



MONTCLAIR STATE
UNIVERSITY

Montclair State University
**Montclair State University Digital
Commons**

Theses, Dissertations and Culminating Projects

1-2021

Deep Learning for Electricity Forecasting Using Time Series Data

Hanan Abdullah Alshehri
Montclair State University

Follow this and additional works at: <https://digitalcommons.montclair.edu/etd>



Part of the [Computer Sciences Commons](#)

Recommended Citation

Alshehri, Hanan Abdullah, "Deep Learning for Electricity Forecasting Using Time Series Data" (2021).
Theses, Dissertations and Culminating Projects. 678.
<https://digitalcommons.montclair.edu/etd/678>

This Thesis is brought to you for free and open access by Montclair State University Digital Commons. It has been accepted for inclusion in Theses, Dissertations and Culminating Projects by an authorized administrator of Montclair State University Digital Commons. For more information, please contact digitalcommons@montclair.edu.

ABSTRACT

The complexity and nonlinearities of the modern power grid render traditional physical modeling and mathematical computation unrealistic. AI and predictive machine learning techniques allow for accurate and efficient system modeling and analysis. Electricity consumption forecasting is highly valuable in energy management and sustainability research. Furthermore, accurate energy forecasting can be used to optimize energy allocation. This thesis introduces Deep Learning models including the Convolutional Neural Network (CNN), the Recurrent neural network (RNN), and Long Short-Term memory (LSTM). The Hourly Usage of Energy (HUE) dataset for buildings in British Columbia is used as an example for our investigation, as the dataset contains data from residential customers of BC Hydro, a provincial power utility company. Due to the temporal dependency in time-series observation data, data preprocessing is required before a model can be created. The LSTM model is utilized to create a predictive model for electricity consumption as output. Approximately 63% of the data is used for training, and the remaining 37% is used for testing. Various LSTM parameters are tested and tuned for best performance. Our LSTM predictive model can facilitate power companies' resource management decisions.

MONTCLAIR STATE UNIVERSITY

Deep Learning for Electricity Forecasting Using Time Series Data

by

Hanan Abdullah Alshehri

A Master's Thesis Submitted to the Faculty of
Montclair State University

In Partial Fulfillment of the Requirements

For the Degree of

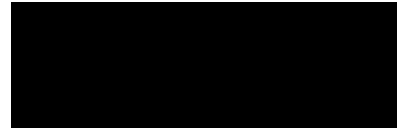
Master of Science

January 2021

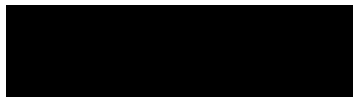
College/School: Science and Mathematics

Department : Computer Science

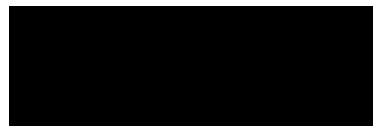
Thesis Committee:



Dr. Michelle Zhu
Thesis Sponsor



Dr. Weitian Wang
Committee Member



Dr. Jiaying Wang
Committee Member

Deep Learning for Electricity Forecasting Using Time Series Data

A THESIS

Submitted in partial fulfillment of the requirements

For the degree of Master of Science

By

Hanan Abdullah Alshehri

Montclair State University

Montclair, NJ

2021

Copyright c 2021 by Hanan Abdullah Alshehri . All rights reserved

ACKNOWLEDGMENTS

Foremost, I would like to express my deepest appreciation and thanks to my advisor Dr. Michelle Zhu. You have been a great mentor, and I am very grateful for your having motivated me and helped me clarify my thoughts. I would also like to thank you for encouraging my research and helping me develop research skills. Your advice on both research methods and how to complete my master's program have been invaluable.

I owe thanks from the bottom of my heart to a very special person, my husband, Abdulaziz. His support and understanding during my pursuit of the master's degree made completion of this thesis possible. I appreciate my little boy Hussam, who has been the light of my life for the last two years; he has given me the extra strength and motivation to get things done.

Most importantly, I would like to thank my family members for all their support and love over the years. I am fortunate to have a beautiful mother who supported me in all my pursuits. Last but not least, I would like to give special thanks to my father who always believed in my ability to be successful, not only in the academic arena, but in life.

Table of Contents

Abstract.....	i
Acknowledgments.....	v
List of Tables.....	vii
List of Figures.....	viii
Chapter 1: Introduction.....	1
Deep learning Methods	1
1.1 Convolutional Neural Network(CNN)	2
1.2 Recurrent Neural Network(RNN)	4
1.3 Long Short- Term Memory(LSTM).....	6
Chapter 2: Related works	8
Chapter 3: Data Source	10
3.1 Dataset Description.....	11
3.2 Data Preparation.....	13
Chapter 4: Algorithm Design	14
Chapter 5: Experimental Results.....	16
Chapter 6: Conclusion and Future Work.....	32
References	33

LIST OF TABLES

[1] Residential File Data (data, hours, energy_ kwt).....	11
[2] Import libraries and read weather dataset file using pandas.....	11
[3] Four years of hourly energy data	12
[4] Time series weather dataset consists of data (hours, temperature, humidity, pressure, weather, energy_ kw/h, and var1(t)).....	16
[5] Comparison of time lag (leg=1, leg 2& leg=5)with metric	17
[6] Pearson correlationbetween variables.....	18
[7] Kendall correlation between dataset variables.....	20
[8] The Comparison of Batch size vs Metric-1.....	23
[9] The Comparison of Batch size vs Metric-2.....	24
[10] Comparison between two Optimizers Adam and SGD.....	26
[11] Categoricalenergy consumption level.....	28
[12] Transformed dataset.....	29
[13] Pearson correlation	29
[14] Kendall correlation	29
[15] Month column in the dataset.....	30
[16] : Month with energy consumption	30

LIST OF FIGURES

[1] The architecture of Deep Learning neural networks and the types of layers.....	2
[2] Convolutional Neural Network Layers (convolution, fully, Nx binary classification)	2
[3] Simple Convolution of a (5x5) matrix with a (3x3) kernel... ..	3
[4] Max Pooling layer applied a single slice of an input volume.....	4
[5] Recurrent Neural Network (RRN) take some input X and feeds that input into the RNN...	5
[6] LSTM cell diagram with various components (input, forget get, and output gate)	6
[7] Heat map for Pearson correlation between variables	19
[8] Batch size 50.....	21
[9] Batch size 72.....	21
[10] Batch size 100.....	22
[11] SGD Optimizer	27
[14] Adam Optimizer	27
[15] Energy consumption versus months	31

CHAPTER 1

INTRODUCTION

Electricity has increasingly become a necessity in daily living, and over the past few decade's electricity usage has skyrocketed. In 2018, the overall global energy consumption grew by 2.3%, a rise driven by economic growth and increased regional cooling and heating demands. However, with the advent of smart grids, precise electric load forecasting has become increasingly critical as it enables power companies to schedule the optimized loads and minimize the use of wasteful electrical products [1]. Since electricity consumption varies according to the time of day, models are constructed using time series data to forecast electricity consumption for any given time [2]. The Deep Learning model utilizes neural networks and has proven to be very successful in modeling complex systems [3]. Deep Learning models can achieve high levels of accuracy and efficiency compared with mathematical modeling approaches[4].

1. Deep Learning Methods

Deep Learning is a branch within the area of machine learning, and it involves learning of new knowledge without the computer being specifically programmed. Deep Learning is used in fields such as telecommunications, banking, biomedicine, spam detection, and image classification. As illustrated in Fig. 1, Deep Learning physically involves an input layer with three neurons that receives input information which is then passed on to a hidden layer. The hidden layer performs mathematical calculations based on the input--the greater the number of hidden layers between the input and the output, the greater the depth. Each hidden layer represents an individual machine learning algorithm. The output is finally produced by the last hidden layer.

