overall score for each model. The criterion is based on Bayesian statistics and uses a penalization term to discourage overly complex models that overfit the data. In time series analysis, models with lower BIC values are considered to be a better fit for the data than models with higher BIC values. The BIC is particularly useful in time series analysis as it helps to identify the number of factors that should be included in a model, and also helps to determine if a simple or complex model is appropriate for the data. Overall, the BIC is a valuable tool in time series analysis, as it provides a way to compare different models and select the best one for a specific time series. As in 23, VAR model with 26 time steps is the best one.

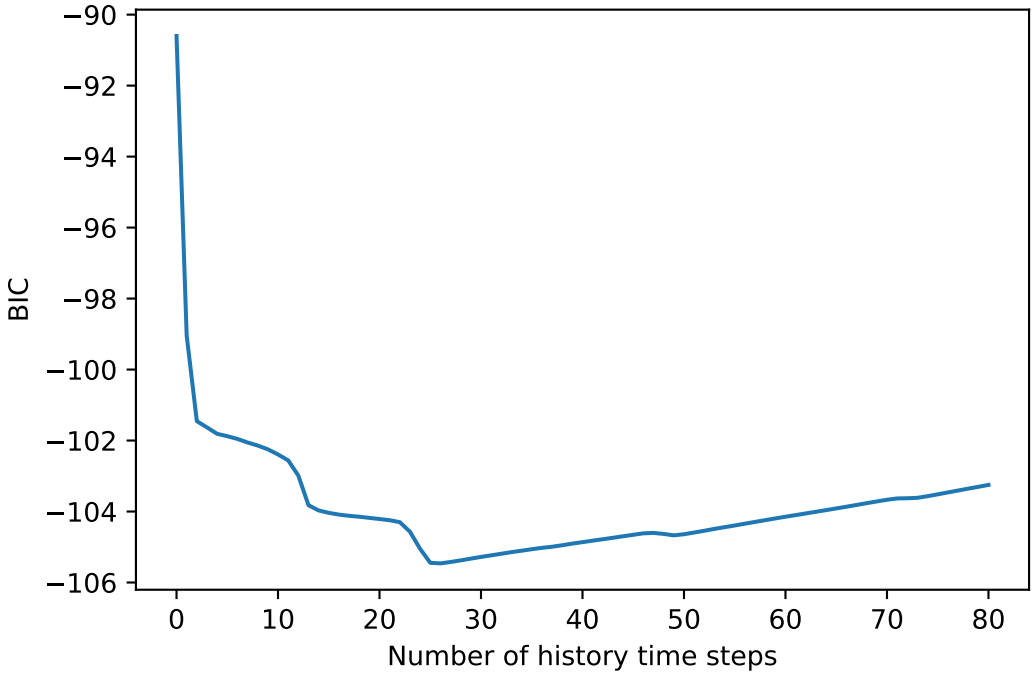


Fig. 23: BIC results of VAR.

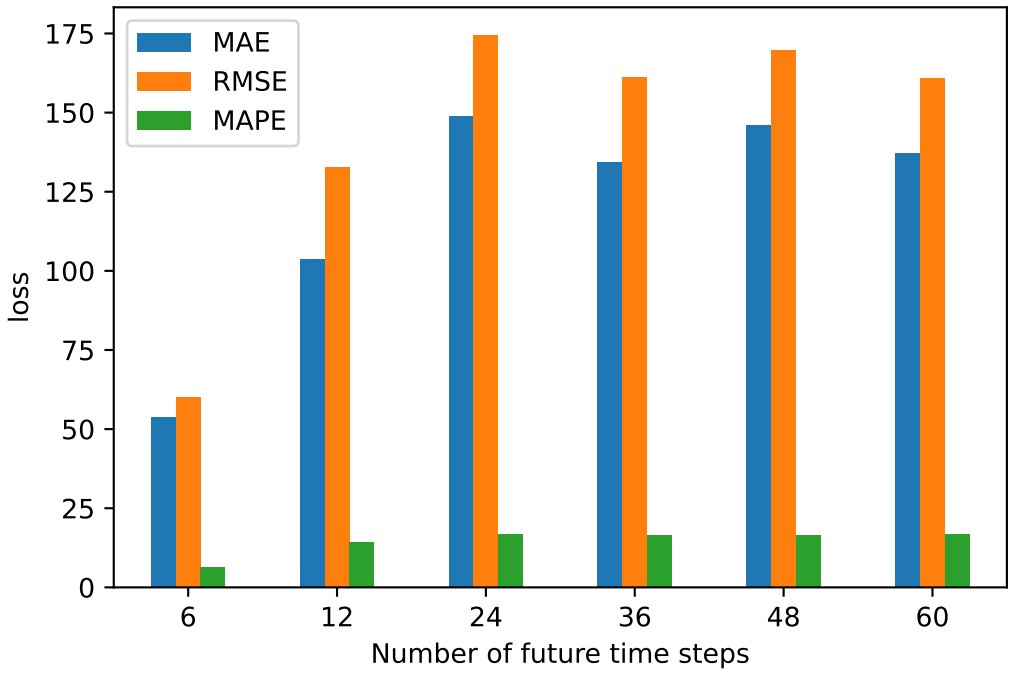


Fig. 24: Performance of VAR in predicting different number of future days.

