Forecasting Carbon Intensity Using a Machine Learning Approach

Chris Harvey[1]

School of Computing Science, Newcastle University, UK

Abstract

Carbon intensity is defined as the number of grams of carbon dioxide (CO2) that are released to produce one kWh of electricity [1]. Renewable energy sources (solar, wind, hydro, biomass) produce almost zero CO2 emissions and therefore have a very low carbon intensity. The energy sector is responsible for 73.2% of global greenhouse gas emissions [2], switching to an energy supply with a lower carbon intensity is imperative if we are to reduce our effect on climate change. Forecasting the carbon intensity is critical for real-time energy management applications in various sectors where energy usage can be scheduled to periods where the energy is cleanest. This project will develop machine learning models to perform short-term carbon intensity forecasting. A variety of machine learning methodologies and variables will be evaluated to achieve the best test accuracies.

Keywords: Machine learning, carbon emission, energy management.

1. Introduction

1.1. Context

In recent years there has been a growing focus on reducing the emissions that we are producing, for example the UK Government has made a pledge in 2021 to reach net zero emissions by 2050 [3]. For this ambitious goal to be met drastic changes will need to be made in nearly every sector in the UK. It is vital that we meet this goal because climate change is now ‘beyond dangerous’ [4]. The energy sector will have a major role to play in this transition to a zero-emission future. Switching to cleaner sources for our energy supply is a long-term solution to reducing emissions because it will take several years to build new energy infrastructure. In the short term we can reduce emissions by better scheduling energy loads to times where the energy supply has a low carbon intensity.

[1] c.harvey2@newcastle.ac.uk

To schedule energy loads most effectively to reduce emissions we need to be able to predict the carbon intensity accurately and easily. This could be accomplished with a manual approach using analysts who study the data and make predictions, but this is costly and much slower than automated prediction methods. This problem is well suited to a machine learning approach, which would study the historic data and learn from past prediction to make more accurate prediction in the future.

1.2. Dissertation Structure

This dissertation will focus on developing multiple different machine learning models for this problem and comparing the results obtained from each. The rest of this section will clearly define the problem space with respect to this project, as well as outline the aims and objectives for this dissertation. Section 2 details the research carried out for this project. Section 3 outlines the design of the proposed solution. Section 4 discusses the implementation of the proposed design. Section 5 contains the evaluation of the finished solution, and finally Section 6 contains the final conclusions of the dissertation with respect to the aims and objectives.

1.3. Problem Space

The problem space is the problem, and everything associated with the problem. For this project the main stages of the project are data collection, model development and model evaluation.

In the data collection phase, the sources of the data will need to be identified, and it will need to be assessed whether enough data is available for this project. If enough data cannot be collected, then data augmentation techniques will need to be researched to improve training and prevent model’s overfitting the training data.

In the model development phase, the machine learning concepts to be developed will need to be chosen. There will likely be more possible concepts than what can be developed for this project, so extensive research will need to be carried out to ensure that the most promising concepts are chosen.

The last phase of the project is the model evaluation phase, this will involve testing the finished models and comparing results. For this suitable performance metrics will need to be chosen that best allow for suitable comparisons to be made. This could be a challenge due to the requirement for real-time predictions from the models. For example, a model that has an accuracy of 100% but takes 4 hours for each prediction would not produce any actionable data once the prediction has been made. For this problem to be solved, different performance metrics will need to be researched, implemented and then the importance of each metric for this application will need to be assessed.

1.4. *Aim and Objectives*

1.4.1. *Aim*

The aim of this dissertation is to develop a machine learning model capable of accurately predicting carbon intensity values 24 to 48 hours into the future.

1.4.2. *Objectives*

1. Collect the necessary data needed for carbon intensity prediction.

   For this objective to be achieved I will need to collect data from various sources, this will include historical carbon intensity data, and possibly weather data and location data.

2. Develop multiple machine learning models to predict carbon intensity values.

   For this objective to be achieved I will need to research the most promising machine learning concepts to use. I will then develop these selected machine learning concepts to produce a working solution.

3. Evaluate the machine learning models against each other, to select the best solution.

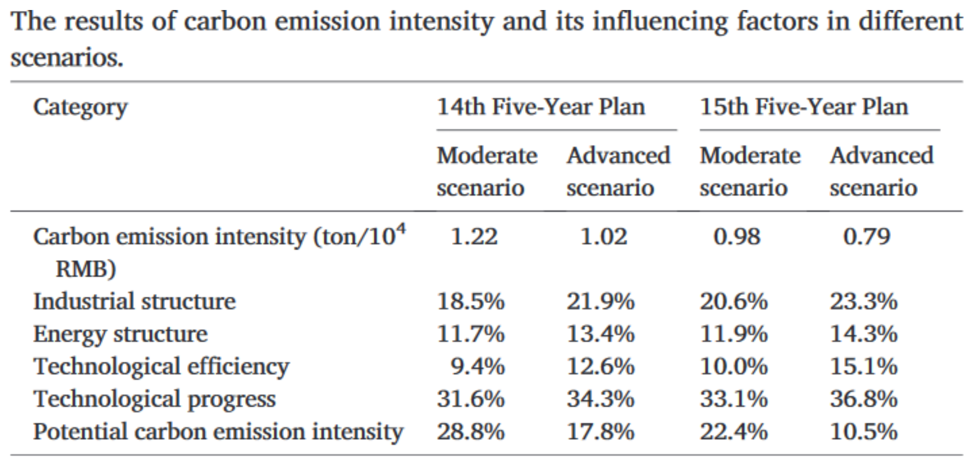
   For this objective to be achieved I will need to evaluate the machine learning models developed and decide which model is the best for accurately, efficiently, and quickly predicting carbon intensity values.

2. Research

Before proposing a new solution, research into the background of carbon intensity needed to be conducted. The aim of this was to discover any existing solutions for carbon intensity prediction and to better understand how carbon intensity data is used. This research helped to inform the design of this projects proposed solutions. Once this had been completed research into different approaches for time series prediction could be conducted. The aim of this was to discover the range of possible machine learning methods that could be used for this project and to understand which methodologies would best suit this application.

2.1. *Carbon Intensity*

The following paper focuses on the decomposition and prediction of China’s carbon emission intensity [5]. The main goal of this study was to predict long-term changes in carbon emission intensity. For this application they used one value for each five-year period. For the predictions to be made Index Decomposition Analysis (IDA) and Production-Theoretical Decomposition Analysis (PDA) were used. These techniques were applied to data from the last three five-year periods for China’s carbon emission intensity. Predictions were made based on two scenarios, the ‘moderate’ scenario involves current trends continuing and targets set by the Paris agreement are met. The 'advanced' scenario involves more efforts being made to reduce emissions than the 'moderate' scenario.



The results of carbon emission intensity and its influencing factors in different scenarios.