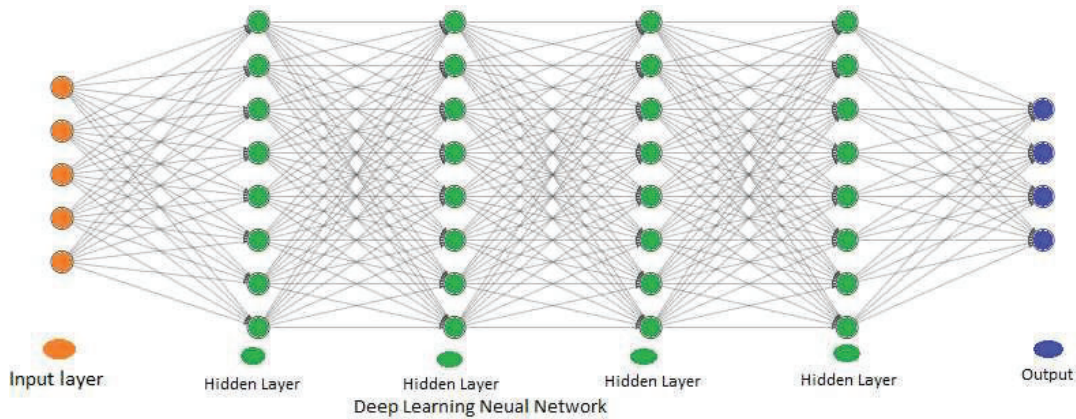


Figure1: The architecture of Deep Learning neural networks and the types of layers
 (from <http://alexlenail.me/NN-SVG/LeNet.html>)

1.1 Convolutional Neural Network (CNN):

The human brain’s biological processes inspired the development of CNN. CNNs can be used to identify important features in an image, classify an image, process natural language, and investigate financial time series [4]. CNN, a well-known architecture of Deep Learning in Figure 2, influenced by living creatures' natural visual perception processes, is a regularized version of multilayer perceptron. Multilayer perceptron refers to fully connected networks in which every neuron in one layer is connected to all neurons in the adjoining layer.[5]

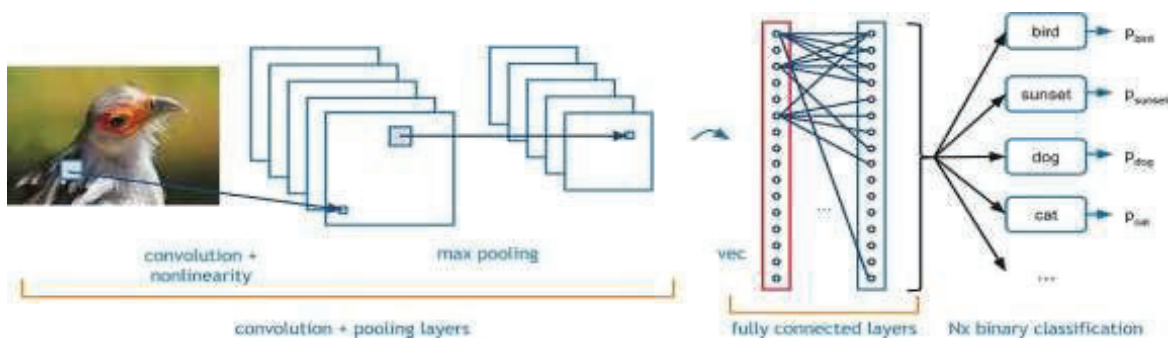


Figure 2: Convolutional Neural Network Layers (convolution, fully, Nx binary classification) [5].

A. Convolution:

The first layer is the most critical component layer that extracts the most important features of an image and applies filters such as an Activation Function (ReLU & Sigmoid). The convolution's purpose is to identify an image's features and decrease a picture's size for faster computations [11]. Figure 3 below shows how the convolution operates. The computer scans a picture segment with a distance of 5×5 picture input and redoubles it with a 3×3 filter to make a 3×3 feature map as the output. After the convolution, the picture is smaller [12]. Mathematically, a convolution of two functions f and g is defined as in [11]

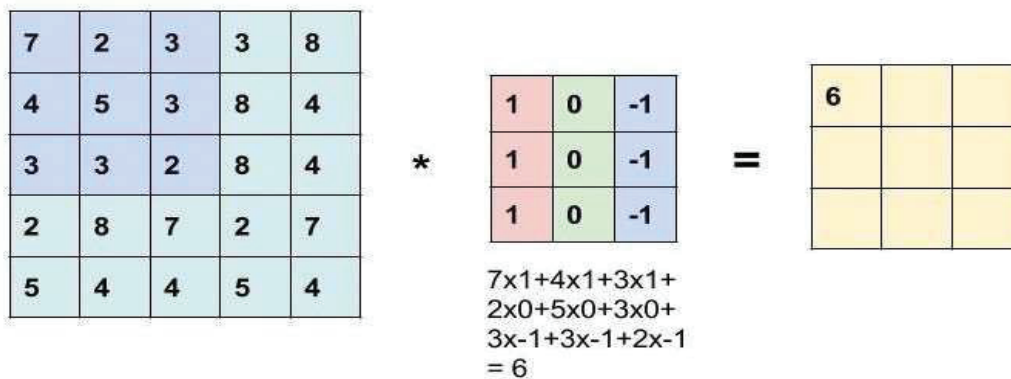


Figure 3: Simple Convolution of a (5x5) matrix with a (3x3) kernel[11].

B. Pooling:

Other than convolutional layers, a CNN usually uses pooling layers to minimize the size of the representation and to speed computation. Retaining important features while downsizing the image is called *Spatial Pooling* or *subsampling*, and it can be accomplished through various methods such as Max, Average, and Sum, etc. [12]. The common approach used in pooling is max pooling. Suppose someone has a 4×4 input and wants to apply a type of pooling called Max pooling as seen in Figure 4. This particular implementation of max pooling renders a 2×2 output. If the 4×4 input is broken into various regions, the output

becomes the size of 2x2. Each output will be the max from the corresponding shaded region. Hence, in the upper left, the max of these four numbers is 9. On the upper right, the max of the numbers is 2. In the lower left, the biggest number is 6. On the lower right, the biggest number is 3. Thus, to compute each frequently occurring number on the right, the max is taken over a 2x2 region. A filter size of two (f=2) is applied, when a 2x2 region and a stride of two (S=2) is taken. These are the hyperparameters of max pooling. It can be seen that pooling picks the maximum value of a 2x2 array, and subsequently moves these windows by two pixels [12].

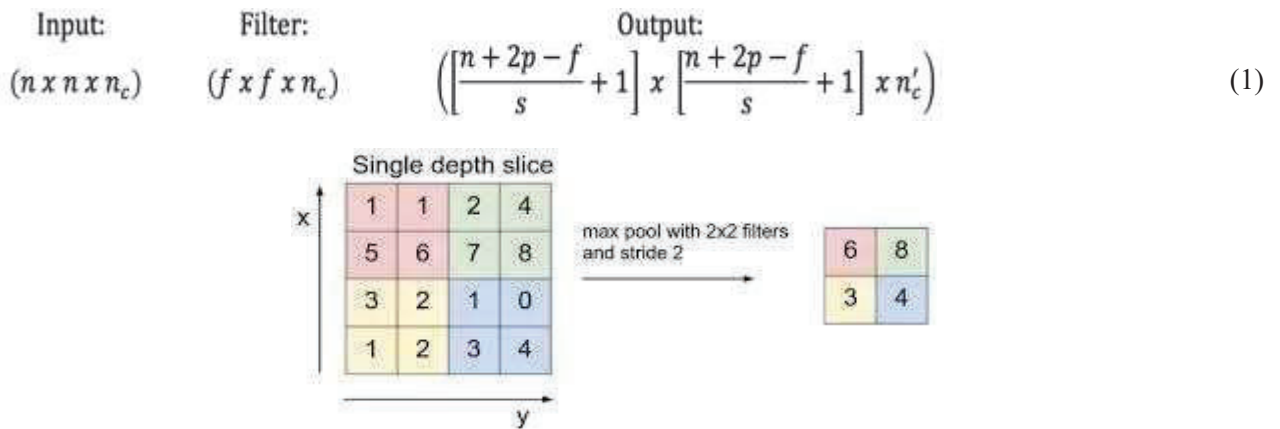


Figure 4 : Max Pooling layer applied a single slice of an input volume[12].