The given bar chart compares the performance of VAR with three loss function, namely MAE, RMSE and MAPE, in predicting different number of future time steps. Looking at the chart in detail, the measures value is quite low in the near future and very high in the far future. It could be explained that VAR models are capable of capturing dynamic relationships between variables, which is an important advantage for near-term forecasting. While VAR models are good for near-term forecasting, they may not be the best choice for forecasting further into the future.

4.3 Facebook Prophet

The experiment results in 25 shows that Facebook Prophet model is worse than VAR model with respect to the loss value in all cases. When seeing in 24, the results shows that VAR is stable in the near future, but unstable in the far future. But, when seeing in 25, Prophet is stable in the near and far future. One key aspect of the Prophet model that helps it maintain stability in both the near and far future is its use of multiple seasonality components with linear and non-linear growth. This allows the model to capture different types of seasonal patterns in the data, such as daily, weekly, and yearly patterns, and account for changes in these patterns over time.

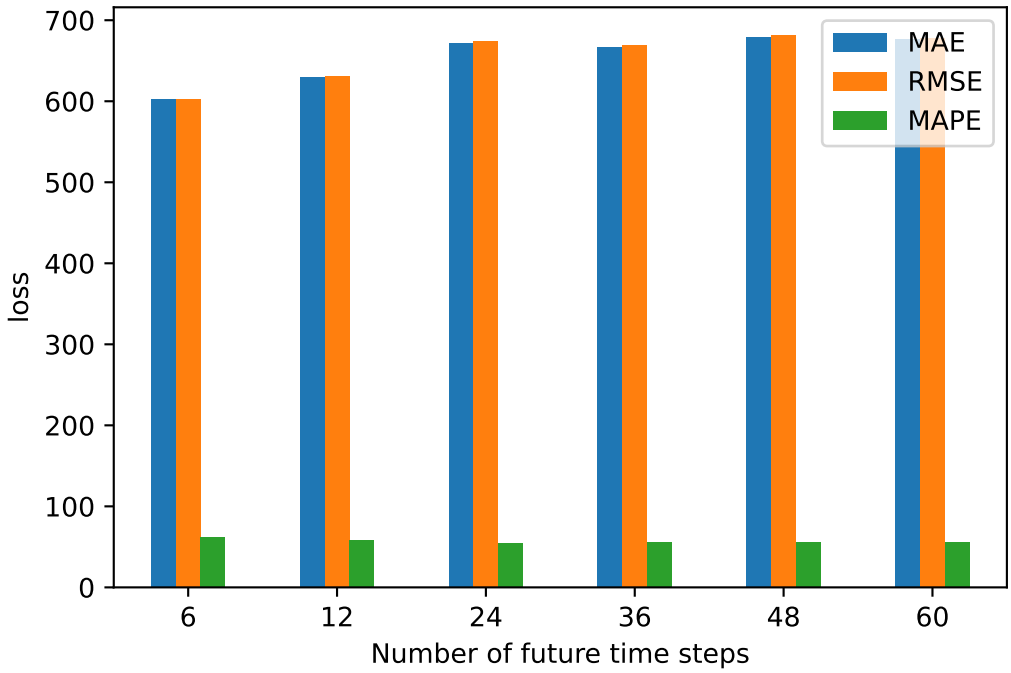


Fig. 25: MAPE of Deep Learning with different number of history time steps.

4.4 Deep Learning

4.4.1 Impact of history time steps on the per deep learning models

The given graph 26 compares **Mean Absolute Error (MAE)** value of five best model of five algorithms respectively that estimate the Panama electric load based on the number of history time steps spaced 6 hours starting from 12 to 54 before the predicted times.

Overall, all of models have worst result at the last time steps because of some considerable fluctuations. In addition, BiLSTM model and Encoder_Decoder LSTM model often have MAE value higher than other models. Looking the graph in detail, we can see that GRU model, BiLSTM model and Encoder_Decoder model often have MAE value exceed 100. However, the **GRU model** illustrates the **best result** and overcomes in comaparision with all of time steps of the other models with MAE values of virtually 42 at 42 time steps.

