Capstone Project Draft
Energy Forecasting for a UK Motor Manufacturing Plant

Samantha Knox
Data Analytics Degree Apprenticeship
Newcastle University
2022
s.knox3@newcastle.ac.uk

Acknowledgements

Thank you to Nissan Motor Manufacturing UK Ltd for providing the data and hardware used to train models throughout this project. Thank you to Dr Stephen McGough (Newcastle University) and James Maddison (Nissan) for supervising this project.

Contents

1 Executive Summary

With the recent introduction of data visualisation software at Nissan, awareness of energy consumption is being observed on a wider scale than ever before. Coupled with exponentially increasing energy costs across the UK, awareness of company spending on energy is also higher than ever. Advancement within the company to install measuring infrastructure has made it possible to now collect, analyse and present data rapidly, in ways that were previously impossible. This report aims to support Nissan in the next stage of their digital intelligence development journey to look at future forecasting methods that can make predictions for energy consumption, allowing the company to make more proactive decisions.

The purpose of this project was to build and train a machine learning model that could accurately predict future energy consumption within the Nissan Paintshop, based on previous data gathered across the site over a 15 month period.

In this work, three different machine learning algorithms were modelled to develop a time-series power forecast. The proposed models were Long Short-Term Memory (LSTM), the Gated Recurrent Unit (GRU) and the Recurrant Neural Network (RNN). These models were selected because they are traditionally suited to this type of sequence-dependent, time-series analysis.

Prior to the models being applied, the data was amalgamated from various sources into a single file using Python within a Jupyter notebook (Appendix A). This was then cleaned and pre-processed using cyclical encoding and normalization methods (Appendix B) so that it could be further explored. The data was segregated into training, validation and test sets, which provided the opportunity for an unseen data set to be used when gathering results to ensure an unbiased experiment.

The initial results were recorded and improved further by adjusting hyperparameters, namely the learning rate and number of epochs. The final results showed an improvement by increasing the number of epochs (from 15 to 50) and maintaining the same learning rate (0.001). Mean squared error was used to compare the success of the three models. The GRU model performed the best, as expected, as this model is well suited to smaller time-series datasets such as this.

Nissan has never previously collected data of this volume or carried out data analysis on such a large scale. Whilst the company have previously carried out some machine learning activities for action recognition and categorisation (for identifying defects within vehicles in real-time), forecasting is a novel concept for the company. It is proposed that the activity is rolled out across the wider plant in the near future, as the raw energy data already exists for the other shops.

2 Introduction

Electricity demand analysis has traditionally been done by means of classical statistical tools based on time series models (Óscar Trull et al. [2019]). In the recent years, machine learning techniques have been successfully applied for electricity demand forecasting, due to their ability to capture complex non-linear relationships in the data. Time series forecasts are developed based on time series analysis, which comprises methods for analysing time series data to extract meaningful statistics and other characteristics of the data. The goal of time series forecasting is to predict a future value or classification at a particular point in time. Machine learning can be used to provide a method to make such predictions. Machine learning is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning models use historical data as an input to predict new output values.

Time series forecasting starts with a historical time series. The historical data is analysed for patterns of time decomposition, such as trends, seasonal patterns, cyclical patterns, and regularity. Time series data can be defined as a chronological sequence of observations on a target variable.

Time series analysis provides a body of techniques to better understand a dataset. The most useful of these include trends, which show the optional and often linear increasing or decreasing behavior of the series over time and seasonality, which shows any repeating patterns or cycles of behavior over time. Finally noise, which determines the optional variability in the observations that cannot be explained by the model.

Often, series problems are real-time, continually providing new opportunities for prediction. This adds accuracy to time series forecasting that quickly identifies bad assumptions and errors in modeling.

This project sought to explore 15 months of historical Nissan energy and production data from 11th December 2020 to 15th April 2022, which sufficiently exposed the model to all seasons across a year. This data was used to build and train a machine learning model to accurately predict Nissan site energy consumption for one shop (Piantshop) within the plant to predict anticipated energy consumption. Production hours and volumes were taken into account to more accurately estimate demand whilst manufacturing vehicles.

Three different neural networks were considered in this investigation: LSTM, GRU and RNN. All RNNs have feedback loops in the recurrent layer, which lets them maintain information in their 'memory' over time. However, it can be difficult to train standard RNNs to solve problems that require learning long-term temporal dependencies, such as this. This is due to the gradient of the loss function decays exponentially with time (the vanishing gradient problem, Hochreiter [1998]), hence why the RNN may have performed worst of the three models. LSTM networks are a type of RNN that uses special units in addition to standard units. Their gated architecture (input, output, forget) lets them learn longer-term dependencies. GRU are similar to LSTM but less complex as they have fewer number of gates and are therefore suited to smaller datasets, whereas LSTM is suited to larger datasets. The Nissan Energy Dataset [1] used for this project is not particularly large, therefore was better suited to the GRU model, which performed best of all. After carrying out the investigation, the most accurate results came from the GRU model after tweaking several hyperparameters (training with 50 epochs and setting the learning rate to 0.001). This yielded an energy profile that closely matched the actual results of the validation dataset.

Mean Squared Error (MSE) was used to evaluate the model. The mean squared error (MSE) shows how close a regression line is to a set of points. It does this by taking the distances from the points to the regression line (i.e the actual values) and squaring them. The squaring is necessary to remove any negative signs. It also gives more weight to larger differences. The lower the MSE, the better the forecast.