1.2 Recurrent Neural Network (RNN):

Recurrent Neural Networks (RNN) are designed to deal with time-varying inputs in which relevant data is used to determine if time t might have happened at other times in the past. Standard RNNs such as a single layer of Tanh have a simple repeating module structure. A Recurrent Neural Network is a form of Deep Learning Network that includes loops/repetition within the networks to enable memory. Previously stored memory information is used to predict the next value. These networks are useful in recognizing the sequence dependency of data. In

addition to predicting future data, it also maintains some advantages of time serial data. RNN can retain data for a long time, but it is incredibly hard to publish all data when the time period is excessively long. It becomes untrainable when a network contains too many deep layers—a condition called *problem of the vanishing gradient*. To address this gradient disappearance issue experienced by RNN, researchers have constructed an architecture called *long- term short-term memory* (LSTM). This network provides historical information relevant to more recent time steps and uses a better framework [6] to find and pass on information.

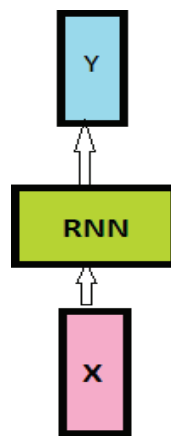


Figure 5 : Recurrent Neural Networks (RNN) take some input X and feeds that input into the RNN

As Figure 5 shows, a Recurrent Neural Network takes some portion of input X and passes it through an internal hidden state. the portion will be updated every time the RNN reads a new input. The internal hidden state will be returned to the model the next time an input is read. After some time steps, it is expected that the RNN will predict an output vector.

$$h_t = fw(h_{t-1}, x_t) \tag{2}$$

Inside the green RNN block in Figure 5, some recurrence relation with function f is computed. Thus, function *f* will depend on some weight *W*, and will accept previously hidden state *h_{t-1}*.

The input at the current state x_t , will become the output for the next hidden state or the updated hidden state designated as h_t . The next input becomes the new hidden state h_t , which will be passed into the same function and become input x_{t+1} . The same function f , and the same weights w will be used for every time step of the computation.

1.3 Long Short-Term Memory (LSTM):

LSTM is the most widely used Deep Learning method, and it is the one that is employed in this thesis. It is an artificial recurrent neural network architecture used in the field of Deep Learning. LSTM has feedback links, unlike normal feed-forward neural networks. Not only can it process single data points, it can also process the whole data sequences. LSTMs have a chain-like structure; however, the repeating modules are different. They have cell blocks instead of standard neural network layers, and the cells have various components such as an input gate, a forget gate, and an output gate, as shown in figure 6.

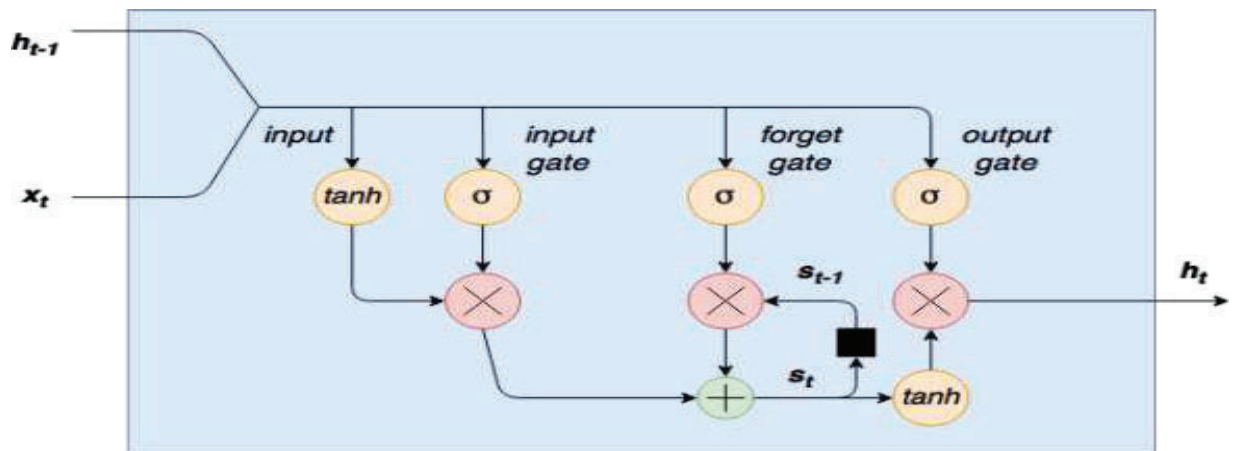


Figure 6: LSTM cell diagram with various components (input, forget gate, and output gate) [10].

There are two values on the left side: x_t is a new sequence value at time t , and h_{t-1} is a previous output at time $t-1$. The first stage for this composite input is squashed by a \tanh layer, which is responsible for taking numbers and transposing them to a percentage between 1 and -1

to ensure that all numbers are homogeneous.

$$f_t = \sigma(w_f [STm_{t-1}, E_t] + b_f) \quad (3)$$

The input subsequently passes into the second state via an input gate. This input gate works to sigmoid switch off unneeded input value items. It has a sigmoid function as activation, and its output vector is the forget valve, which will be applied to the old memory C_{t-1} by element-wise multiplication [14]. A sigmoid function outputs a value between 0 and 1.

$$i_t = \sigma(w_i [h_{t-1}, x_t] + b_i) \quad (4)$$

$$\tilde{c} = \tanh(W_c [h_{t-1}, x_t] + b_c) \quad (5)$$

The next step is the Forget Gate Loop. LSTM cells have an internal state variable added to input data to create an efficient recurrence layer. This additional operation helps to minimize the risk of vanishing gradients. The forget factor is:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \quad (6)$$

This repetition loop is controlled via a forget gate that works the same as the input gate, and helps the network learn state variables that should be “remembered” or “forgotten.” The last step is an output layer tanh squashing function controlled by an output gate. It determines which values are allowed as cell h_t output.

$$o_t = \sigma(w_o [h_{t-1}, x_t] + b_o) \quad (7)$$

CHAPTER 2

RELATED WORKS

A LSTM-RNN-based univariate model for demand-side load prediction was developed and optimized by Salah Bouktif and his team. The model can accurately predict both short-term ranging from a few days to 2 weeks and medium-term ranging from a few weeks to few months. Bouktif introduces seven forecasting techniques in the field of energy prediction that represents a range of commonly used machine learning approaches. The best model to analyze the consumption data is used as the benchmark. Using France Metropolitan electricity consumption data as a case study [3], Bouktif discusses wrapper and embedded feature selection strategies, including the removal of recursive features, and additional regressor trees to validate the significance of model inputs.

Upeka Somaratne, Anupiya Nugaliyadde, and Kok Wai Wong focus on models that can be used to predict short-term, mid-term, and long-term electricity consumption for either a single house or a block of houses. Their research compares ARIMA (Autoregressive Integrated Moving Average), ANN, and DNN. RNN and LSTM measures are established by performing tests for the public London Smart Meter dataset. Although ARIMA shows that short-term predictions perform well, they do not perform well when compared to other models as duration increases. For short-term prediction, RNN and LSTM have been shown to perform similarly to ARIMA, while outperforming all other models in mid-term and long-term predictions [4].

Rahman and team have developed and optimized a novel Deep Recurrent Neural Network (RNN) model with one-hour resolution. Imputation on an electricity usage dataset comprises segments of missing values using the Deep Natural Network . The proposed models are used to predict hourly electricity consumption for the Public Safety Building in Salt Lake City, Utah,

and aggregate hourly electricity consumption in residential buildings in Austin, Texas [5].

Kwok Shah and team have studied nine time series datasets to predict a building's energy consumption requirement, and the datasets are used in other forecasting areas such as economics, quality management, stock market, and weather. These models can also be used in building systems to detect faults over time [6].

CHAPTER 3

Data Source

The capabilities of Deep Learning Neural Networks suggest they are a good fit for time series data. Neural networks can learn an arbitrary complex model from datasets and provide a mapping relationship from input to output. Technically, available sequence contexts provided as input may directly allow neural network models to learn both trends and seasonality. Long Short-Term Memory (LSTM) can solve many time series tasks that are unsolvable by feed-forward networks using fixed-size time windows, and the LSTM networks support efficient learning of temporal dependencies [7]. In our study, we develop an LSTM model for multivariate time series forecasting using Keras library as well as Pearson and Kendall correlation function. Adam and Stochastic Gradient Descent (SGD) optimizers under various batch sizes (50,75, and 100) are also tested and compared.