Regarding to the other models, both of two models demonstrates the good results and the best result at 30 time steps with MAE values of just below 80. It is note of that the best results of Stacked LSTM model and Attention LSTM model are better than the other model results except for GRU model at 42 time steps. Table 1 shows the reason why we use Stacked LSTM instead of 1-layer LSTM. By stacking two layers, the model can learn more complicated representations of the input sequence.

The given graph 27 render a break down of **Root-mean-square deviation (RMSE)**

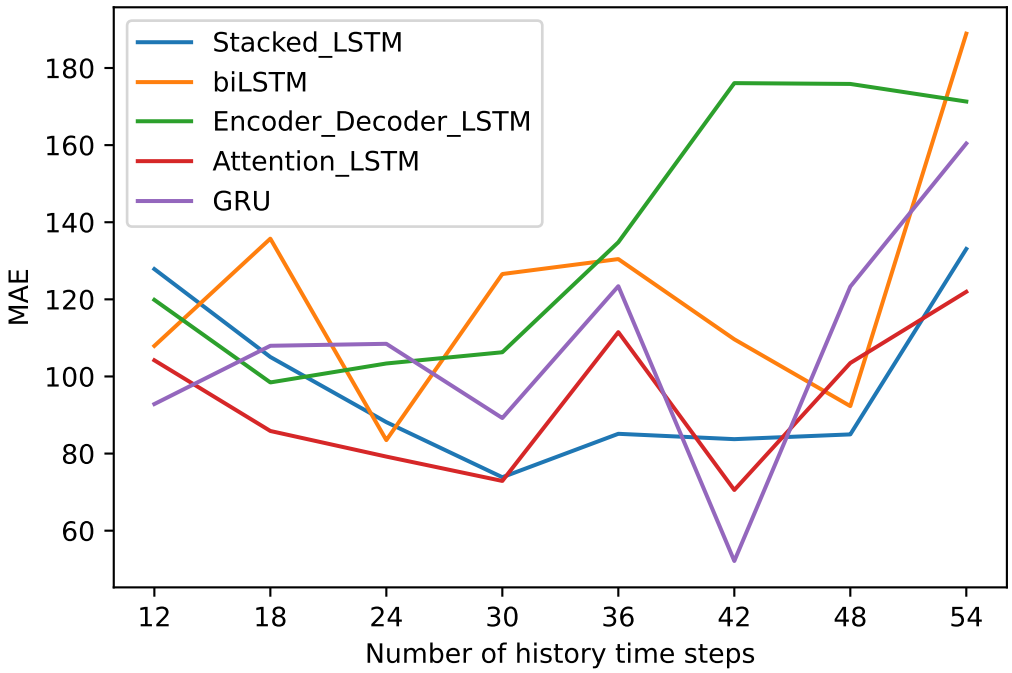


Fig. 26: MAE of Deep Learning with different number of history time steps.

value of five best model of five algorithms respectively that estimate the Panama electric load based on the number of history time steps spaced 6 hours starting from 12 to 54 before the predicted times.

Overall, the result of five models quite similar with that of above graph 26 despite of some differences. Likewise, BiLSTM model and Encoder_Decoder LSTM model continue to have high RMSE value in comparision with other models.

The GRU model remain the best results in comparision with all of time steps of the other models with RMSE value of virtually 60 at 42 time steps. Besides, the Attention LSTM model has the best result at the same time steps (30 time steps) but the former has result better than the later with RMSE value of nearly 85 and 90 respectively.

The given graph 28 illustrates **Mean absolute percentage error (MAPE)** value of five best model of five algorithms respectively that estimate the Panama electric load based on the number of history time steps spaced 6 hours starting from 12 to 54 before the predicted times. Overall, there are many significant changes in comparision with two above graph 26 27. Specially, Attention LSTM model become has the worst result compared to other model. While Encoder_Decoder LSTM model and BiLSTM still have the bad result, the GRU model continue to has the smallest MAPE value compared to all of time steps of the other models with MAPE value of nearly 4.6 at 42 time steps, followed by Stacked LSTM model with RMSE value of virtually 6.4 at 30 time steps. It is note of that Attention LSTM