[1] Created as part of this investigation from several other data sources

3 Background

3.1 Context

Nissan Sunderland opened in 1986 and [2] built its 10 millionth car in 2019. By 2021, the unit cost of energy had doubled and is set to increase further in 2022 following the announcement by the chief executive of Energy UK in Q4 of 2021 that energy prices are expected to rise by at least another 30-50 percent (Forbes [2022]), which will cost the company tens of millions of pounds.

Nissan have made energy reduction activities a key objective in 2022, in order to reduce costs where possible and support their target of achieving Net Zero by 2050 (Lange et al. [1997]). Matching electrical energy consumption with the right level of supply is important, because excess electricity supplied cannot be stored, unless converted to other forms, incurring additional costs and resources (Shaqsi et al. [2020]). Additionally, underestimating energy consumption could have negative repercussions, with excess demand overloading the supply line and even causing blackouts (RAE [2014]). Therefore, there is a clear benefit in monitoring the energy consumption of buildings, including offices and production facilities.

Nissan are currently planning the installation of a 20MW solar farm to supply energy to the site, including a new £1 billion Gigafactory (Jolly [2021]). The challenge of accurately estimating future energy requirements when determining the specification for the new site relies heavily on understanding what NMUK currently consume and when. Additionally, managers would gain key insights into factors affecting their building's energy demand, whereby forecasts provide a benchmark to single out anomalously high/low energy consumption and alert managers to faults within the building. This would provide opportunities to address high consumption and improve energy efficiency.

3.2 Defining Neural Network Architecture

A network architecture defines the way in which a machine learning model is structured and more importantly, what it's designed to do. The architecture will determine the model's accuracy (a network architecture is one of many factors that impacts accuracy), what the model can predict, what the model expects as input and output and the combination of layers and how data flows through the layers

The three models that were used for this investigation were RNN, LSTM and GRU.

3.2.1 RNN

Recurrent Neural Networks are a Deep Learning and Artificial Neural Network design that is suited for sequential data processing. They are ideal for time series data, such as this. Normally, feed forward neural networks are only meant for independent data points. However, when data is sequenced such that one data point depends on the previous, the network needs to incorporate the dependencies between the points. RNNs have a "memory" that helps them store the states of previous inputs to generate the next output of the sequence. The key benefit of employing RNNs instead of conventional neural networks is that the characteristics in standard neural networks are not shared. In RNN, weights are shared over time. RNNs can recall their prior inputs, whereas Standard Neural Networks cannot.

In the forward pass of an RNN, the network computes the values of the hidden units and the output after k time steps. Each recurrent layer has two sets of weights; one for the input and one for the hidden unit. The last feed-forward layer computes the final output for the kth step.

3.2.2 LSTM

Long Short Term Memory is a special kind of RNN capable of learning long term sequences as introduced by Hochreiter and Schmidhuber [1997]. It is designed specifically to avoid long term dependency problems. It works by remembering long sequences for a long period of time. LSTMs were also designed to address the vanishing gradient problem in RNNs. LSTM uses gates as filters that remove unwanted selected and irrelevant information. There are a total of three gates that LSTM

[2] https://www.topgear.com/car-news/top-gear-advice/beginners-guide-nissan-sunderland

uses: Input Gate, Forget Gate, and Output Gates. Similar to GRU, these gates determine which information to retain. LSTM uses these gates as filters to remove unwanted selected and irrelevant information. The input gate decides what information will be stored in long term memory. It works with the information from the current input and short term memory from the previous step. At this gate, it filters out the information from variables that are not useful. The forget gate decides which information from the long term memory will be kept or discarded by multiplying the incoming long term memory by a forget vector generated by the current input and incoming short memory. The output gate will take the current input, the previous short term memory and the newly computed long term memory to produce a new short-term memory which will be passed on to the cell in the next time step. The output of the current time step can also be drawn from this hidden state.

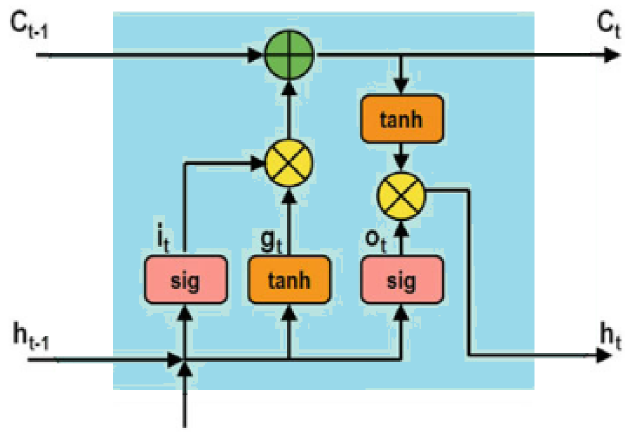


Figure 1: Internal Structure of LSTM Model

The scheme of an LSTM cell can be seen in the Figure 1, where ft is the forget gate, it is the input gate and ot the output gate. ft decides what information should be thrown away or saved. A value close to 0 means that the past information is forgotten while a value close to 1 means that it remains. It decides what new information ot to use to update the ct memory state. Thus, ct is updated using both ft and it. Finally, ot decides which is the output value that will be the input of the next hidden unit.