3.1 Dataset Description:

Our model has been tested on the Hourly Usage of Energy (HUE) Dataset for Buildings in British Columbia, Canada. The HUE dataset contains donated data from residential customers of BC Hydro, a Crown Corporation provincial power utility owned by the government and people of British Columbia. A residential building with a three-year consumption history and a porthole allowing a maximum three-year data download, including weather information, provides the dataset [6]. The household data file consists of date, hour, and energy (kw/h) attributes, as shown in Table 1, and weather data consisting of date, hour, temperature, humidity, pressure, and weather attributes, as shown in table 2.

Table 1 : Residential file data (data, hours, energy_ kwt)

```
household.head(5)
```

	date	hour	energy_kWh
0	2012-06-01	1	1.011
1	2012-06-01	2	0.451
2	2012-06-01	3	0.505
3	2012-06-01	4	0.441
4	2012-06-01	5	0.468

Table 2 : Weather dataset import using pandas.

```
weather.head(5)
```

	date	hour	temperature	humidity	pressure	weather
0	2012-01-01	1	2.4	74.0	102.63	Cloudy
1	2012-01-01	2	2.8	69.0	102.62	Cloudy
2	2012-01-01	3	2.8	69.0	102.69	Cloudy
3	2012-01-01	4	2.6	71.0	102.75	Cloudy
4	2012-01-01	5	2.8	69.0	102.78	Cloudy

3.2 Data Preparation for LSTM:

Basic data preparation constitutes the initial step before advancement to the LSTM approach. Residential and weather files are merged, and based on date and hour, while NA values are replaced with “not recorded” values. New dataset files are then created, totaling four-years of hourly data, as shown in Table 3. The files are saved for merging both similar dataset files based on date and time attributes, thus creating a new dataset file to predict power consumption.

The second step involves a LSTM dataset creation to frame the dataset as a supervised learning problem and to normalize input variables. Supervised learning problems are employed to predict power consumption at a given hour considering power consumption (kw/h) and weather conditions during previous time steps. Pearson and Kendall's correlations are performed to find a relationship between input variables and power consumptions. Corresponding heat maps are also shown in Table 3.

Table 3: Four years of weather and energy data.

	date	hour	temperature	humidity	pressure	weather	energy_kWh
0	2012-06-01	1	13.8	83.0	101.47	Rain Showers	1.011
1	2012-06-01	2	13.3	84.0	101.39	Rain Showers	0.451
2	2012-06-01	3	13.0	83.0	101.39	Rain	0.505
3	2012-06-01	4	12.5	86.0	101.26	Drizzle	0.441
4	2012-06-01	5	12.7	85.0	101.21	Drizzle	0.468

CHAPTER 4

ALGORITHM DESIGN

The LSTM algorithm based on multivariate steps have three main modules, namely data reading, conversion, and LSTM modeling. The data conversion module changes the time series dataset into supervised learning sequences and determines variable datasets with a predictive value.

Algorithm 1 TransformData($data_F, i$) in Data Set

Input: the input raw dataset $data_F$, a positive integer i

Output: the output transformed dataset $data_O$

1. Read files and merge to data
2. **for** each $p \in data_F$ **do**
3. $colData \leftarrow colData.add(p.name)$
4. **If** $data_F.column = null$
5. Replace null with 0
6. **end If**
7. **end for**
8. $data_F \leftarrow data_F[data_F[column]]$
9. $data_O \leftarrow data_F.p[0 : new\ size * i,]$
10. $reshape(data_O)$
11. $plot(data_O)$
12. **return** $data_O$

In algorithm 1, the raw dataset consisting of household and weather files are passed and transformed. The renders dataset is represented as data O in the algorithm. The algorithm steps from

line 1 and 2 read in the input datasets of weather and household consumption and merged them based on a common column. This newly created dataset is checked and the null rows are identified and replaced with estimated values in lines 3-6 . In line 8, the newly merged and verified dataset is plotted to help visualize the data.

Algorithm 2 Multivariate LSTM Time Series

Inputs: Time series data_o

Outputs: RMSE of the forecasted data_o

```
1.    $X \leftarrow train$ 
2.    $Y \leftarrow Validation$ 
3.   model = Sequential()
4.   model.add(LSTM(no. of neurons), statefull=True))
5.   model.compile(loss='mse', optimizer='adam')
6.   for each  $p$  in range( $totalepoches$ ) do
7.       model.fit( $X$ ,  $Y$ , epochs= $i$ , shuffle=False)
8.       model.reset_internalstates()
9.   end for
10. return model

       # Make -step forecast
11.  forecast_lstm(model,  $X$ )
12.   $yhat \leftarrow model.predict(X)$ 
13. return  $yhat$ 
14. # Perform validation on the testing data
15.  for each  $p$  in range( $totallength(testdata)$ ) do
16.      # make a step forecast
17.       $X \leftarrow testdata[p]$ 
18.       $yhat \leftarrow forecast\_lstm(model, X)$ 
19.      #Recording the forecasted data
```

```

20.         predict.append(yhat)
21.         expectdata ← test[i]
22.     end for
23.     MSE ← mse(expected_value, predictions)
    Return (RMSE ← sqrt(MSE))

```

In algorithm two, we pass the dataset transformed from algorithm one, apply a multivariate LSTM approach, and compute the Root Mean Square Error (RMSE) to evaluate the prediction accuracy. Prepared training and testing data are passed to variables X and Y for lines 1 and 2. Data preprocessing for the LSTM model is conducted from lines 3-5. Supervised learning is used and defined by the neurons used in the input layers. The Mean Square Error (MSE) is used for the model, and the Adam optimizer is chosen as a gradient. In lines 6-10, a model is fit and checked for accuracy and data loss. The model runs to determine the total number of epochs and find the training loss. The internal states of the model is refreshed and a final model is constructed, The forecasting for multivariate data occurs in lines 11-13 . The square error means between the expected and predicted values are computed as RMSE for evaluation purpose.

CHAPTER 5

Experimental results

Following the preparation of a household dataset for the LSTM, three data transformations are performed before fitting a model and making a forecast. Time series data need to be made stationary by using lag =1 to remove an increasing trend. Data is subsequently organized with a sliding window to capture previous time step observations in order to forecast a current time step.

Normalization is conducted to address different input variable scales. The learning problem is framed to predict energy consumption at the current hour (t) given energy consumptions and weather conditions at a prior time step. The dataset is transformed using the series-to-supervised () function.

Input values between -1 and 1 are normalized in order to meet the LSTM model's default hyperbolic tangent activation function. Table 4 shows the transformed data for the first six records. Of interest is the impact of the weather conditions, temperature, humidity, and pressure on electricity consumption.

Table 4 : Time series weather dataset consists of data (hours, temperature, humidity, pressure, weather, energy_kw/h, and var1(t))

	Hours	Temperature	Humidity	Pressure	Weather	Energy_KW/h	var1(t)
1	0.000000	0.628342	0.804598	0.527273	0.727273	0.085649	0.045455
2	0.045455	0.614973	0.816092	0.515152	0.727273	0.038207	0.090909
3	0.090909	0.606952	0.804598	0.515152	0.681818	0.042782	0.136364
4	0.136364	0.593583	0.839080	0.495456	0.090909	0.037360	0.181818
5	0.181818	0.598930	0.827586	0.487880	0.090909	0.039648	0.227273

On the other hand, from the below table, we compared different lag numbers (lag=1, lag=2, and lag =5) to see the result of the different sliding window sizes. Training and testing loss decrease as time lag increases. Whereas training and testing accuracy of model increases when time lag increases. From table 5, it can be seen that lag 5 gives the highest accuracy and lowest loss.

Table 5 : Comparison of time lag (leg=1, leg 2& leg=5)with metric

Metric	Lag=1	Lag=2	Lag=5
Train Loss	0.27197579580899245	0.27317812563894	0.264903220806492
Test Loss	0.2827653962881494	0.28460874373428713	0.28298440000462055
Train Accuracy	0.710730593607306	0.71175799086758	0.7324771689497717
Test Accuracy	0.6843821949347659	0.6920555341303509	0.7132445826276539
MSE	0.258064	0.253009	0.225625

Correlations:

Correlation is a bivariate analysis that measures the strength of association between two variables and the direction of the relationship. A value of ± 1 indicates a perfect degree of association between two variables.