Category	14th Five-Year Plan		15th Five-Year Plan	
	Moderate scenario	Advanced scenario	Moderate scenario	Advanced scenario
Carbon emission intensity (ton/10^4 RMB)	1.22	1.02	0.98	0.79
Industrial structure	18.5%	21.9%	20.6%	23.3%
Energy structure	11.7%	13.4%	11.9%	14.3%
Technological efficiency	9.4%	12.6%	10.0%	15.1%
Technological progress	31.6%	34.3%	33.1%	36.8%
Potential carbon emission intensity	28.8%	17.8%	22.4%	10.5%

Figure 1 Carbon emission intensity predictions for moderate and advanced scenarios [5]

This paper predicted carbon emission intensity using factor analysis and an improved extreme learning machine [6]. Firstly, this involved stochastic frontier analysis to find the factors influencing carbon intensity. A model was then created to predict carbon intensity using the factor analysis and an extreme learning machine. An extreme learning machine is a type of a feed-forward neural network. The following paper compares ELMs to Convolutional Neural Networks (CNNs) [7]. They discovered that the accuracies obtained from each model are on par with each other, but the ELM requires less than 2% of the training required for the CNN. In [6] a Whale Optimisation Algorithm is used to improve the ELM. The resulting model has the best Mean Absolute Percentage Error (MAPE) and Root Mean Square Error (RMSE) of any of the six different models being compared.

The following paper proposed a Day-ahead Carbon Intensity Forecasting (DACF) system which used machine learning to make predictions [8]. The prediction method being used involves two stages. The first stage involves forecasting the energy production for all the energy generating sources. The second stage then combines the production forecast with the carbon-emission rate for each source to generate a carbon intensity forecast. The DACF was trained on data from a three-year period (2019-2021) and the models were re-trained every six months. The machine learning model used for prediction was an Artificial Neural Network (ANN) which was developed using Python and Keras [9].

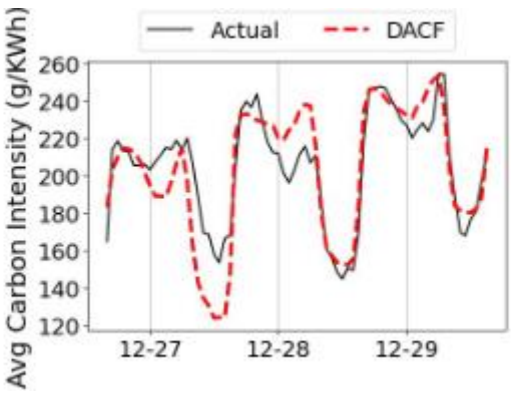


Figure 2 Carbon intensity predictions made by Day-ahead Carbon Intensity Forecasting system (DACF) [8]

The National Grid's Carbon Intensity API [10], is the main source of data for this project. They provide carbon intensity data for the United Kingdom, but also provide carbon intensity forecasts for each 30-minute period several days into the future. These forecasts are updated every 30-minutes and are made utilising an ensemble of state-of-the-art supervised machine learning regression models. There isn't any information on what machine learning models are being used for these predictions. This makes designing a model to improve on these predictions very challenging.

2.1.1. Summary of existing solutions

Paper	**Problem**	**Methodology**	**Results**	**Comments**
[5]	Long-term carbon intensity prediction.	IDA & PDA	Predictions of next five-year period for two scenarios.	ML approach not used so this paper applies minimally to the aims of this project.
[6]	Long-term carbon intensity prediction.	ELM	Impressive accuracies obtained by ELM.	ELM could be a possibility for this project.
[8]	Short-term carbon intensity prediction.	ANN	Accurate short-term carbon intensity predictions.	DACF prediction model could be adapted to fit the aim and objectives of this project.

[10]	Short-term carbon intensity prediction.	Unknown.	Accuracies are not published.	Lack of information on prediction techniques will make comparison difficult.

2.1.2. Comparison of existing solutions

The first paper [5] focused on predicting long-term changes in carbon intensity for China. These predictions were made using analysis techniques, which does not transfer well to making short-term predictions of carbon intensity in real-time.

The second paper [6] reviewed also focused on predicting carbon intensity values for China. However, this paper leveraged machine learning techniques to make the predictions, they made use of an ELM. The models were trained on 12 years of data and used to make long-term predictions for numerous provinces in China. The ELM compared very well to the other machine learning models for this application. It is unknown if this would work just as well for short-term predictions, but this is a very promising model that can be used for this project.

The third paper [8] used an ANN to make short-term carbon intensity predictions for six regions across the US and Europe. This paper unlike the previous two reviewed focused on short-term carbon intensity prediction which makes it very relevant to this project. The results from the ANN shown in Figure 2 demonstrates that the model can successfully and accurately predict short-term trends in carbon intensity values.

The fourth source reviewed [10] was the National Grid's Carbon Intensity API. This resource published carbon intensity data along with predictions several days into the future. Due to the machine learning methods used being undisclosed it is difficult to theorise how to make improvements on the predictions that they are making.

From this literature review the most promising technology is the ELM which could be used for this project as it has not been used for short-term carbon intensity predictions previously.

2.2. *Machine Learning Time Series Forecasting*

The remainder of this section will contain the research into several machine learning model concepts that can be used for time-series forecasting.

2.2.1. Long Short-Term Memory

The machine learning model Long Short-Term Memory (LSTM) was proposed in 1997 by [11]. This model is a type of Recurrent Neural Network (RNN) which is capable of learning order dependence between predictions. RNNs make use of context for each prediction, but the context must be learned by the model. LSTMs work well for time series forecasting because the context of previous predictions is stored by the model and used to inform future predictions.

2.2.2. Multilayer Perceptron

The Multilayer Perceptron (MLP) is a fully connected type of feedforward ANN [12]. MLPs consist of a system of interconnected nodes, these nodes are connected by weights and output signals which allows the model to approximate extremely non-linear functions [13]. MLPs form the basis for all neural networks and are the simplest type of machine learning model which allows them to be very general purpose. MLPs can be used for time-series forecasting, but other more complex models will likely provide better results than the MLP.

2.2.3. Convolutional Neural Network

Convolutional Neural Networks (CNNs) are comprised of three types of layers, which are convolutional layers, pooling layers, and fully connected layers [14]. Convolutional layers focus on the use of learnable kernels, which the input data is split into. CNNs are typically used on image data, but they can be used for time-series forecasting with some modification to the input data.

2.2.4. Deep Learning

Deep learning (DL) models contain multiple processing layers, which allow them to learn representations of the data with multiple abstractions [15]. DL allows for models to recognise complex patterns in the data, but the deep nature of the models can significantly increase training time. DL has been used for a wide range of applications and can be used for time-series forecasting with some data modification before it is fed into the model.

2.2.5. Autoregressive Model

An Autoregressive Model (AR) is a model predicts the value of a variable based on its past values. An AR model uses a set of lagged values of the dependent variable as predictors [16]. AR models are simple and effective at modelling linear relationships in time-series data. These types of models are widely used for time-series forecasting in areas such as stock market predictions and climate forecasting. Auto Regressive Integrated Moving Average (ARIMA) is a widely used class of AR models.