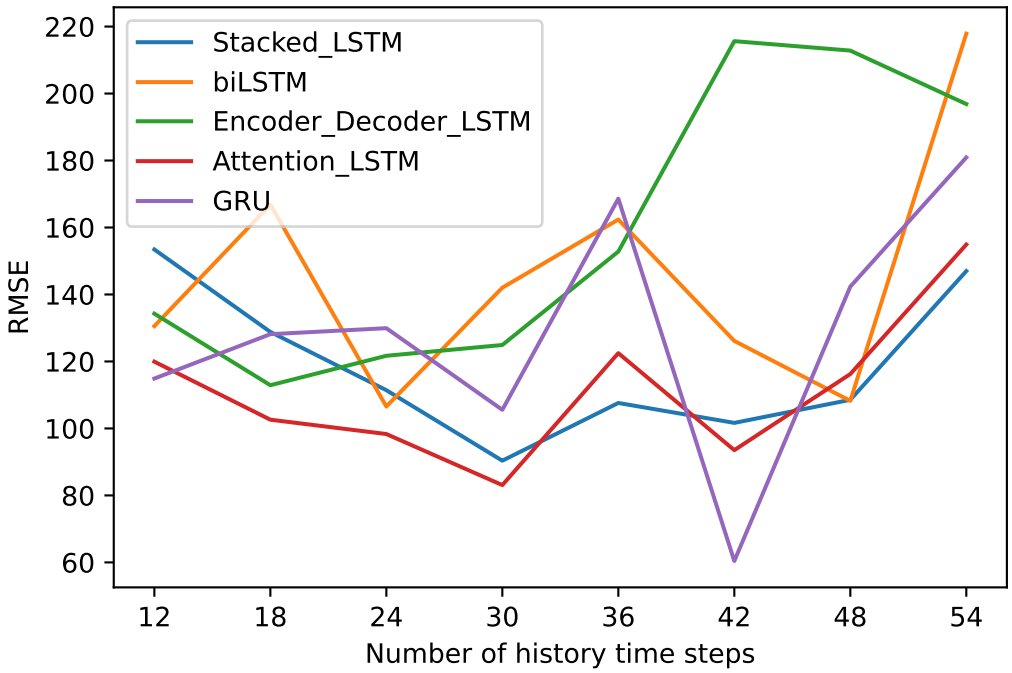


Fig. 27: RMSE of Deep Learning with different number of history time steps.

model has the worst result, followed by Encoder_Decoder LSTM model at all time steps.

In wrap up, the GRU model had an exceptional result at the 42nd time step . On the other hand, the Attention LSTM model showed a poor result in the third graph 28, but it was quite stable when predicting the future using MAE or RMSE, regardless of the time step. By contranst, the BiLSTM model's performance varied significantly throughout the course of the forecast period, making it less reliable. The GRU and Encoder/Decoder LSTM models, with the exception of the GRU model's performance at the 42nd time step, were better suited for forecasting the future with a smaller input. In addition, the Stacked LSTM model performed well while utilizing a big input. The best time steps for these model is from 24 to 30 and between 42 and 48. If the inputs is smaller than 24 or bigger than 48, it can be meet underfitting and overfitting respectively. The time steps from 30 to 42 is not good for prediction may be beacause of data which has some anomalies.

	1 layer MAE	2 layers MAE	1 layer MAPE	2 layers MAPE
LSTM	98.176	73.862	8.065	6.397
GRU	118.966	52.136	9.502	4.635

Table 1: Performance of LSTM and GRU when having 1 or 2 LSTM/GRU layers

The experiment results in 29 and 30 regarding the use of clipping techniques in GRU provide insights into the effects of clipping on the training process. The results showed that using clipping led to a more stable relationship between the validation loss and the training