A. Pearson :

We perform a *Pearson* Correlation between the input variable and the electricity consumption predictor and plot a heatmap. The Data Frame contains household data between 2012 and 2016 for a single weather station.

Pearson r correlation is the most widely used correlation statistic to measure the degree of relationship between linearly related variables.

$$r_{xy} = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n \sum x_i^2 - (\sum x_i)^2} \sqrt{n \sum y_i^2 - (\sum y_i)^2}} \quad (9)$$

r_{xy} =Pearson

r correlation coefficient between x and y

n = number of observations

x_i = value of x (for ith observation)

y_i = value of y (for ith observation)

As shown in Table 6, each matrix entry contain correlation value between a weather-related variable and energy consumption for residential buildings.

Table 6 :Pearson correlation between variables

	Hours	Temperature	Humidity	Pressure	Weather	Energy_KW/h
Hours	1.000000	0.169634	-0.245732	-0.029589	0.008227	0.439296
Temperature	0.169634	1.000000	-0.417683	-0.257251	0.099938	-0.073435
Humidity	-0.245732	-0.417683	1.000000	-0.043905	-0.039370	0.000469
Pressure	-0.029589	-0.257251	-0.043905	1.000000	-0.076648	-0.034855
Weather	0.008227	0.099938	-0.039370	-0.076648	1.000000	-0.011065
Energy_KW/h	0.439296	-0.073435	0.000469	-0.034855	-0.011065	1.000000

The heat map can demonstrate the correlation between various weather conditions and energy consumption. A correlation function `.corr` from pandas' package is calculated, and a correlation matrix is generated as shown in Figure 7. Blue means positive while yellow means negative, and the stronger the color, the larger the correlation magnitude.

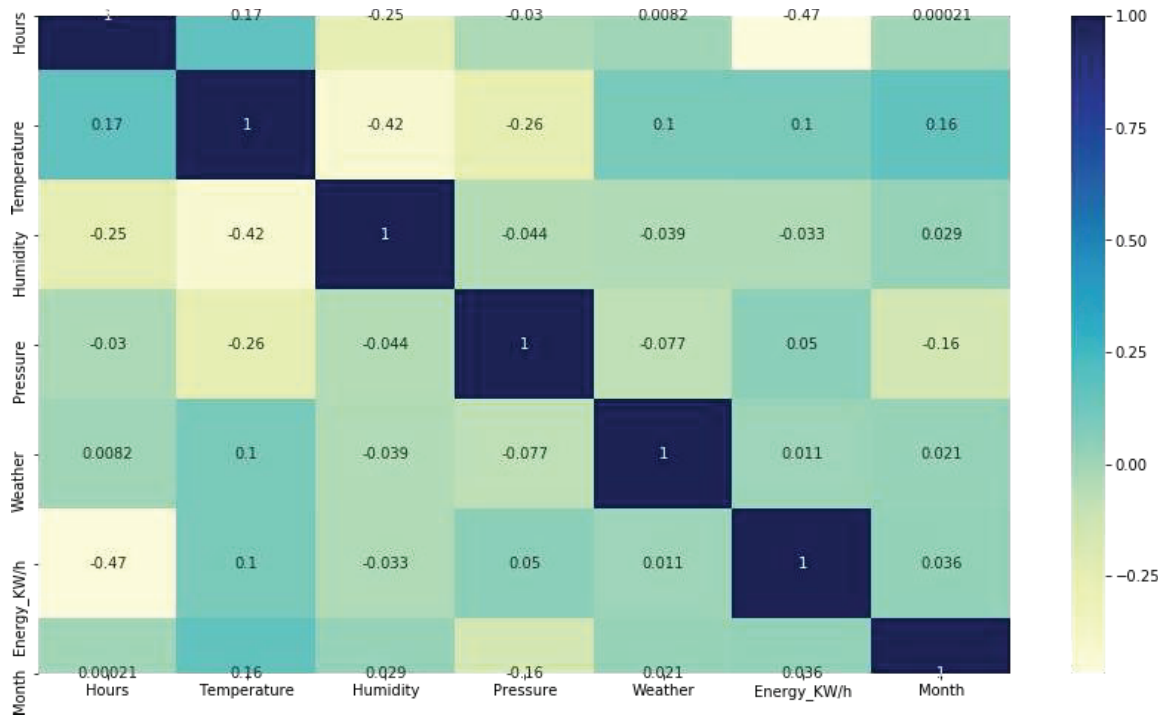


Figure 7: Heat map for Pearson correlation between variables

Blue indicates a positive correlation, and the darker the color, the stronger the correlation. Using the heat map in Figure 7 as an example, if one wants to know the correlation between weather and humidity, it can be seen that there is no correlation based on the value of -0.039. A correlation between hours of the day and energy consumption is easily discerned. Humidity and temperature show a negative correlation, meaning that as temperature rises, humidity decreases.

B. Kendall rank correlation(non- parametric):

The *Kendall* Correlation between the input variable and the electricity consumption predictor along with the plotted heat map is displayed in Table 7. The Kendall rank correlation (non-parametric) is an alternative to Pearson's correlation (parametric) and is used to test similarities in ordering data when quantities includerank. Other types of correlation coefficients use observations as the correlation basis. Kendall's correlation coefficient uses pairs of observations

and determines the strength of association based on the pattern of concordance and discordance between pairs. The Kendall Correlation between weather variables (pressure, humidity, and weather) and the electricity consumption predictor generates the heat map.

Table 7: Kendall correlation between dataset variables

	Hours	Temperature	Humidity	Pressure	Weather	Energy_KW/h
Hours	1.000000	0.111041	-0.168589	-0.023209	-0.001488	0.395454
Temperature	0.111041	1.000000	-0.343640	-0.184341	0.064808	-0.032665
Humidity	-0.168589	-0.343640	1.000000	0.005374	-0.003560	-0.023464
Pressure	-0.023209	-0.184341	0.005374	1.000000	-0.059159	-0.031770
Weather	-0.001488	0.064808	-0.003560	-0.059159	1.000000	-0.002189
Energy_KW/h	0.395454	-0.032665	-0.023464	-0.031770	-0.002189	1.000000

Finally, inputs (X) are reshaped into the 3D format expected by LSTMs, namely samples, timesteps, and features. The training and testing datasets have about 17K hours of data for training and about 10K hours for testing.

Batch Sizes:

Batch size is one of the most important hyperparameters to tune as it defines the number of samples propagated through a network. The smaller the batch, the less accurate the gradient estimate will be. However, a smaller batch size means less memory and faster computation. Practitioners often want to use a large batch size to train their model as it allows a computational speedup from the GPU's parallelism.

LSTM has 50 neurons in the first hidden layer and one neuron in the output layer in order to predict energy consumption. The input shape is a one-time step with six features. A Mean Absolute Error (MAE) loss function and an efficient Adam version of stochastic gradient descent is used. Additionally, the experiment is conducted with three different batch sizes (50, 72, and 100) to indicate which batch size produces the best results with the least amount of loss and with a high degree of accuracy.