2.2.6. Extreme Learning Machine

Extreme Learning Machines (ELMs) are feedforward neural networks which provide good generalisation performance at extremely fast learning speeds [17]. Unlike standard neural networks ELMs do not require gradient-based backpropagation to work and instead uses Moore-Penrose generalised inverse to set its weights. ELMs can achieve accuracies on par with CNNs in less than 2% of the training time [7]. Due to the impressive results for carbon intensity prediction by [6] using an ELM this machine learning model shows promise over the other options.

2.2.7. *Comparison of Machine Learning Models*

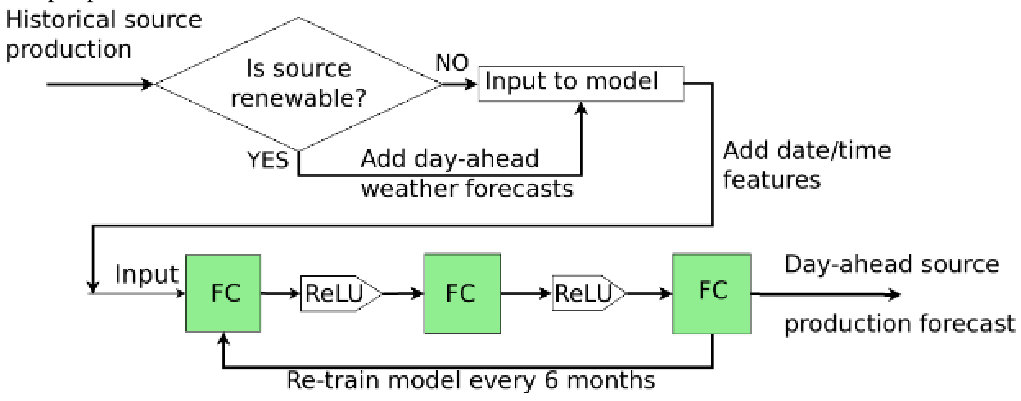
The LSTM and the AR models are the only two models which incorporate past predictions and values from the training data to make predictions. This will fit the application of time series forecasting well due to sequential predictions being linked to each other in some way.

Alternatively, the MLP, CNN, DL and ELM models attempt to learn the patterns in the data by training their weights, and without any memory of previous values. Sequential predictions by these models are not linked in any way, so these models might perform worse at predicting long-term trends in the data.

3. Design

3.1. *Proposed Solutions*

This section will propose the most promising machine learning models based on the research conducted into the models in the previous section. The proposed system will work in a similar way to the DACF proposed by [8]. The model will use historical carbon intensity data from [10] as well as weather forecast data. Some machine learning model will then take this data and predict carbon intensity for at least the next 24 hours. Longer-term predictions may be possible depending on the accuracies obtained from the proposed solution.


Historical source production
Is source renewable?
NO
Input to model
YES
Add day-ahead weather forecasts
Add date/time features
Input
FC
ReLU
FC
ReLU
FC
Day-ahead source production forecast
Re-train model every 6 months
Figure 3 ANN model design and structure for DACF [8]

Above is the model design of the ANN used by the DACF. The machine learning model used for this is application is a very simple fully connected feed-forward neural network. The model structure is composed of three fully connected layer and two Rectified Linear Unit (ReLU) layers. The ReLU activation function will output the value from the previous layer if it is positive and will output zero if it is negative. This activation function is used commonly with DL models and allows for DL models to account for non-linearity in the data.

Of the existing solutions reviewed, the DACF proposed by [8] most closely matched the aims and objectives of this project. This is why he proposed solution of this project will be very similar to the ANN model design shown in Figure 3. However, a different machine learning model will need to be used to allow for new findings to be made. This means that the MLP researched in 2.2.2 is no longer an option. Of the remaining options the CNN and DL which are most used for image data might be overkill for this application. This is because they are complex models used to find complex patterns in the data and the carbon intensity data used for this project is simple by comparison. AR models are a type of classical model which does not necessarily need to make use of machine learning for prediction. Since this model does not use machine learning it will be discounted because the aim of this project is to develop a machine learning model for prediction.

The two remaining models include the LSTM and the ELM which are the most promising machine learning models for this application. The LSTM is more commonly used for time-series forecasting, but with some modification the ELM will work as well. Below is the proposed design for both chosen models.

3.2. *Long Short-Term Memory Model Design*

An LSTM is composed of a cell, an input gate, an output gate and a forget gate [18]. The cell will remember values over an arbitrary length of time, and the three gates regulate the flow of information for each cell. An LSTM is constructed with many of these cells which form a chain.

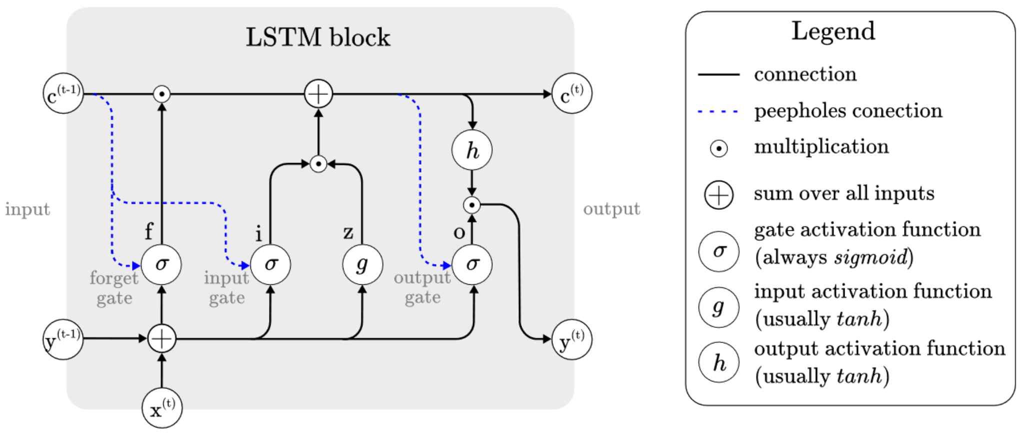


Figure 4 Architecture of a typical LSTM block [18]

The connection at the top of Figure 4 which runs from $c^{(t-1)}$ to $c^{(t)}$ represents the chain of cells. The first step in the block is to decide what information to discard from the cell state, this decision is made by the forget gate layer. The next step is to decide what new information should be stored in the cell state, this decision is made by the input gate layer. Finally, the block needs to decide what to output to the next block in the chain. This is accomplished by the output gate layer. Figure 4 shows a modified version of an LSTM which makes use of peephole connections, this modification was proposed by [19].

LSTM models are commonly used for time-series prediction, therefore developing an LSTM for carbon intensity forecasting should be straightforward. The design of the

LSTM will allow for important recent readings to be stored in the model and referenced for future predictions. This is important for time-series data as each value is dependent on the previous readings.

3.3. *Extreme Learning Machine Model Design*

ELMs are Single-Hidden Layer Feedforward Neural Networks (SLFNs) which have proven very capable at fast learning. ELMs are a classical one hidden layer neural network but without a learning process. If an ELM contains enough hidden neurons and has enough training data, it can make predictions accurately with much less training time than other classical models. ELMs do not perform iterative tuning because the model does not incorporate any backpropagation method.