The computation process can be given as follows (Tovar et al. [2020]):

$ft = \sigma(wf[ht\text{-}1, Xt] + bf)$

$it = \sigma(wi[ht\text{-}1, Xt] + bi)$

$ot = \sigma(wo[ht\text{-}1, Xt] + bo)$

$at = tanh(wa[ht\text{-}1, Xt] + ba)$

$ct = ft \times ct\text{-}1{+}it \times at$

$ht = ot \times tanh(ct)$

where σ is the sigmoid activation function and it can be defined as:

$\sigma(x) = (1 + e\text{-x})\text{-}1$

3.2.3 GRU

GRU networks are designed to handle the vanishing gradient problem. They have a reset and update gate, which determine the information to be retained for future predictions. The workflow of the GRU is the same as the RNN. To solve the vanishing gradent problem faced by standard RNN, GRU incorporates the two gate operating mechanisms called Update gate and Reset gate. The Update Gate is responsible for determining the amount of previous information that needs to pass along to the next state. This allows the model to copy all the information from the past and eliminate the risk of the vanishing gradient. The Reset Gate is used by the model to decide how much of the past information is needed to be neglected, therefore deciding whether the previous cell state is important or not. Finally the non-linear activation function is applied and the next sequence is generated.

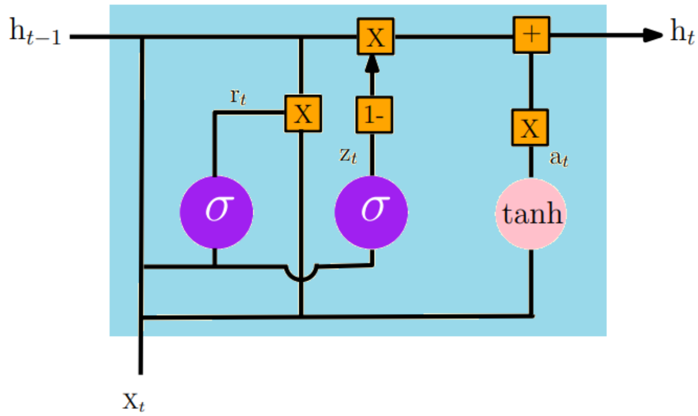


Figure 2: Internal Structure of GRU Model

According to Figure 2, the formulas of GRU can be given as:

$zt = \sigma(wz[ht\text{-}1, Xt] + bz)$

$rt = \sigma(wr[ht\text{-}1, Xt] + br)$

$at = tanh(rt \times wa[ht\text{-}1, Xt] + ba)$

$ht =$ (1-zt) $\times\, at + zt \times ht\text{-}1$

where Xt is the vector input of training data at time t, and ht is the outcome of the current layer at time t. zt and rt represent the update and the reset gates respectively. at is the candidate activation.

4 Method / Work Undertaken

To predict the energy consumption, the three neural network models were used and the methodology shown in Figure 3 was followed, which details the steps used to construct the proposed predictive models. The first step of analysing and pre-processing the time series data was arguably the most important (and time-consuming) step.

4.1 Data Preprocessing and Exploratory Analysis

Four separate datasets were obtained from Nissan for use in this project. The Energy Meter Dataset, The Paintshop Calendar, V070 Achievement Data and The Shop Meter Dataset. An additional validation dataset was added later to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters. All data files and Jupyter notebooks are contained within the appendices.

The Energy Meter Dataset was collected from buss-bar mounted energy smart meters [3] and stored on the Nissan server in .avro file format. This was converted into .csv format and read into Python using a Jupyter Notebook. Preliminary analysis of the data showed that every meter records a reading once per minute. Every 5 minutes, the files are merged together, then once per hour a file is populated. The file name is of the format shown in Figure 4.

The filename is generated at the time the hourly merge is created. This is not the time the data was recorded, although it will be within an hour. Within the file itself, 40 columns exist, containing both categorical and numerical data. The majority of the data is of no relevance to this investigation, so was stripped out to leave the fields in Figure 5.

[3] https://www.elcomponent.co.uk/wp-content/uploads/2019/03/AEM33-Product-Specification.pdf

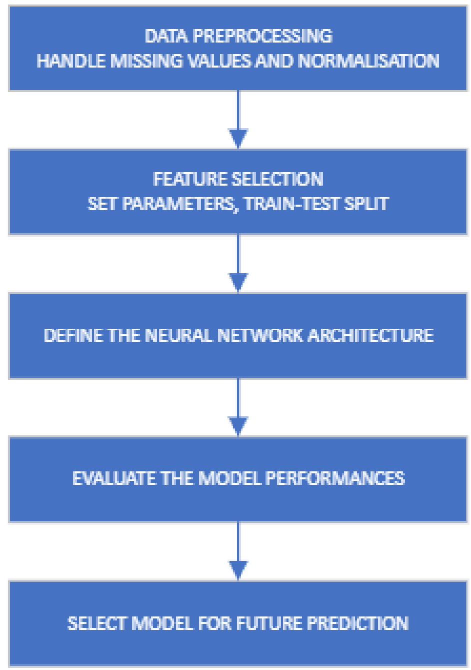


Figure 3: The Proposed Methodology

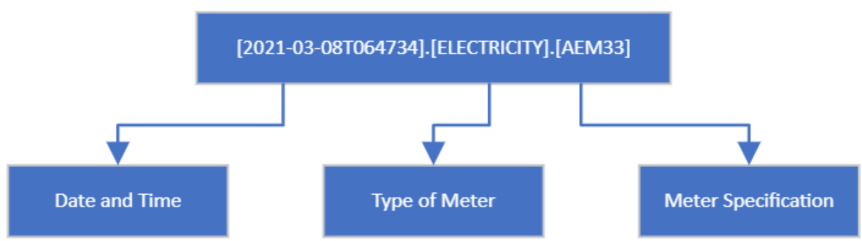


Figure 4: Meter Data File Name Format

Column Name	**Description**
Meter name	*The name of the individual smart meter*
Timestamp	*Time of read in epoch time*
Totalizer kwh	*Kilowatts per hour usage*
Totalizer kwh scaled	*Kilowatts per hour normalized using scale factor*