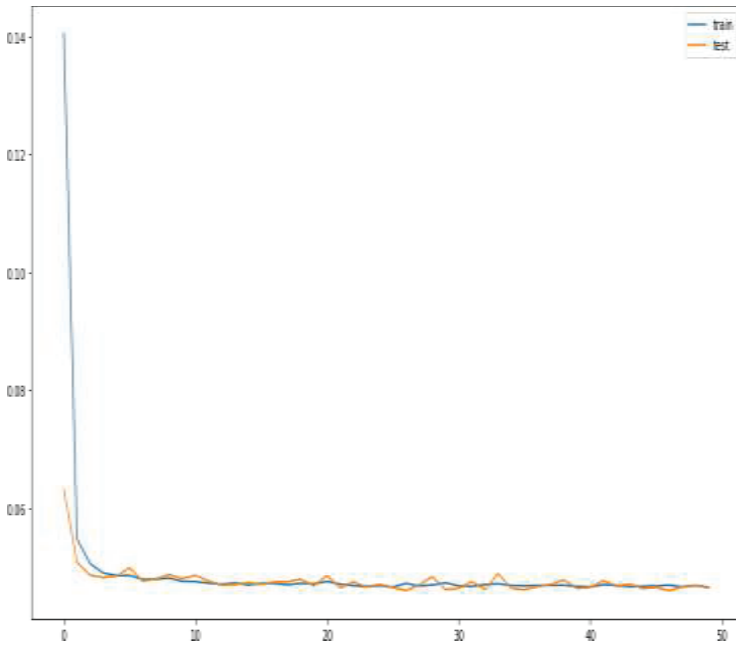


Figure 8: Results for batch size 50

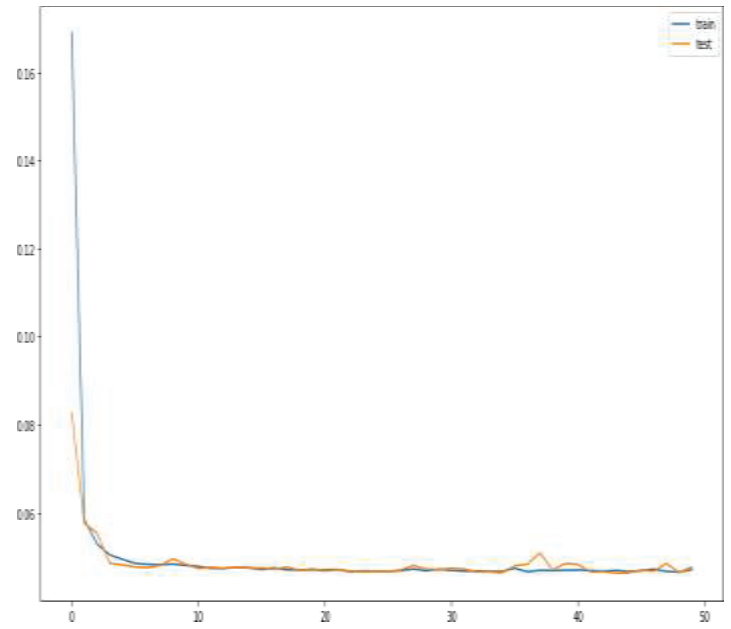


Figure 9: Results for batch size 72

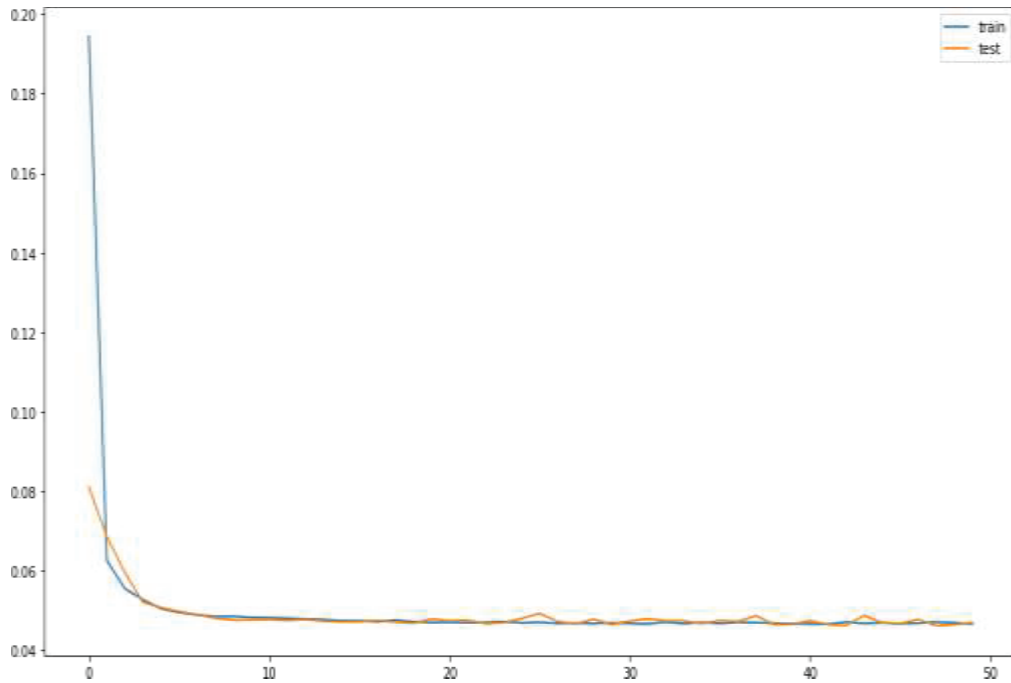


Figure 10: Results for batch size 100

Explanation for These Different Batch Size:

As shown the Figure 8, 9 and 10, the blue line represents training loss, whereas the orange line represents validation loss. The X-axis shows total epochs while the Y-axis represents the training and testing loss during each period. As seen in Figure 8, maximum training loss is 0.14, while extreme validation loss is 0.06 that reduce as epochs increase. Figure 9 shows that the maximum training loss is 0.16, while the maximum validation loss is 0.08, which reduces as epochs increase. Figure 10 shows that the extreme training loss is 0.20, while the maximum validation loss is 0.08 that reduces as epochs increase. Based on the above observations, we choose the lowest batch size of 50 for higher accuracy and minimal loss.

Table 8: The comparison of batch size vs metric.

Batch Size : 50						
Metric	Epoch1	Epoch 10	Epoch 20	Epoch 30	Epoch 40	Epoch 50
Train Loss	0.1388	0.0489	0.0471	0.0475	0.0473	0.0471
Test Loss	0.0765	0.0495	0.0472	0.0475	0.0469	0.0468
Train Precision	0.4158	0.5593	0.5604	0.5593	0.5586	0.5599
Test Precision	0.5539	0.5535	0.5786	0.5714	0.5830	0.5666
Train Recall	0.4736	0.6368	0.6368	0.6368	0.6368	0.6368
Test Recall	0.6045	0.6362	0.6362	0.6362	0.6362	0.6362
Train F1-Score	0.4414	0.5954	0.5959	0.5953	0.5949	0.5960
Test F1-Score	0.5774	0.5917	0.6060	0.6019	0.6084	0.5992
Batch Size 72						
Metric	Epoch1	Epoch 10	Epoch 20	Epoch 30	Epoch 40	Epoch 50
Train Loss	0.1488	0.0482	0.0488	0.0469	0.0478	0.0473
Test Loss	0.0749	0.0478	0.0485	0.0472	0.0488	0.0467
Train Precision	0.4045	0.5598	0.5609	0.5601	0.5608	0.5603
Test Precision	0.5561	0.5674	0.5814	0.5437	0.5813	0.5461
Train Recall	0.4581	0.6369	0.6369	0.6369	0.6369	0.6369
Test Recall	0.6051	0.6364	0.6364	0.6364	0.6364	0.6364
Train F1-Score	0.4273	0.5954	0.5963	0.5959	0.5962	0.5957
Test F1-Score	0.5792	0.5998	0.6364	0.5863	0.6076	0.5877
Batch Size 100						
Metric	Epoch1	Epoch 10	Epoch 20	Epoch 30	Epoch 40	Epoch 50
Train Loss	0.2058	0.0495	0.0474	0.0475	0.0470	0.0472
Test Loss	0.1107	0.0485	0.0482	0.0467	0.0472	0.0477
fTrain Precision	0.3065	0.5611	0.5598	0.5595	0.5602	0.5591
Test Precision	0.5075	0.5629	0.5770	0.5740	0.5748	0.5768
Train Recall	0.3545	0.6368	0.6368	0.6368	0.6368	0.6368
Test Recall	0.6222	0.6362	0.6362	0.6362	0.6362	0.6362

Train F1-Score	0.3228	0.5963	0.5957	0.5954	0.5959	0.5952
Test F1-Score	0.5572	0.5972	0.6051	0.6034	0.6038	0.6049

Table 8: The comparison of batch size vs metric

Metric	Batch size 50	Batch size 72	Batch size 100
Train Loss	0.04713788243376365	0.04896468774045575	0.04727578801910933
Test Loss	0.04705696579071549	0.04791939740524351	0.04876118863434668
Train Precision	0.5670033688414586	0.5448899116145965	0.5666582686171684
Test Precision	0.5451831038141924	0.5782648367373351	0.5598513563027028
Train Recall	0.6346587823406202	0.6346587823406202	0.6346587823406202
Test Recall	0.6342992508323155	0.6342992508323155	0.6342992508323155
Train F1-Score	0.5985583281408162	0.5859758628557806	0.5983479982097399
Test F1-Score	0.5860141067765743	0.6047965267703758	0.5943351973193092

The above table shows the results of various batch sizes. Models are fitted with three different batch sizes: 50, 72 and 100. The training loss for batch 72 is higher than that of batch size 50. The test loss for batch 50 is the lowest and batch 100 has the highest test loss. Train precision is the lowest for batch 72, and the testing precision value is the lowest for batch size 50. Recall values for training and testing is the same for all batch sizes. The F1-score is the lowest for batch size 50. It appears that smaller batch size results in less model loss.