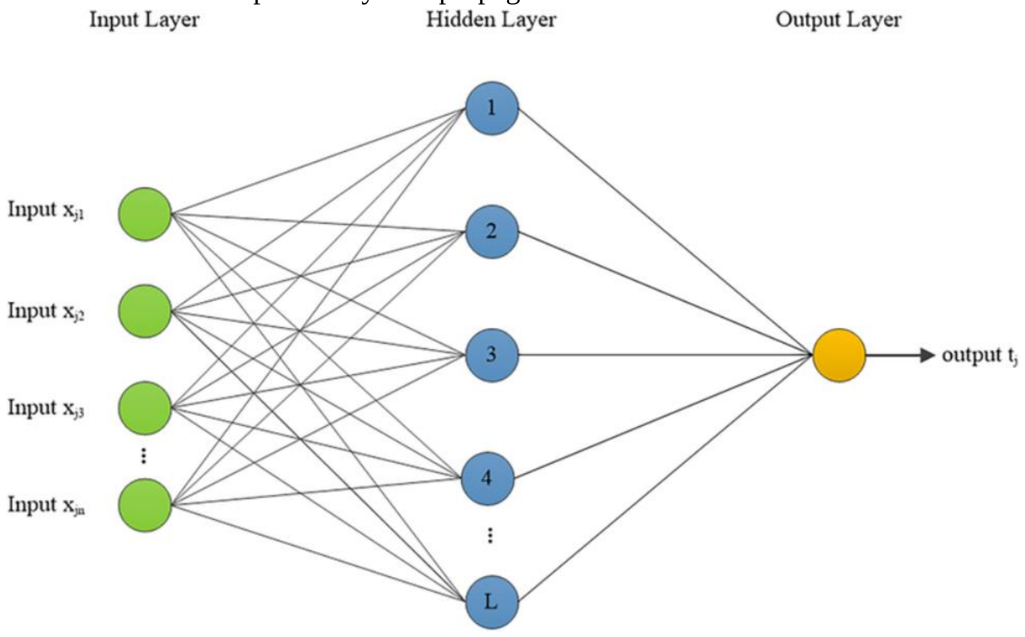


Figure 5 Diagram of Single-Hidden Layer Feedforward Neural Networks [20]

Figure 5 shows the basic concept of any SLFN, depending on how the model is trained it can be an ELM as well as a SLFN. The model only consists of three layers which are the input layer, the hidden layer, and the output layer. The input layer does not perform any computation on the data, and the output layer does not perform any transformation on the data. The hidden layer is the “single” layer referred to in the definition of SLFN.

To train this model as an ELM first the input layer weights, and bias are set randomly and never adjusted. The weights determine how proportionally important a particular input is, and the bias determines which hidden layer nodes are given priority over others. In an ELM the output weights are independent of the input weights which is the opposite of a machine learning model trained using the backpropagation method. The hidden nodes are also assigned random weights, these weights might not be changed, but in most cases the weights are learnt in a single training step.

ELMs are a relatively new concept in machine learning, so they have seen little use for time-series forecasting so far. Their main advantage is the very fast training times, which makes them ideal for big data applications. The carbon intensity data used by this project will be from the past few years, but this will still be a relatively small amount of data. Because of this train times are not as important as the accuracies obtained. Investigating the advantages and disadvantages of this approach when train times are not as important will be the focus of implementing this model.

4. Implementation

This section will document the implementation of the LSTM and ELM machine learning models.

4.1. Chosen Technologies

Before the implementation could begin the technologies that this project would utilise needed to be chosen. For the development platform I decided to use Google Colab [21]. Colab notebooks are Jupyter Python notebooks [22] which are hosted by Google. These notebooks are hosted by Google for the express purpose of machine learning applications. Specialised hardware for these applications including Graphics Processing Unit's (GPU's) and Tensor Processing Unit's (TPU's) is available as well.

For the machine learning library, I chose to use Keras [9] which is a Python API for Tensorflow that was developed by Google. This was a library I had previous experience using in the past, and it is built-in to Google Colab so use of Keras in Colab is seamless.

4.2. Data Collection

Before beginning data collection, I needed to determine what data to collect. I decided to start with a univariate approach, so only using the carbon intensity data to predict future carbon intensity. If the results obtained from the univariate approach were not sufficient, then other sources of data can be explored.

The carbon intensity data used for this project is from the National Grid's Carbon Intensity API [10]. This API offers several different types of data, this includes carbon intensity by UK region as well as data on the sources of the current energy production. One of the datasets offered is the national carbon intensity value every half hour which dates back a few years and is the data used by this project. This data also comes with a carbon intensity prediction which was made at the time and will be useful for comparing the accuracy of my models.

The first complication I encountered with collecting the data was that the Carbon Intensity API [10] only allows for 31 days' worth of data to be downloaded at once. To solve this issue, I created a batch file which ran a command to download the data for each month from 2018-2022. This gave me a total of 84,664 carbon intensity values to use for training, testing, and validating the models.

4.3. Data preparation

Once the data had been collected it needed to be prepared into a format that could be used by a machine learning model. Originally the data was in 60 separate json files as each file was for one month of data from the five-year period. I wrote some Python code which read each of the 60 files in chronological order and appended the data to a Pandas DataFrame [23]. I chose to use a Pandas DataFrame because it is widely used for data science applications and includes a wide range of tools for data modification. This DataFrame was then exported to a csv file which can be loaded for running the model.

The data originally came with a ‘from’ variable and a ‘to’ variable which denoted the time-period from which the data was recorded. I decided to reduce this to one variable which denotes the midpoint of the time-period. This reduces the data that needs to be stored and loaded to run the models. This will also make displaying plots easier as the midpoint can be used for each datapoint, without it needing to be calculated in the future.

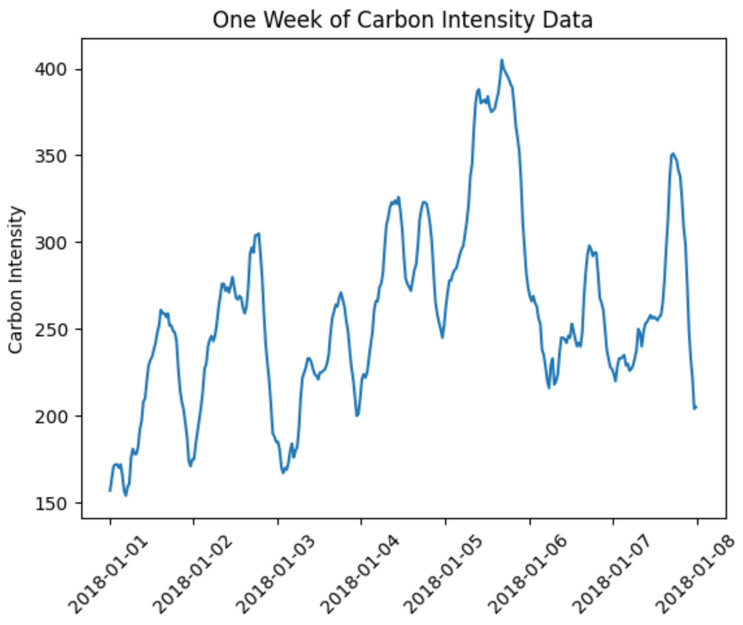


Figure 6 Carbon intensity data from one week.