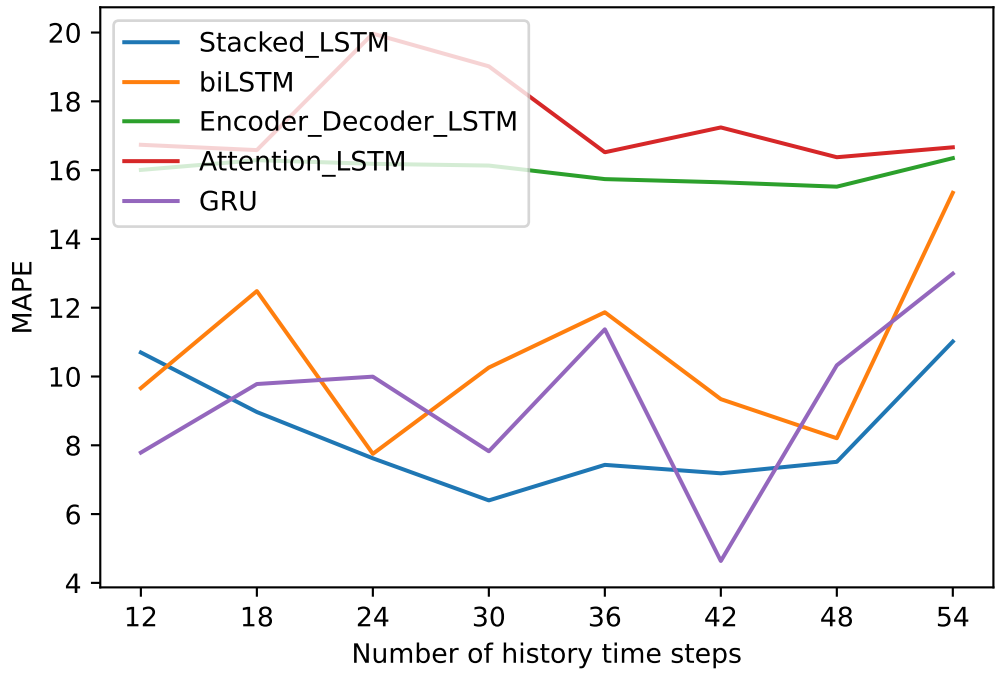
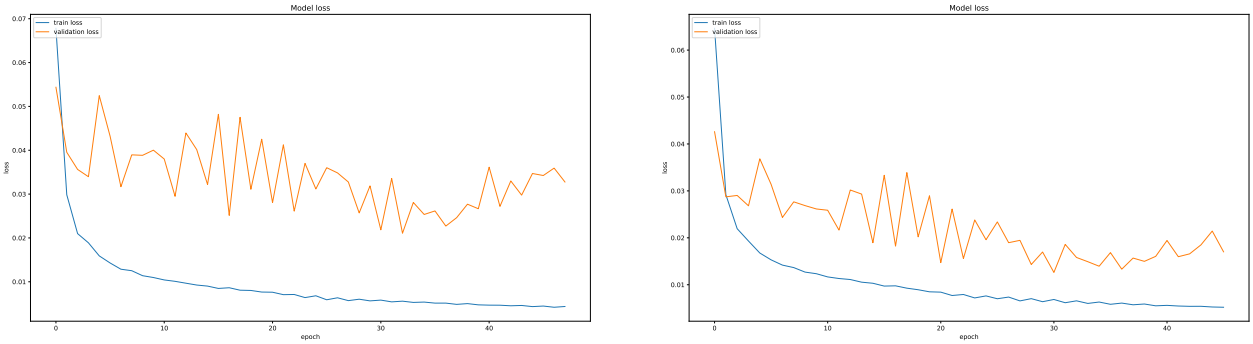


Fig. 28: MAPE of Deep Learning with different number of history time steps.

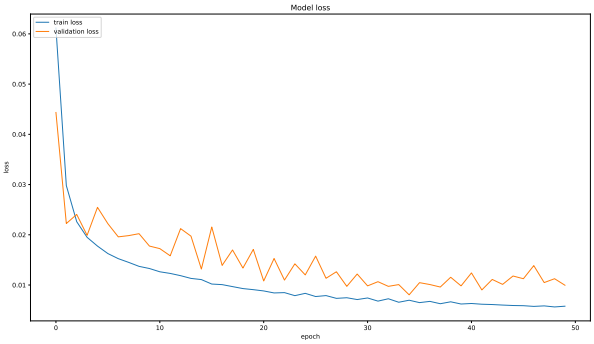
loss. The validation loss was found to be closer to the training loss when clipping was applied, suggesting that the model was able to generalize better. However, the overall performance of the model with clipping was found to be worse compared to the model without clipping. This raises questions about the effect of clipping on the learning process and the information that is being outlier during the clipping process.