Figure 5: Useful Column Names

The unix timestamp was unreadable in this format, so was converted to a readable date using a Pandas dataframe. The kwh values could then be plotted over the duration for each meter within the Paintshop (see Figure 6).

The Paint Calendar dataset includes the time and date of production, which allowed periods of energy use to be split into production hours and non-production hours. This was useful in the prediction for attributing energy consumption to times for planned production, which are expected to be higher.

The A070 Dataset refers to a production achievement point, in this case, a point between Paintshop and Trim & Chassis. At this stage in the process, painted car body shells are shipped to the Trim &

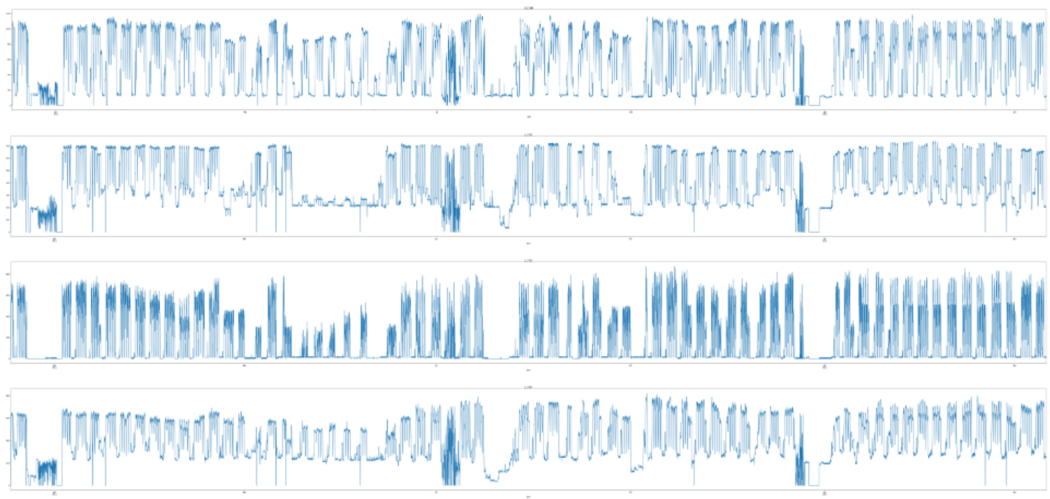


Figure 6: Example Meter Readings

Chassis building to be fitted with internal components. The dataset counts the number of vehicles that pass this particular point in the production process per hour. Combining this with the Paint Calendar gives insight into the volume of vehicles produced by the Paintshop during production and non-production hours. An additional column was added to indicate if each hour referred to a production hour or not.

Finally, the Shop Meter List Dataset provides a list of meters within each shop, broken down by ID, location and meter name. As this investigation only focuses on the Paintsthop meters, meters within this building were identified and the rest were filtered out. These meters could then be combined with the other datasets using their metername. This resulted in a final dataframe of 19921 rows of data across 5 columns.

Data is often incomplete and incoherent, therefore data cleaning aims to remove noise, fill in missing values and correct inconsistencies in the data. Some meters within the Shop Meter Dataset were added midway through the year and did not record data for the full period, therefore care was taken to handle the missing values and ensure that their readings were not removed for their period of inactivity. Once the missing dates were accounted for and readings set to zero, each meter was checked to ensure a reliable plot over the time period.

The easiest way to encode time-related information is to use dummy variables (also known as one-hot encoding, Murphy [2022]). The $pd.get\_dummies$ function was used to create dummy variables. In this investigation, the dummy variable approach was used to to capture the meter on which the observation was recorded.

4.2 Feature Selection

4.2.1 Training and Testing Dataset

The dataset was divided into three groups: training, validation and test (Tovar et al. [2020]). The training dataset is a dataset of examples used during the learning process and is used to fit the network parameters, such as the weights, and to determine the optimal combinations of variables that will generate a good forecasting model. The validation dataset is a sample of data held back from training the model used to give an estimate of model skill while tuning the models hyperparameters in order to avoid overfitting. Finally, the test set is an unseen, unbiiased data set that will be used to evaluate competing models. In this work, the training set consists of 80 percent of the whole dataset, with 20 percent reserved for validation. A completely separate dataset was made available for testing.

4.2.2 Cyclical Encoding with Sin/Cosine Transformation

There is a clear cyclical continuity present with the time variable in this investigation. Patterns can easily be identified over a daily, weekly and monthly period, where production is higher during the day on weekdays. Patterns would also be expected seasonally, whereby additional heating costs would contribute to higher energy costs in the winter. When working with energy consumption data, it is important to include information about the month of the observed consumption, as it makes sense that there is a stronger connection between two consecutive months. Using this logic, the connection between December and January and between January and February is strong. In comparison, the connection between January and July is not that strong. The same applies to other time-related information as well. Trigonometric functions can be used for this purpose, to encode the cyclical time feature into two features.