Error! Reference source not found. displays the carbon intensity data from one week in 2018. From this plot the general short-term trends in the data can be seen. From **Error! Reference source not found.** it is clear to see that the carbon intensity is decreasing for much of the morning, it then reaches a minimum and then proceeds to climb for the rest of the day until the late evening when it begins to decrease again. The volatility of this data can clearly be seen as well, for example the minimum of some days is almost above the maximum from other days within this week of data. This may make prediction challenging as this volatility is not exclusive to this portion of the dataset.

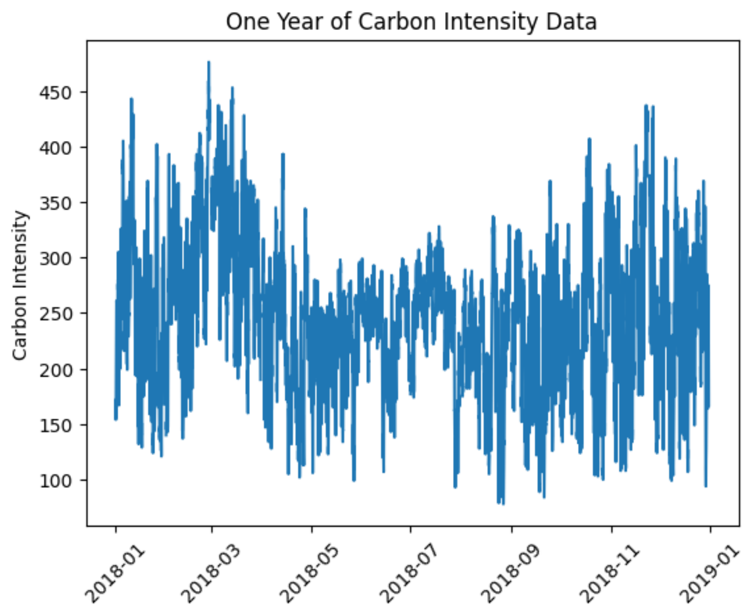


Figure 7 Carbon intensity data from one year.

Error! Reference source not found. shows the carbon intensity values for the year of 2018, and from this plot the long-term seasonal trends in the data can be seen. Generally, the carbon intensity in the spring and summer months is lower than the carbon intensity in the autumn and winter months. From this seasonality in the dataset, either multiple years of data will need to be used for training to allow for seasonality to be learned, or alternatively only recent data could be used for training which would eliminate most of the seasonality.

The next step in data preparation was to remove any unnecessary data not needed by the machine learning model. For this only the carbon intensity column, and the datetime column were required, so this was selected from the dataset.

To account for missing values in the dataset, I created a new empty DataFrame which used the range of dates from the dataset to determine the length of the DataFrame. This DataFrame was longer than the original which indicated that missing values existed in the original DataFrame. These two DataFrames were then merged using an outer join. This yielded a DataFrame where the datetime column was complete, but the carbon intensity column contained several missing values. Pandas interpolate feature was used to correct for these missing values. Because the data was uniformly separated linear interpolation can be used, and this managed to interpolate the data very well.

I then decided to normalise the carbon intensity values to between zero and one. This is common for machine learning applications where the approximate upper and lower bounds are known, few or no outliers exist in the data, and the data is uniformly distributed across that range [24]. Upon implementing the data normalisation, I discovered the normalised values were very close to zero for almost all the data. This means that outliers are present in the data, and this results in a larger upper bound for the data than what the upper bound should be.

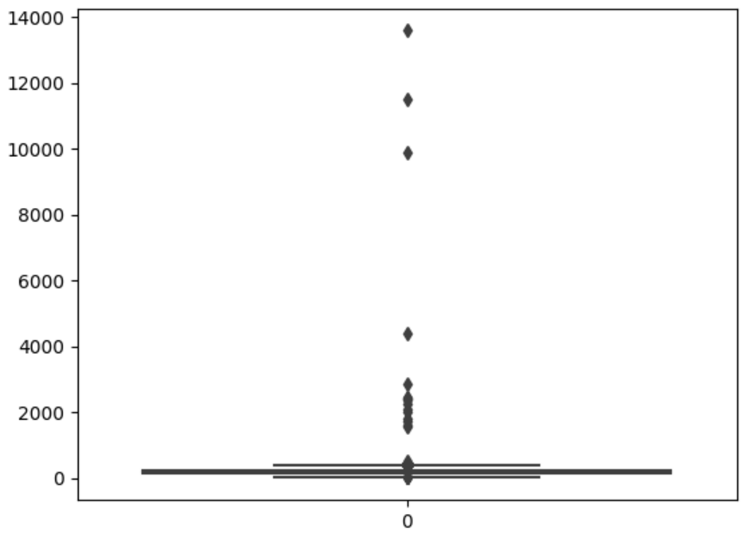


Figure 8 Box plot of carbon intensity data.

Error! Reference source not found. shows a box plot of the carbon intensity data, from this box plot it is clear to see there are several outliers which are much larger than the values in the rest of the dataset. There is a high chance that these values are anomalous due to how extreme they are, therefore, these values should be removed from the dataset.

To accomplish this, I extracted the outlier information from this box plot which was produced using the Python library Seaborn. This gave me the information that only a few outliers existed below the minimum, and that the lowest outlier above the maximum was 387. In total 595 outliers were found which represents 0.7% of the data. I was apprehensive about modifying 0.7% of the data in the dataset so I instead decided to only modify the values that were highly likely to be anomalous. These anomalous values include the values lower than the minimum as well as all values greater than 500. This now totalled 22 outliers which represented only 0.03% of the data. To fix these anomalies and maintain the completeness of the data I decided to replace each anomalous value with the average of the first non-anomalous value which was recorded before and after it.

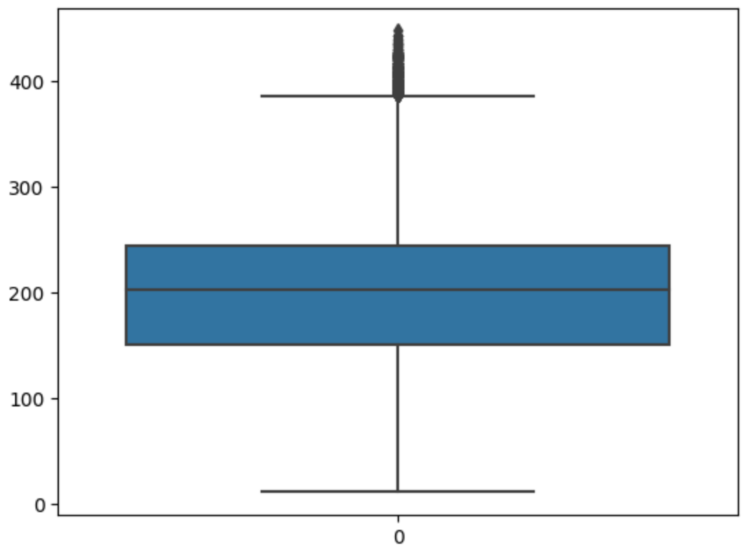


Figure 9 Box plot of carbon intensity data with outliers corrected.