Optimizers:

In order to reduce losses, optimizers are used to change the attributes of a neural network such as weights and learning rate. We compare two popular optimizers, namely Stochastic Gradient Descent (SGD) and Adam.

1) Stochastic gradient descent(SGD):

SGD is a first-order optimization algorithm using the first order derivative of a loss function. It calculates how weights are updated in order to reach the minimum for function. Batch gradient descent results in redundant computations for large datasets, as well as recomputes gradients for repeated examples before each parameter update. SGD is usually much faster and can be used to learn online. The example in the training group is randomly shuffled, and a learning rate η and an initial vector of parameters w are chosen. This is repeated until the minimum is obtained.

2) Adam:

Adaptive Moment Estimation works with momentums of the first and second order. The velocity should not be too high since the minimum could be jumped over. Adam integrates both RMSProp and Momentum. Momentum (v) provides a short-term memory to the optimizer. Rather than trusting the present gradient, preceding and current gradients are used to compute a ratio β_1 . [13] RMSProp takes 1/square root of the gradient into consideration. It means that the optimizer takes a larger step if the variance is small (confident), and vice versa. [13]

Table 10 : Comparison between optimizer Adam and SGD

Metric	Adam	SGD
Train Loss	0.04713788243376365	0.0553449156332642
Test Loss	0.04805696579071549	0.052878668074217705
Train Precision	0.5670033688414586	0.5785953706257964
Test Precision	0.5451831038141924	0.573666933453687
Train Recall	0.6346587823406202	0.6346587823406202
Test Recall	0.6342992508323155	0.6342992508323155
Train F1-Score	0.5985583281408162	0.6051398191277839
Test F1-Score	0.5860141067765743	0.6022044993939164

As demonstrated in the above table, the training and testing loss from ADAM is less than the SGD optimizer. Train precision for Adam is greater compared to SGD. The testing precision of SGD is slightly higher. The recall value for both training and testing data is equal.

$$\text{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{Total}}$$

Whereas F1-score is represented as:

$$\text{F1 Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{10}$$

F1 score for both training and testing data for Adam optimizer is greater than SGD. Thus, Adam optimizer seems to be the best choice for our model.

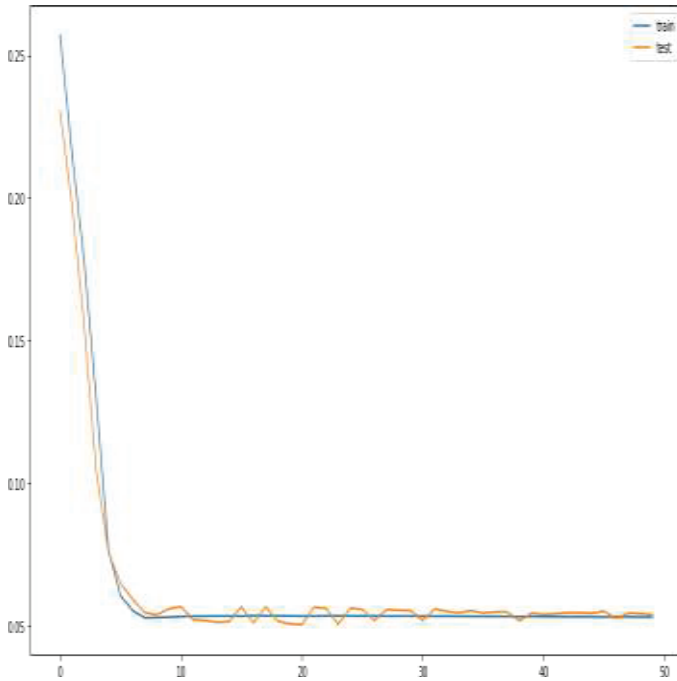


Figure 11: SGD Optimizer.

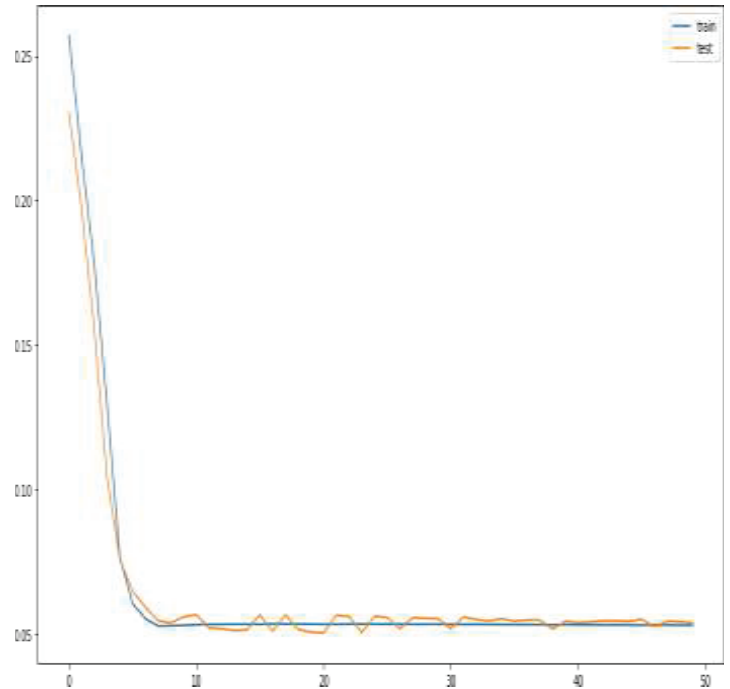


Figure 12: Adam Optimizer

As shown in Figure 11, epoch=50, batch size=50, and optimizer =SGD/Adam. The X-axis shows total epochs while the Y-axis represents the training and testing loss during each period. The maximum value for training loss is 0.25 and the maximum value for validation loss is 0.23, whereas (Figure12) the maximum training loss value is 0.14. The validation loss maximum value is 0.06. Loss from Adam is much less than that of SGD. Adam is more accurate when compared to SGD. We keep track of both the training and test loss during training by setting the validation data argument in the fit() function.

Evaluation:

After the model fit, forecasting for the entire test dataset is performed. The Root Mean Squared Error (RMSE) for the model is calculated using forecasted/predicted and actual values.

Where Y_t is the actual value of a point for a given period t , n is the total number of fitted points and \hat{Y}_t is the fitted forecast value for the period t . Train and test losses are printed at the end of each training epoch. At the end of the run, the final RMSE of the model on the test dataset is printed, and the model achieves a respectable RMSE of 0.214.

System Modifications:

Based on promising results, we convert the numeric energy consumption value to a categorical value in order to achieve a better prediction result. The following rules are used:

High: if consumption kWh is greater and equal to 1.0

Medium: if consumption kWh is between 0.5 and 1.0

Low: if consumption kWh is less than 0.5

Table 11: Categorical energy consumption level

date	hour	temperature	humidity	pressure	weather	energy_kWh
2012-06-01	1	13.8	83.0	101.47	Rain Showers	high
2012-06-01	2	13.3	84.0	101.39	Rain Showers	low
2012-06-01	3	13.0	83.0	101.39	Rain	medium
2012-06-01	4	12.5	86.0	101.26	Drizzle	low
2012-06-01	5	12.7	85.0	101.21	Drizzle	low

Table 12: Transformed dataset

	Hours	Temperature	Humidity	Pressure	Weather	Energy_KW/h	var1(t)
1	0.000000	0.628342	0.804598	0.527273	0.727273	0.0	0.045455
2	0.045455	0.614973	0.816092	0.515152	0.727273	0.5	0.090909
3	0.090909	0.606952	0.804598	0.515152	0.681818	1.0	0.136364
4	0.136364	0.593583	0.839080	0.495456	0.090909	0.5	0.181818
5	0.181818	0.598930	0.827586	0.487880	0.090909	0.5	0.227273

Table 13: *Pearson* correlation.