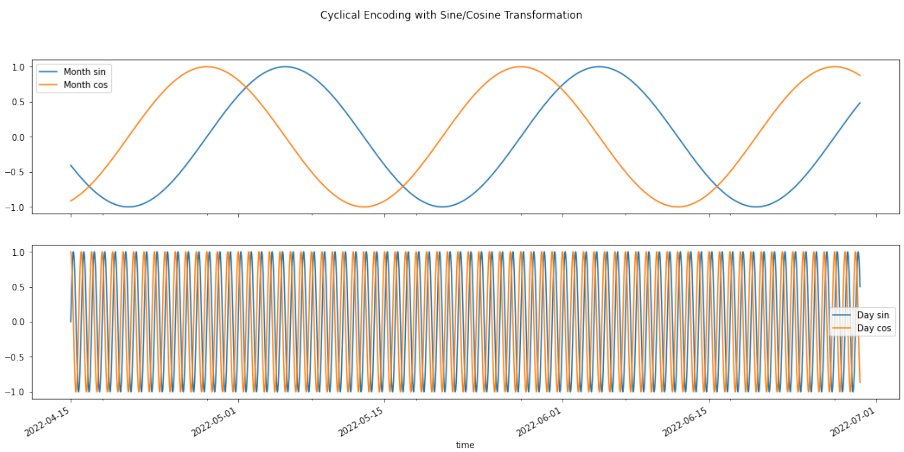


Figure 7: Monthly and Daily Sin / Cosine Cycle

The transformed data plotted in Figure 7 demonstrates why two curves must be used instead of one. Due to the repetitive nature of the curves, if you drew a straight horizontal line through the plot for a single day, you would cross the curve in two places. This would not be enough for the model to understand the observation's time point. But with the two curves, there is no such issue, and a user can identify every single time point. This is visible when we plot the values of the sine/cosine functions on a scatter plot. In Figure 8 we can see the circular pattern, with no overlapping values.

4.2.3 Data Normalisation

The data was normalised by subtracting the mean (µ) of each feature and dividing by the standard deviation (σ) to re-scale real-valued numeric attributes into a 0 to 1 range. This makes model training less sensitive to the scale of features and results in faster convergence.

4.3 Building the Model

The model was defined as being Sequential (Kumar et al. [2019]). Machine learning models that input or output data sequences are known as sequence models. Each data reading is dependent on those that come before or after it. The model was built from multiple layers. The first layers define the model and are either the RNN, LSTM or GRU layers. In this model, 3 RNN/LSTM/GRU layers are stacked on top of each other, making the model capable of learning higher-level temporal representations. In order for the model to know what shape it should expect, the first layer receives information about its input shape. In this case, an $inputshape$ argument is passed to the first layer.

The first layers return their full output sequences, but the last one only returns the last step in its output sequence, thus dropping the temporal dimension (i.e. converting the input sequence into a single vector).

In the background, the dense layers perform matrix-vector multiplication. The values used in the matrix are actually parameters that can be trained and updated with the help of back-propagation.

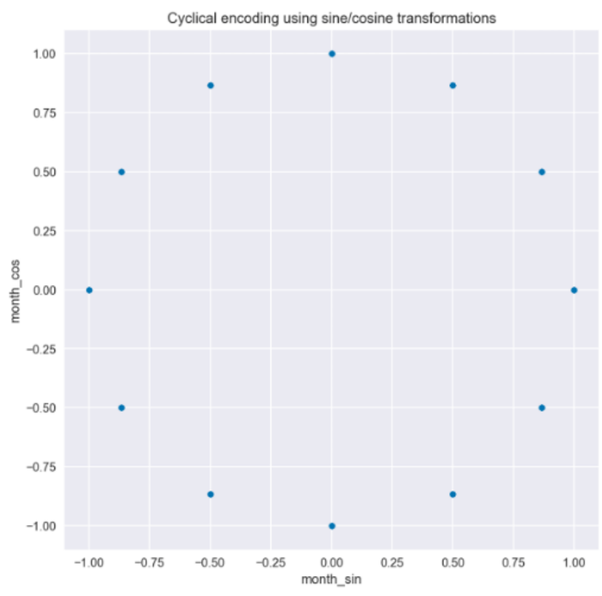


Figure 8: Daily Cos / Sin Plot

4.3.1 Activation Function

Swish is the chosen activation function, $f(x) = x \times sigmoid(\beta x)$, where β is a learnable parameter. Like the more commonly used ReLU activation function, Swish is bounded below, meaning as x approaches negative infinity, y approaches some constant value. It is unbounded above, meaning as x approaches positive infinity, y approaches infinity (see Figure 9) (Zoph and Le [2018]). Unboundedness is desirable for activation functions because it avoids a slow training time during near-zero gradients — functions like sigmoid or tanh are bounded above and below, so the network needs to be carefully initialized to stay within the limitations of these functions. However, unlike ReLU, Swish is smooth (it does not have sudden changes of motion or a vertex). Additionally, Swish is non-monotonic, meaning that there is not always a singularly and continually positive (or negative) derivative throughout the entire function. The Swish function has a negative derivative at certain points and a positive derivative at other points, instead of only a positive derivative at all points, like Softplus or Sigmoid (ICL [2018]).