Error! Reference source not found. shows the carbon intensity data with outliers corrected, as can be seen the outliers that have been found by the box plot are not as extreme as they were before. The outliers that have been found are most likely extreme readings and not anomalies because there are more than a hundred values that are outside the maximum of this box plot, and these are all in very close proximity to each other.

The next step in the data preparation phase was to produce features and labels from the dataset which are necessary for training. A feature in time series data is a series of values, and the label is the value that immediately follows this series. The model is trained with these features and asked to predict the label for each set of features which is how the models' weights are improved during training. The feature length is a parameter which can be tuned to improve the performance of the model, this is because the more data the model has to make each prediction the more accurate that prediction should be. But increasing the feature length will lead to diminishing returns, and a larger feature length will lead to less total features so there is a sweet spot for this value that needs to be found.

4.4. Long Short-Term Memory Implementation

For the development of the LSTM my plan was to develop a very simple LSTM that could be tested on a small subset of the data to inform any necessary changes that would need to be made to improve results. This approach would allow me to test changes very quickly, and rapidly prototype different versions of the LSTM.

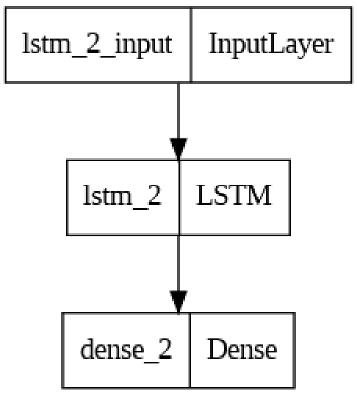


Figure 10 Layers of first LSTM.

Figure 10 shows the layers of the first LSTM model that was developed. This model only contained one LSTM layer with units of 50 and a dense decision layer with units of one. This model was trained on only one month of data, and I initially decided to use a feature length of 100 which would hopefully provide the model with ample data for each prediction. This model was trained for 100 epochs which each only took two seconds because of the simplicity of this model. The model was trained on the carbon intensity data from the month of January in 2020. To evaluate the model, it was asked to predict the carbon intensity for the first day in February 2020 and this was compared with the actual values.

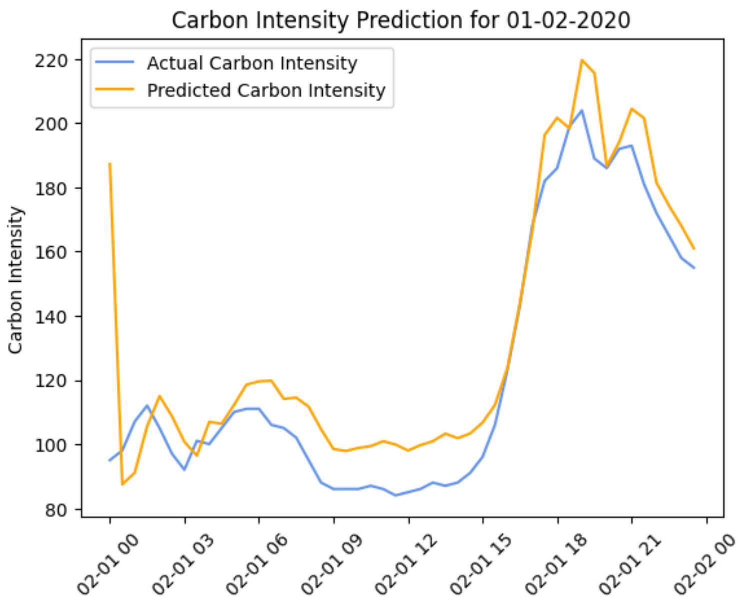


Figure 11 LSTM next-step carbon intensity prediction for 01-02-2020

As can be seen in **Error! Reference source not found.** the model is capable of accurate next-step prediction. The predictions represent 24 hours ahead of any training data the model was trained on. However, for these predictions the real data was used to compose the features for each prediction. This means that each prediction was made with the reading 30 minutes before it. The next step was to test this model for multi-step prediction, which will allow for predictions to be made more than 30 minutes in advance. To accomplish this carbon intensity values are predicted one at a time, and the previous predictions are used to generate the feature for the next prediction. This will likely not be as accurate because the uncertainty will grow over time rather than it being reset every 30 minutes like it was in **Error! Reference source not found.Error! Reference source not found.**.

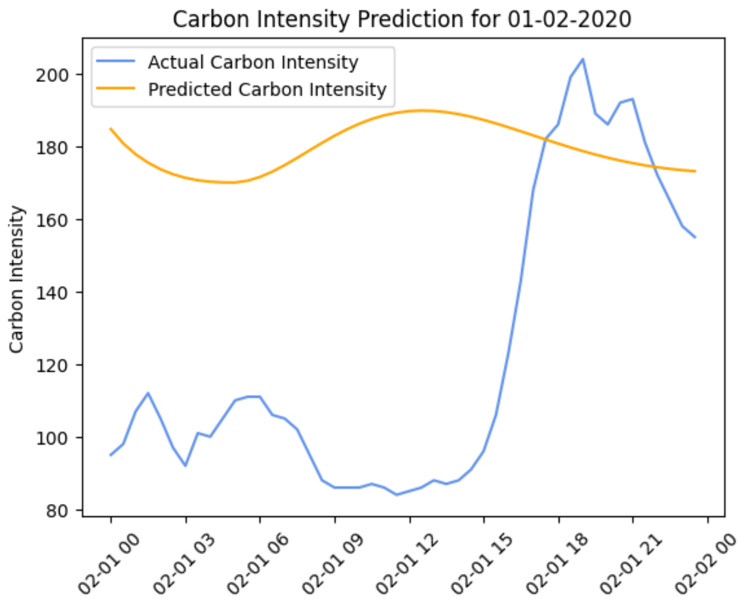


Figure 12 LSTM multi-step carbon intensity prediction for 01-02-2020.

Error! Reference source not found. shows the predicted carbon intensity values from the LSTM when multi-step prediction was used. This data is drastically different from the next-step predictions from **Error! Reference source not found.**. As can be seen the model is attempting to predict the daily changes in carbon intensity, but the predicted values are far from the actual values. The predicted data also appears much smoother than actual data, so the model has not managed to learn the volatility of the carbon intensity data. This is most likely due to a lack of training data, or training time, but a lack of training data is the more likely problem here as only one month was used for training.

To test whether this was the issue, the entire five-year dataset was used for training, but similar results were obtained. This showed that a significant change in the machine learning pipeline was needed to obtain accurate multi-step predictions. The first step in improving the model performance was to conduct further data exploration. For this the steps from 4.3 to remove outliers and complete the dataset were implemented. This again yielded the same results as before, so other factors would now need to be investigated to improve results.