	Hours	Temperature	Humidity	Pressure	Weather	Energy_KW/h
Hours	1.000000	0.169634	-0.245732	-0.029589	0.008227	-0.466126
Temperature	0.169634	1.000000	-0.417683	-0.257251	0.099938	0.100333
Humidity	-0.245732	-0.417683	1.000000	-0.043905	-0.039370	-0.032907
Pressure	-0.029589	-0.257251	-0.043905	1.000000	-0.076648	0.050179
Weather	0.008227	0.099938	-0.039370	-0.076648	1.000000	0.011228
Energy_KW/h	-0.466126	0.100333	-0.032907	0.050179	0.011228	1.000000

Table 14: Kendall correlation

	Hours	Temperature	Humidity	Pressure	Weather	Energy_KW/h
Hours	1.000000	0.111041	-0.168589	-0.023209	-0.001488	-0.377004
Temperature	0.111041	1.000000	-0.343640	-0.184341	0.064808	0.098599
Humidity	-0.168589	-0.343640	1.000000	0.005374	-0.003560	-0.030394
Pressure	-0.023209	-0.184341	0.005374	1.000000	-0.059159	0.033513
Weather	-0.001488	0.064808	-0.003560	-0.059159	1.000000	0.009093
Energy_KW/h	-0.377004	0.098599	-0.030394	0.033513	0.009093	1.000000

Numeric energy consumption is now changed to high, medium and low as categorical variables. When the weather seems to be mostly cloudy, energy consumption appears to be high. Most energy consumption occurs between 9:00 a.m. and 1:00 p.m.

Monthly Energy Consumption:

We also want to investigate how energy consumption changes with different months. To determine that, the month is extracted from the date and a new month column is appended to the dataset. The data shows that the people tends to use more electricity in cold months. It is probably due to heating need.

Table 15: Month Column in the dataset

date	hour	temperature	humidity	pressure	weather	energy_kWh	month
2012-06-01	1	13.8	83.0	101.47	Rain Showers	high	6
2012-06-01	2	13.3	84.0	101.39	Rain Showers	low	6
2012-06-01	3	13.0	83.0	101.39	Rain	medium	6
2012-06-01	4	12.5	86.0	101.26	Drizzle	low	6
2012-06-01	5	12.7	85.0	101.21	Drizzle	low	6

Table 16 : Month with energy consumption

	Hours	Temperature	Humidity	Pressure	Weather	Energy_KW/h	Month
1	0.000000	0.628342	0.804598	0.527273	0.727273	0.0	0.454545
2	0.045455	0.614973	0.816092	0.515152	0.727273	0.5	0.454545
3	0.090909	0.606952	0.804598	0.515152	0.681818	1.0	0.454545
4	0.136364	0.593583	0.839080	0.495456	0.090909	0.5	0.454545
5	0.181818	0.598930	0.827586	0.487880	0.090909	0.5	0.454545
	var1(t)						
1	0.045455						
2	0.090909						
3	0.136364						
4	0.181818						
5	0.227273						

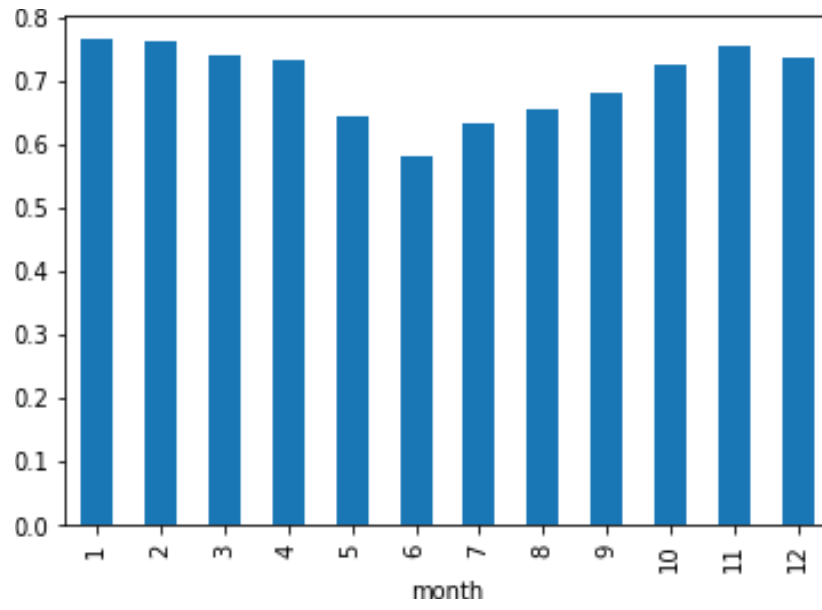


Figure 13: Energy consumption versus months.

CHAPTER 6

Conclusion and Future Work

In the past decades, electric power systems (EPSs) have developed from physical structures to smart grids with intelligent monitoring and control technologies. Governments and power companies have installed smart meters in almost every home and building. In this thesis, Deep Learning is used to create a predictive energy consumption system using the LSTM model.

With multivariate time series data, the LSTM model is employed to predict energy consumption by building a model with historic datasets. Merging and preprocessing datasets in order to fit the LSTM model is performed first. Then different batch sizes and two optimizers are used and compared. The Adam optimizer and small batch sizes seem to result in higher accuracy and less loss for both training and the validation dataset.

Correlations between various weather inputs and energy output are also studied using Pearson and Kendall correlations. It is observed that when the weather is mostly cloudy, energy consumption tends to be high. High energy consumption often happens in the morning, and energy consumption is always at the highest during cold seasons.

References

- [1] Bouktif, S., Fiaz, A., Ouni, A. and Serhani, M.A., 2018. Optimal deep learning lstm model for electric load forecasting using feature selection and genetic algorithm: Comparison with machine learning approaches. *Energies*, 11(7), p.1636.
- [2] Nugaliyadde, Anupiya, Upeka Somaratne, and Kok Wai Wong. "Predicting Electricity Consumption using Deep Recurrent Neural Networks." *arXiv preprint arXiv:1909.08182* (2019).
- [3] Rahman, A., Srikumar, V. and Smith, A.D., 2018. Predicting electricity consumption for commercial and residential buildings using deep recurrent neural networks. *Applied energy*, 212, pp.372-385.
- [4] Deb, C., Zhang, F., Yang, J., Lee, S.E. and Shah, K.W., 2017. A review on time series forecasting techniques for building energy consumption. *Renewable and Sustainable Energy Reviews*, 74, pp.902-924.
- [5] Mocanu, E., Nguyen, P.H., Gibescu, M. and Kling, W.L., 2016. Deep learning for estimating building energy consumption. *Sustainable Energy, Grids and Networks*, 6, pp.91-99.
- [6] Makonin S. HUE: The hourly usage of energy dataset for buildings in British Columbia. 2018 Sep 3.
- [7] Graves, A., Mohamed, A.R. and Hinton, G., 2013, May. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing* (pp. 6645-6649). IEEE.
- [8] "Keras LSTM tutorial - How to easily build a powerful deep learning language model," *Adventures in Machine Learning*, 30-Oct-2020.
- [9] Iea, "World Energy Outlook 2019 – Analysis," *IEA*, 01-Oct-2020. [Online]. Available: <https://www.iea.org/reports/world-energy-outlook-2019>.
- [10] Chm, "Convolutional Neural Networks for Beginners using Keras and TensorFlow 2," *LaptrinhX*, 21-Apr-2020.
- [11] H. Pokharna, "The best explanation of Convolutional Neural Networks on the Internet," *Medium*, 28-Jul-2016.

- [12] “CS231n: Convolutional Neural Networks for Visual Recognition,” *Stanford University CS231n: Convolutional Neural Networks for Visual Recognition*. [Online]. Available: <http://cs231n.stanford.edu/>.
- [13] Nokknock “ADAM in 2019- What's the next ADAM optimizer,” Medium, 06-Oct-2020
- [14] S. Yan, “Understanding LSTM and its diagrams,” *Medium*, 15-Nov-2017 <https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714>.