4.4.2 Impact future time steps on the performance of deep learning models

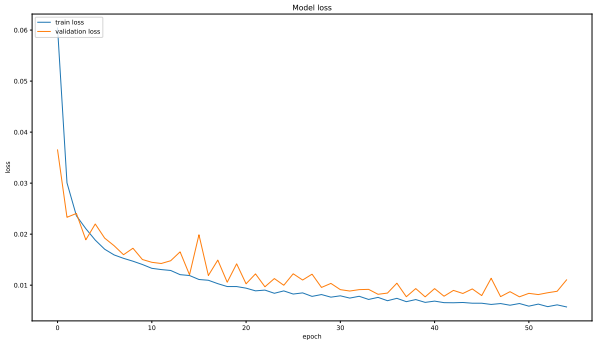


(a) No clipping

(b) Clipping 550



(c) Clipping 650



(d) Clipping 750

Fig. 29: Loss vs validation loss of clipping techniques using GRU

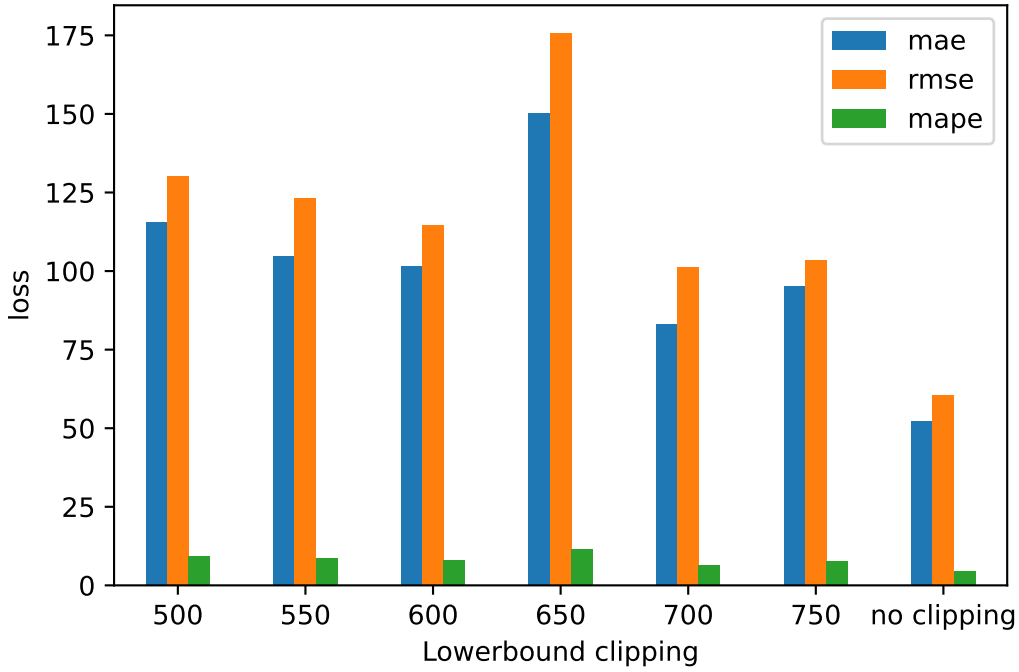


Fig. 30: Performance of GRU with clipping technique.

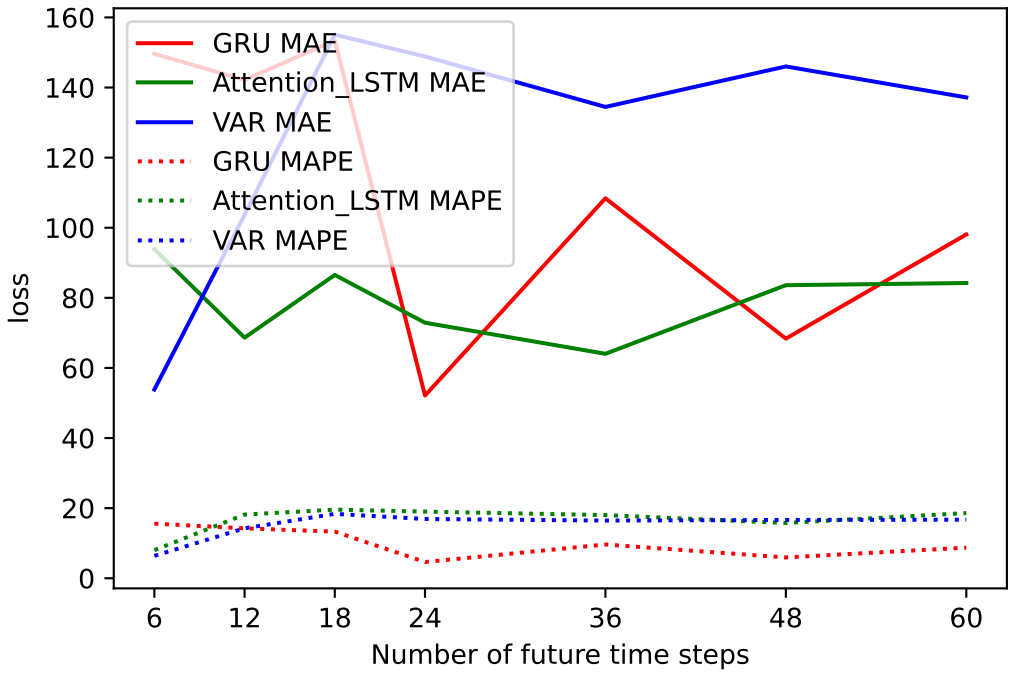


Fig. 31: MAE and MAPE of Deep Learning with different number of future time steps.

The given graph 31 illustrates MAE and MAPE value of three model, namely GRU, Attention LSTM and VAR, with diffenret number of future steps spaced 6 hour from 6 to 60 starting from predicted time.

Looking at graph in detail, we can see that the VAR model performs better in the near term predictions, while the GRU model performs better in the long term predictions. Additionally, the Attention model is said to have relatively stable performance across different time steps.