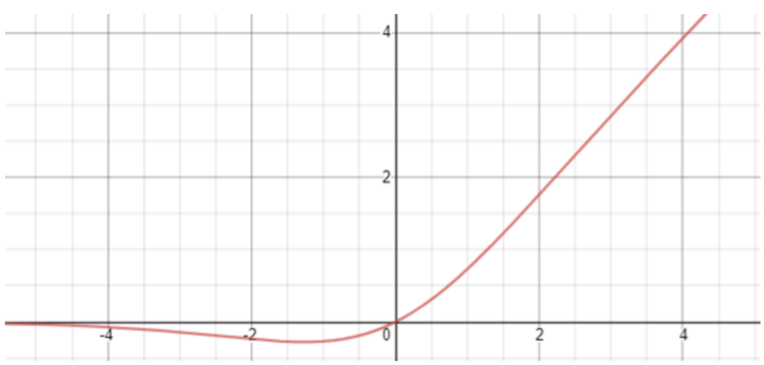


Figure 9: Swish Activation Function

4.3.2 Model Compilation

The model returns Mean Squared Error as the loss function and utilises the popular optimization technique Adaptive Movement Optimization, or Adam (Raff [2022]) as the optimizer. Adam is an update to the RMSProp optimizer (Kingma and Ba [2015]) and uses Momentum and Adaptive

Learning Rates to converge faster. The momentum algorithm accelerates stochastic gradient descent in the relevant direction, as well as dampening oscillations. A learning rate is maintained for each network weight (parameter) and separately adapted as learning progresses.

4.3.3 Output

Hourly predictions were made and compared against the actuals. It appeared that the model was fairly accurate, as the training prediction values and actual values are closely aligned (see Figure 10). In order to compare and determine the best model for this investigation, it was re-run with all of the same parameters for LSTM, GRU and RNN.

4.4 Validation and Bootstrapping

In order to validate the models, a fresh set of data was obtained for a shorter and more recent time period, 19th April 2022 - 29th June 2022. The models were run again using the unseen dataset in order to evaluate the sample, giving an unbiased estimate of model skill.

Bootstrapping was used to approximate the sampling distribution, as described by Kulesa et al. [2015]. Bootstrap resampling is a statistical methodology that relies on random sampling with replacement. It allows assigning measures of accuracy to sample estimates. It has the ability to estimate the sampling distribution of most statistical parameters (Zoubir and Iskandler [2007]). Applying bootstrap resampling in this application provides us with a sampling distribution of every element. The standard deviation of these sampling distributions is a measure of accuracy in the associated element.

These results formed the first column of recorded outputs in Figure 11 for 15 epochs and learning rate 0.001. It was unlikely that the model would be perfectly accurate on the first run, particularly as the hyperparameters were approximated. In order to tune the model, several hyperparameters were adjusted to evaluate their effect on the model.

4.5 Adjusting Hyperparameters

The learning rate was manually adjusted to understand the effect on the model. Learning rate controls the effective capacity of the model in a more complicated way than any of the other parameters; the effective capacity of the model is highest when the learning rate is correct for the optimisation problem, not when the learning rate is particularly large or small. This hyperparameter defines how quickly the network updates its parameters. Setting a higher learning rate accelerates the learning but the model may not converge (a state during training where the loss settles to within an error range around the final value), or even diverge. Conversely, a lower rate will slow down the learning drastically as steps towards the minimum of loss function will be tiny, but will allow the model to converge smoothly.

The number of epochs was also adjusted. When dealing with 3 separate models, the epoch rate was kept low (15) to allow the model to train quickly. Once confidence in the models had been achieved, the number of epochs was increased to 50. To mitigate overfitting and to increase the generalization capacity of the neural network, the model needed to be trained for an optimal number of epochs. The validation set is used to check the performance of the model after each epoch of training. Loss and accuracy on the training set as well as on the validation set are monitored to look over the epoch number to determine if the model has started overfitting.

5 Results and Evaluation

5.1 Evaluate Model Performances

The model performance was evaluated using mean squared error (MSE) and visual inspection of predicted values vs. actual values for each of the the combinations of hyperparameters, as shown in the table in Figure 11. The table shows the MSE and an example weekly profile of the performance of each model for its hyperparameters. It may be easily observed that for combinations with a higher MSE, the predicted values (blue) and actual values (orange) do not align, particularly at the extremities of the graphs which show the weekends (a more unpredictable time for production, due to

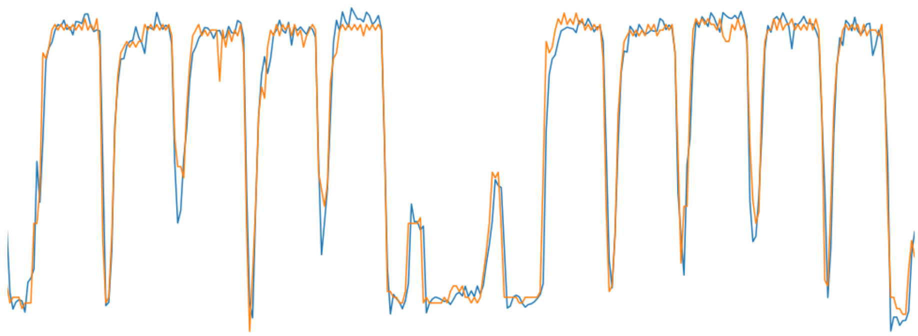


Figure 10: Training Predictions vs Actual

overtime etc.). The peaks and troughs for Monday to Friday are a lot more accurate, as would be expected, as production levels remain fairly consistent across the year for these days/nights and times. The more accurate MSE results have associated graphs that are more closely aligned, such is the case for the GRU model and higher number of epochs for all 3 models.