The next step I took was to analyse the distribution of the data, I plotted this distribution using the Python library Seaborn and found that the data was not normally distributed. LSTM models favour data that is normally distributed so correcting the distribution should lead to improvements in results. For this I decided to perform a Box-Cox transformation [25] on the carbon intensity data. A Box-Cox transformation is a commonly used method for transforming non-normally distributed data to more normally distributed data.

$$\begin{cases} y = \dfrac{x^{\lambda} - 1}{\lambda} \text{ where } \lambda \neq 0 \\ y = \ln x \text{ where } \lambda = 0 \end{cases}$$

Figure 13 Box-Cox transformation formula [25]

This involves finding the most optimal value for λ which produces the most normally distributed data possible. The new normally distributed data is represented by y, and the starting data by x. I implemented this transformation and found the most optimal λ value was 0.8012. This produced a data distribution which was more normally distributed but not by a large amount when compared with the original distribution.

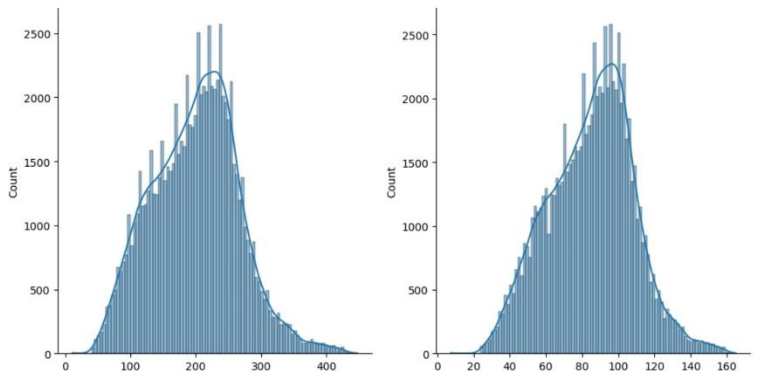


Figure 14 Original plot of carbon intensity distribution (left) and after Box-Cox transformation (right)

I discovered from research of similar solutions online that my original choice of feature length which was 100 was likely too large. I changed this to be a feature length of 20. I also decided to use all the five years’ worth of data collected. I split this data, and used 70% for training, 20% for testing and 10% for validation. The testing data was the most recent 20% of the data. The training and validation data was comprised of the other 80% of the dataset, but this was scrambled to ensure that the training and validation sets contain the highest range of data possible.

For the final LSTM model, I also developed a new method for generating the features and labels. Previously I was using the readings directly before each label to form the feature. In the new method I am treating the data from each hour as independent from the data of other hours. For example, the feature for a label which was at 14:00 would be the previous 20 readings from 14:00.

This approach has several benefits, which include that the model does not need to learn the daily trend in the data, so less training data should be required to reach the maximum accuracy. Predicting the next 24 hours' worth of readings can be done with next step prediction rather than multi-step prediction which helps to reduce the complexity of the problem. Whereas with the previous methodology previous predictions would be required to form the next label, so the accuracy of predictions will decrease dramatically as the proportion of real data in the label's decreases. Treating each hour as independent allows for the hours of each day being predicted to have the same uncertainty, this will increase the usefulness of the forecasts being made. This improved model produced much better results which will be discussed in the evaluation section.

4.5. Extreme Learning Machine Implementation

The implementation of the ELM was much simpler than the LSTM as the machine learning pipeline had already been developed. For the ELM training I decided to make use of all the five years' worth of data and to use a univariate approach because it had worked well for the LSTM. Since the inputs and outputs of the ELM will be the same as the LSTM, the model was the main thing that needed to be changed.

The ELM has some main differences to the LSTM when it comes to implementation, one is that the ELM is only trained for one epoch. Because of this I decided not to use a validation dataset as this would provide minimal benefit to performance. Instead, the data was used to create a larger training dataset which should provide more benefit than a validation dataset. The model only requiring one epoch proved to be as much of a problem as it was a benefit, this was because any plots created from the training metrics for the LSTM were not valid for the ELM. Therefore, it was difficult to work out what in the model needed to be changed to improve results.

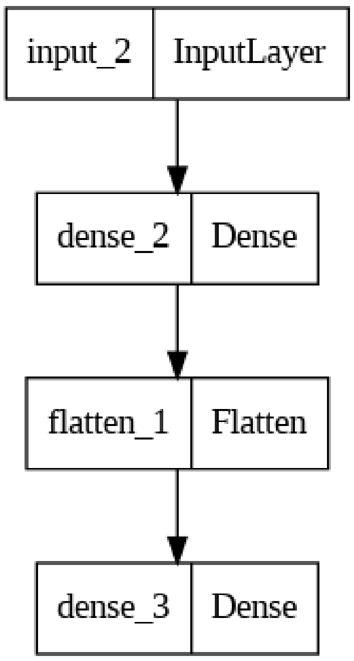


Figure 15 Layers of first ELM

Figure 15 shows the layers of the first ELM I developed; this model contained only the layers that were absolutely required. The first dense layer was the most important because this was the layer that makes ELMs different from any other model. This layer contained 100 nodes but was not trainable and was initialised with random weights for each of the nodes. After this the output needed to be flattened before the final dense decision layer, which contained 24 nodes; one for each hour data being predicted.

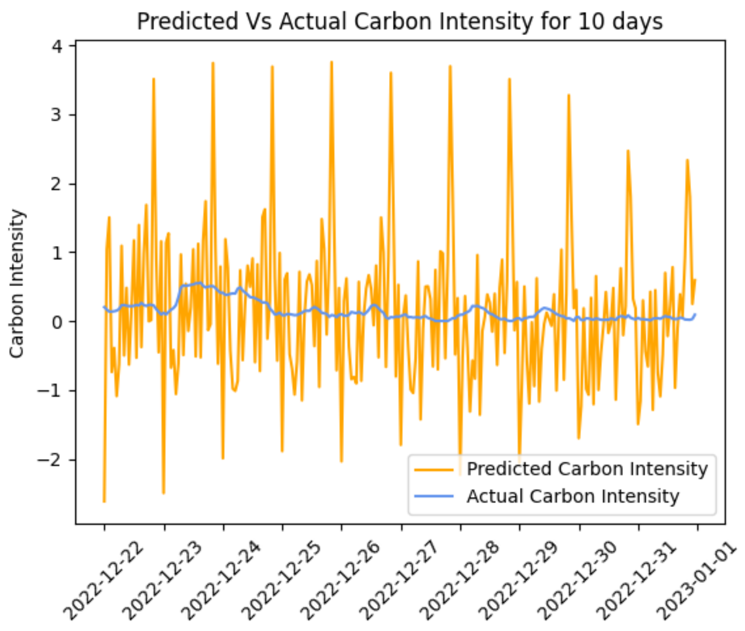


Figure 16 ELM next-step prediction for 10 days

Figure 16 shows the predicted carbon intensities from the ELM shown in Figure 15. The model is not able to accurately predict the carbon intensity values and predicts negative values which are not found anywhere in the training dataset. The model does understand the 24-hour cycle in the data with a large peak being predicted once per day, but the scale of the predicted data is completely wrong.