The VAR model is particularly effective at forecasting the near future since it concentrates on modeling the link between a group of variables and their historical values. On the other hand, GRUs, as a type of recurrent neural network, have the ability to remember past values and use them to make predictions, making it better suited for predicting the far future.

The stability of Attention LSTM can be attributed to its ability to weigh different time steps differently and dynamically adjust the importance of each time step based on the input. The model may successfully decrease the influence of noise in the time series data by emphasizing the most significant time steps and giving varying weights to various time steps. Additionally, the model may selectively concentrate on particular areas of the time series data thanks to the attention mechanism, making it more resistant to changes and outliers in the data. We can see the clear better performances of Deep Learning models in Fig. 32.

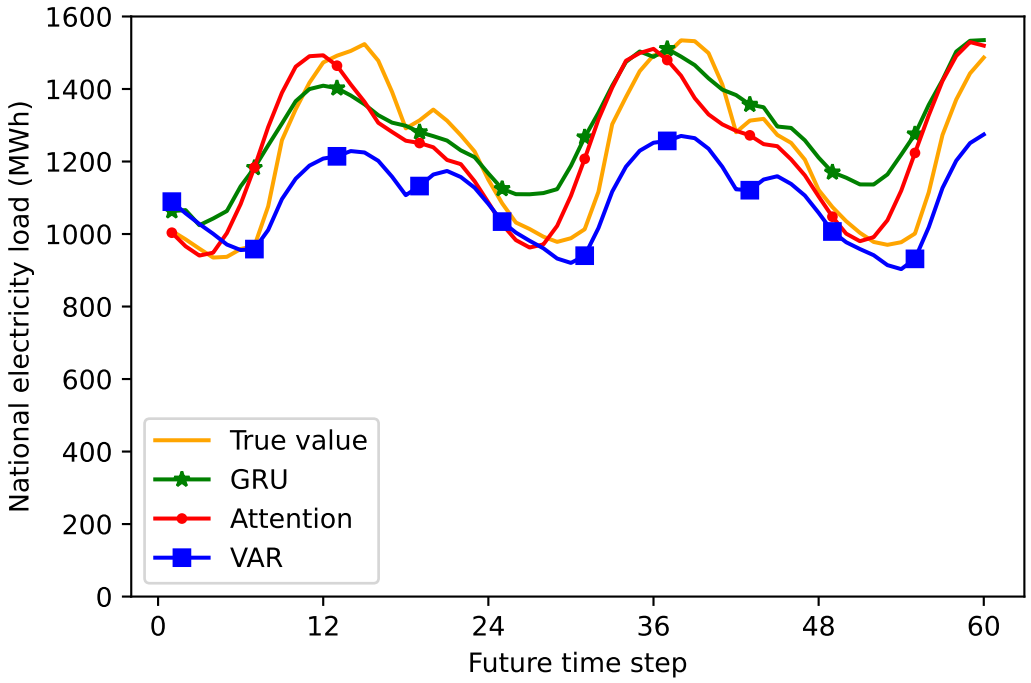


Fig. 32: Forecast vs Actual values.