All three models improved with both an increase in learning rate and epoch size. The Recurrent Neural Network performed worst of all, with a MSE of 261.21. This increased slightly for Long Short Term Memory at 15 epochs, although LSTM fared much better at a higher epoch size. The best performing neural network was the Gated Recurrent Unit, which performed considerably better than the others at the highest learning rate and epoch size with a MSE of 90.71.

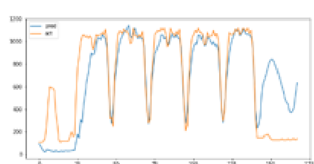
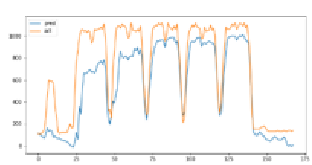
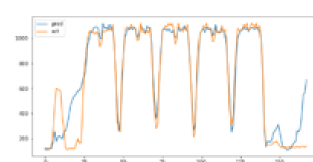
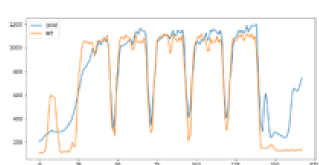
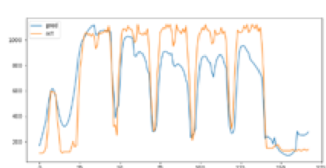
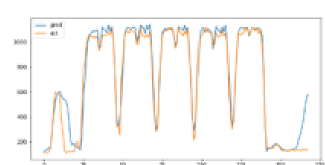
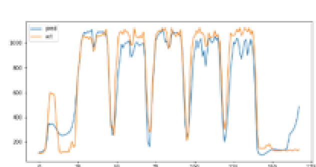
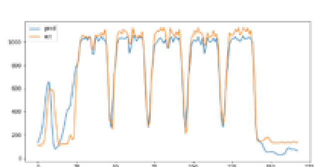
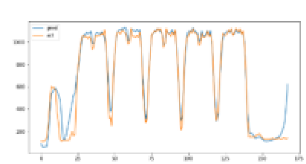
	15 Epochs, Learning Rate 0.001	15 Epochs, Learning Rate 0.005	50 Epochs, Learning Rate 0.001
RNN	261.2130536046231 	228.66681042942932 	136.85931906893344 
LSTM	172.32470411064006 	196.76870012999586 	96.89839947558565 
GRU	122.92771306275709 	109.76544016616958 	90.71524986920659 

Figure 11: Adjusted Hyperparameter Results

5.2 Novelty

Several machine learning algorithms have been studied by researchers in buildings for estimating the energy consumption, energy demand, and their related performance criteria for different circumstances (Hochreiter and Schmidhuber [1997], Mocanu et al. [2014]).

Numerous techniques are widely used to analyze energy consumption prediction problems in various types of buildings, that is, commercial, residential, or industrial sectors, however until now, this has never been carried out at a Nissan plant / for a car manufacturing plant. The component datasets that were merged together to form the final Nissan Energy Dataset have never been analysed with the objective of energy reduction in mind, nor has any sort of forecasting utilising Machine Learning techniques been carried out at Nissan.

5.3 Limitations and Further Work

Whilst the data for this investigation spanned a reasonable time period, it was during a time of unpredictability for motor manufacturing. Much of the data collected for this report was throughout the Covid 19 pandemic. The automotive industry also faced a semiconductor shortage (Kumar et al. [2019]) during this period which resulted in production shifts being cancelled and irregular weekend overtime being carried out. The investigation could likely be improved by using data from a time of "normal production".

Limitations were faced as it was frequently hard to extract the data from it's storage location. The data required for this investigation is collected by various systems and required approval from different management groups. Data retrieval also required the support of the limited data engineering resources available at Nissan to produce files in a readable format. This was challenging to justify when the project was for research and development purposes and therefore doesn't demonstrate a previously proven payback.

The study was limited to the Paintshop. Further work would include widening the scope of the investigation to include all production facilities across the plant. This would greatly increase the complexity of the model, as different shops operate on different shift and overtime patterns.

In order to improve the complexity (and possibly, accuracy) of the hyperparameters, machine learning-based adaptive window size selection method could be applied, such as those discussed by ur Rehman Baig et al. [2020] and Tomar et al. [2022]. Existing prediction methods use fixed size observation windows which cannot produce accurate results because of not being adaptively adjusted to capture local trends in the most recent data. Therefore, those methods train on large fixed sliding windows using an irrelevant large number of observations yielding to inaccurate estimations or fall for inaccuracy due to degradation of estimations with short windows on quick changing trends.

Further forcasting could be carried out to determine the baseload energy consumption and assess optimimum production levels to determine if it worth switching all equipment on and running overtime shifts unless NMUK plan to build x number of vehicles. This could be extended to include daylight hours and external temperature as additional variables.

Further work could also include use of proven methods to adjust hyperparameters, such as Bayesian optimization. This is a much more complicated approach, and is an active topic of research. In the context of hyperparameter optimization, it tries to learn a model of the loss function over hyperparameters, and use this model to adaptively choose the next hyperparameter values to try. The value of the loss function for the new hyperparameters would, in turn, be used to update the model.