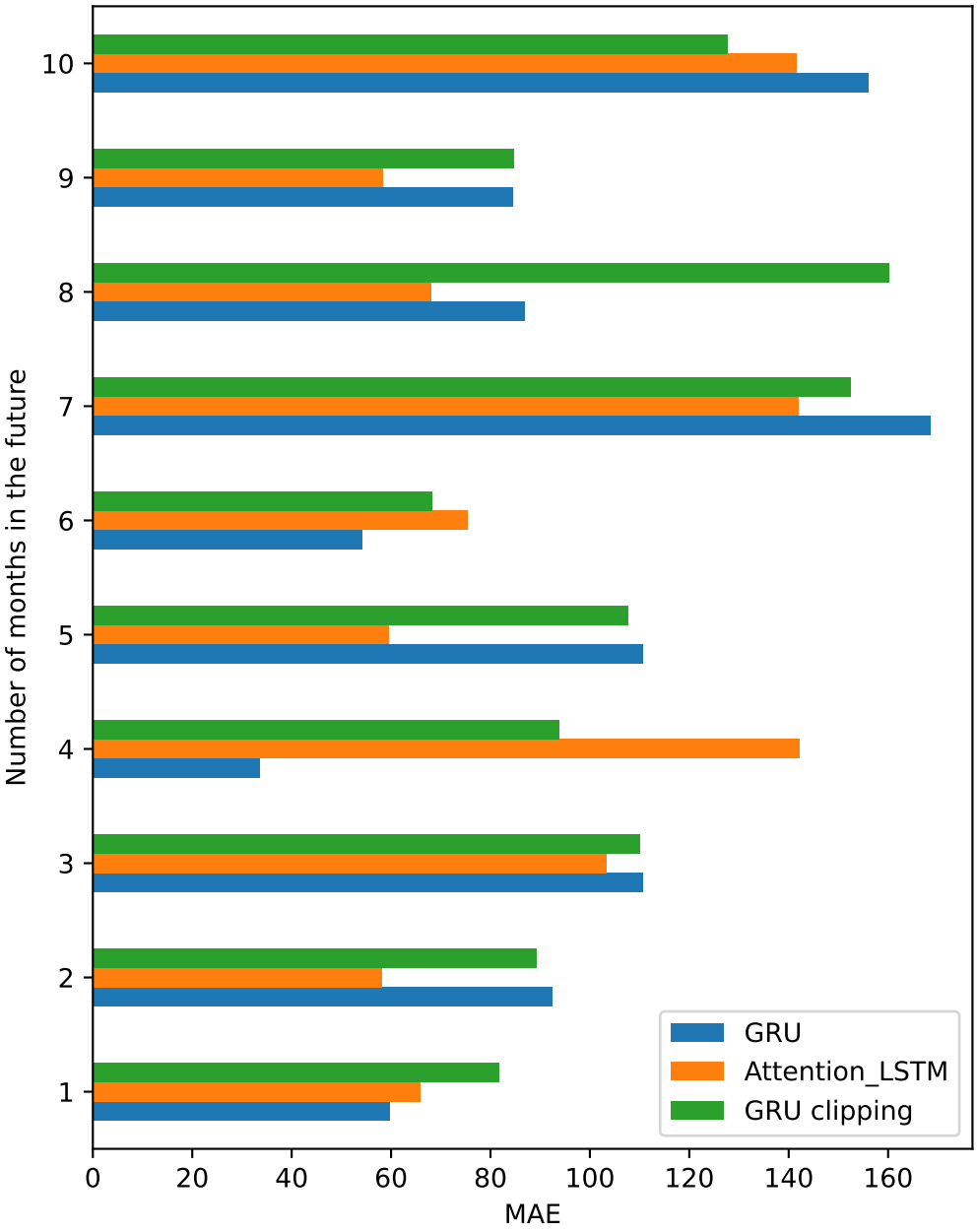


Fig. 33: Endurance of Deep Learning models.

When using the GRU model trained to forecast 24 future time steps to make prediction of the next 6, 12, and 18 time steps, we see a remarkable result. The MAE value is 83.401, 65.231, and 59.07 respectively, which is better than the models only trained to forecast fewer future time steps. With respect to MAPE, we also see that the results are also better with 8.654, 6.452, and 5.367 respectively. These phenomenon asserts that the model trained to forecast fewer future time steps may bring worse results since they can not capture the changing rules of the national electricty based on other 14 attributes effectively.

The endurance of models is presented by how they performs after a certain time not re-training. It is noteworthy in Fig. 33 that 6 out of 10 cases Attention LSTM has the best performance (which is calculated as MAE value of the next 24-forecast-time-step result). This is double as many of GRU. Moreover, the worst or best performance both belongs to GRU. There is only one time when the clipping becomes the best. On the other side, half of the scenerios shows the worst models to be GRU. Only two scenarios when Attention mechanism brings the worst results. In general, the endurance of Attention LSTM is the best and proved that the mechanism of focusing on the important information really can help the network to better understand the context and relationships between the inputs.

5 Conclusion

In this report, we has shown the effective of Deep Learning approaches in multivariate time series forecasting problem.

Link Github: https://github.com/Tahuubinh/DL_HUST_Project

We would like to extend our sincere gratitude to Assoc. Prof. Nguyễn Thị Kim Anh and Prof. Trần Việt Trung, our advisor and mentor, for their unwavering support and guidance throughout our deep learning project. Their expert knowledge and passion for the field of deep learning inspired us to pursue this project and challenged us to expand our understanding of the field. Their patient and encouraging approach provided us with the confidence to take on this project and their valuable feedback helped us to improve our work. We are truly grateful for their dedication to our success and for the time and effort they invested in our growth as deep learning researchers. Thank you, Assoc. Prof. Nguyễn Thị Kim Anh and Prof. Trần Việt Trung, for being an exceptional teacher and mentor.

References

[1] Jason Brownlee. *Deep learning for time series forecasting: predict the future with MLPs, CNNs and LSTMs in Python.* Machine Learning Mastery, 2018.

[2] Ernesto Javier Aguilar Madrid. *Short-Term Electricity Demand Forecasting with Machine Learning.* PhD thesis, Universidade NOVA de Lisboa (Portugal), 2021.