6 Conclusions

The investigation was deemed to be successful, as all 3 models ran effectively and clearly demonstrated that GRU is the preferred method to forcast future energy requirements for the plant. RNN was the worst performing, which may be due to computation being slow, difficulty of accessing information from a long time ago and being unable to consider any future input for the current state. RNNs can struggle to connect previous information to the present task (Zoph and Le [2018]). LSTMs aren't known to have this problem, which is evidenced by the results, which was improved even further in the GRU model , which required less memory and was faster than LSTM, being ideally suited to this relatively small dataset.

Manual hyperparameter tuning was employed, which gave a great degree of control over the model, however proved to be a time-consuming and computation-heavy process. Although the model produced output values that were close to the actual values, it possibly could have been improved further by using automated hyperparameter tuning. If more computing power was available, methods such as Random Search (to create a grid of possible values for hyperparameters) or Basesian Optimization (to find the minimal point in the minimum number of steps) could be used instead.

If more data is collected in the future, or the dataset expands to accommodate additional features, the LSTM model may be preferred, so it would still be worth running both models in future investigations.

It was useful to have an additional test dataset available to evaluate our model again on an unseen energy dataset for the same building to get a more unbiased estimate of its predictive power. As more data is gathered, the size of this test dataset could be increased to improve confidence in the model even further.

Disadvantages of the investigation include the range of the data required from multiple sources. Fortunately, NMUK had begun collecting all of the data required to carry out this investigation in 2021, prior to the investigation being formulated. By including another year of data collection before rolling out on a plant-wide scale would ensure that the model had "seen" a normal year of production, less affected by Covid and the semiconductor shortage.

This investigation has provided the justification required to make proposals to senior management to begin collecting a variety of datasets for other forecasting projects and the possibility of storing data in the cloud for easier retrieval.

References

Óscar Trull, J. Carlos García-Díaz, and Alicia Troncoso. Application of discrete-interval moving seasonalities to spanish electricity demand forecasting during easter. *Energies*, 12, 2019. ISSN 19961073. doi: 10.3390/en12061083.

Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowlege-Based Systems*, 6, 1998. ISSN 02184885. doi: 10.1142/S0218488598000094.

Forbes. Energy update: Energy analysts forecast even larger bills this winter, 7 2022.

Nicholas Lange, C. M. Bishop, and B. D. Ripley. Neural networks for pattern recognition. *Journal of the American Statistical Association*, 92, 1997. ISSN 01621459. doi: 10.2307/2965437.

Ahmed Zayed AL Shaqsi, Kamaruzzaman Sopian, and Amr Al-Hinai. Review of energy storage services, applications, limitations, and benefits. *Energy Reports*, 6:288–306, 12 2020. ISSN 2352-4847. doi: 10.1016/J.EGYR.2020.07.028.

RAE. Counting the cost: the economic and social costs of electricity shortfalls in the uk, 2014.

J Jolly. Uk battery 'gigafactory' plans huge expansion as electric car demand soars, 10 2021.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9, 1997. ISSN 08997667. doi: 10.1162/neco.1997.9.8.1735.

Mario Tovar, Miguel Robles, and Felipe Rashid. Pv power prediction, using cnn-lstm hybrid neural network model. case of study: Temixco-morelos, méxico. *Energies*, 13, 2020. ISSN 19961073. doi: 10.3390/en13246512.

Kevin P. Murphy. *Probabilistic Machine Learning*. 2022.

Deepak Kumar, Anshuman Singh, Pijush Samui, and Rishi Kumar Jha. Forecasting monthly precipitation using sequential modelling. *Hydrological Sciences Journal*, 64, 2019. ISSN 21503435. doi: 10.1080/02626667.2019.1595624.

Barret Zoph and Quoc V Le. Swish: A self-gated activation function. *6th International Conference on Learning Representations, ICLR 2018 - Workshop Track Proceedings*, 2018.

Searching for activation functions. 2018.

Edward Raff. *Inside Deep Learning - Math, Algorithms, Models*. Manning Publications Co., 1 edition, 2022.

Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. 2015.

Anthony Kulesa, Martin Krzywinski, Paul Blainey, and Naomi Altman. Sampling distributions and the bootstrap. *Nature Methods*, 12, 2015. ISSN 1548-7091. doi: 10.1038/nmeth.3414.

Abdelhak M. Zoubir and D. Robert Iskandler. Bootstrap methods and applications. *IEEE Signal Processing Magazine*, 24, 2007. ISSN 10535888. doi: 10.1109/MSP.2007.4286560.

Elena Mocanu, Phuong H. Nguyen, Madeleine Gibescu, and Wil L. Kling. Comparison of machine learning methods for estimating energy consumption in buildings. 2014. doi: 10.1109/PMAPS.2014.6960635.

Shuja ur Rehman Baig, Waheed Iqbal, Josep Lluis Berral, and David Carrera. Adaptive sliding windows for improved estimation of data center resource utilization. *Future Generation Computer Systems*, 104, 2020. ISSN 0167739X. doi: 10.1016/j.future.2019.10.026.

Dimpal Tomar, Pradeep Tomar, Arpit Bhardwaj, and G. R. Sinha. Deep learning neural network prediction system enhanced with best window size in sliding window algorithm for predicting domestic power consumption in a residential building. *Computational Intelligence and Neuroscience*, 2022, 2022. ISSN 16875273. doi: 10.1155/2022/7216959.