From this result I experimented with different model parameters to attempt to improve results. I experimented with changing the number of nodes in the ELM layer but found that 100 nodes was the optimal amount. Next, I experimented with adding layers after the main ELM layer to improve training. I found that two dense layers with the ReLU activation function helped to improve the predicted values. Loss functions were also compared, but the mean absolute error loss function used in my LSTM was found to be the best option. These improvements yielded predictions that were closer to the actual data, but still a long way off when compared to the LSTM. I suspect that to achieve better results a much larger amount of training data would be required. This is because ELMs train on only one epoch which allows them to train on very large datasets very quickly, but very large datasets may also be required to achieve meaningful results.

5. Evaluation

This section will document the evaluation of the LSTM and ELM machine learning models.

5.1. *Long Short-Term Memory Evaluation*

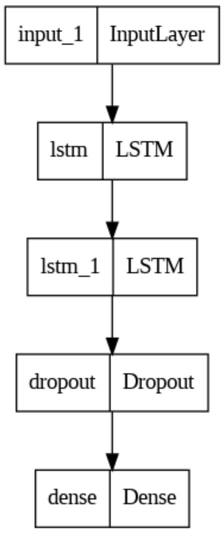


Figure 17 Layers of final LSTM

Figure 17 shows the layers of the final version of the LSTM developed for this project. This model contained two LSTM layers, which is one more than the original LSTM shown in Figure 10. The first of these layers contains 48 nodes, and the second

24 nodes. A dropout layer is then used as I discovered that the model was overfitting, so a dropout value of 0.3 was chosen after testing other values.

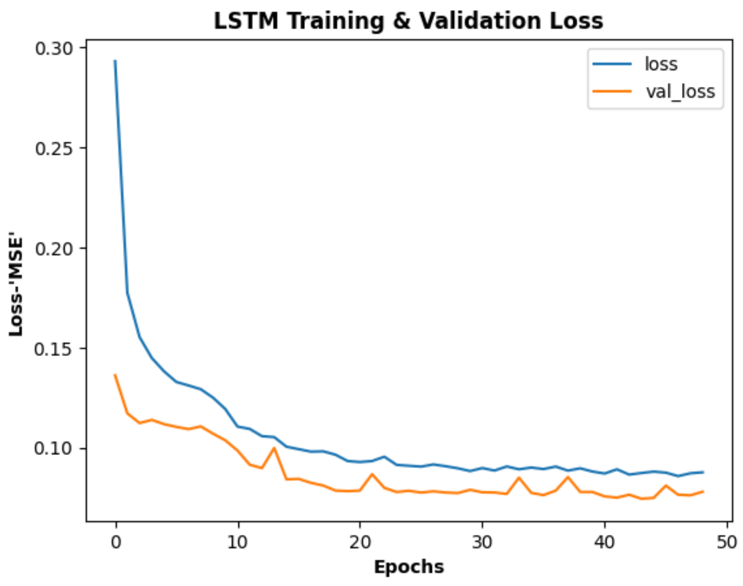


Figure 18 LSTM training losses

As can be seen in Figure 18 the model stops training after 49 epochs. This is because early stopping was used to stop the model from overtraining. The validation loss metric was monitored by early stopping and a patience of five was chosen. This ensured that overtraining was kept to a minimum, but also that training did not stop early if there was a couple of bad epochs in a row.

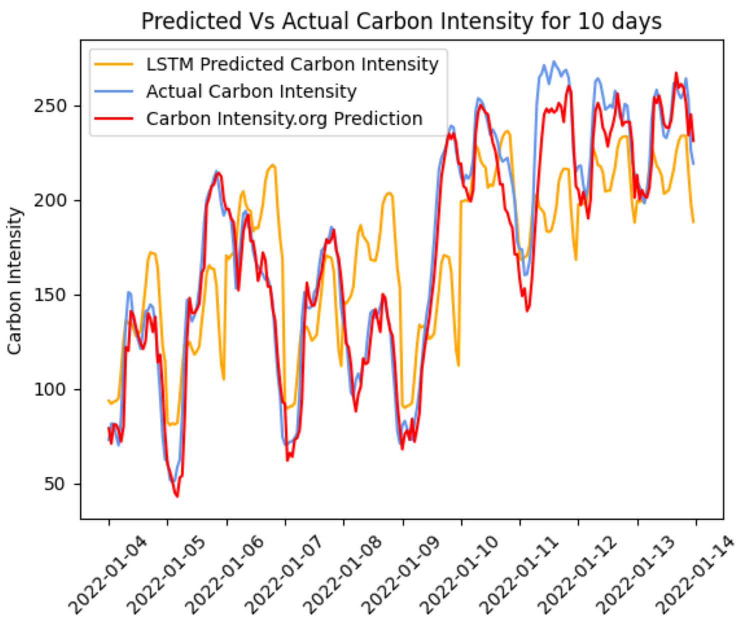


Figure 19 LSTM next-step prediction for 10 days

Figure 19 compares the LSTM to the prediction from the Carbon Intensity API [10] and the actual data. As can be seen the LSTM does replicate the data quite well, but not nearly as well as the prediction from the Carbon Intensity API. The LSTM makes predictions 24 hours in advance, the Carbon Intensity API makes predictions multiple days into the future. However, it is not known how recent the predictions are to the value they are predicting. For example, all predictions might be updated every 30 minutes, so the predicted values available on the Carbon Intensity API may only be 30 minutes ahead which would provide an advantage over my LSTM. Since almost no information is available on how carbon intensity is predicted by the Carbon Intensity API comparisons are very difficult.

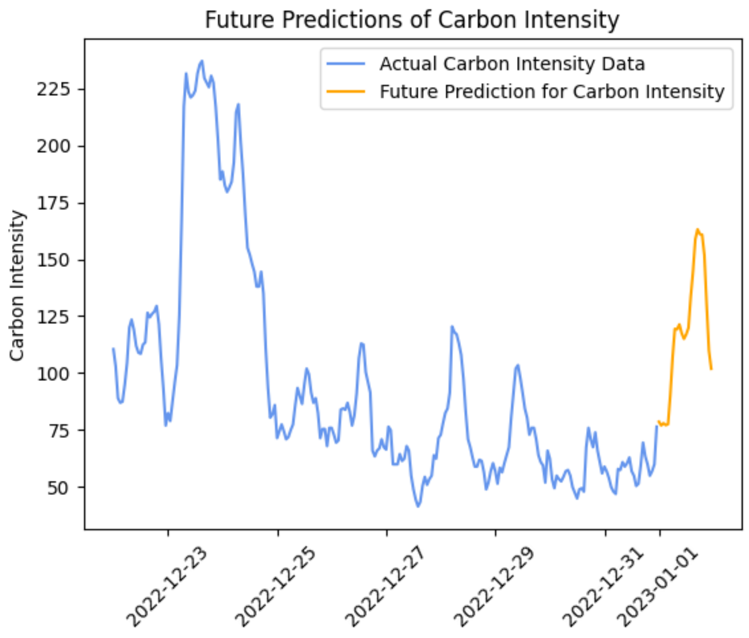


Figure 20 LSTM prediction for 24 hours beyond dataset

Figure 20 shows the LSTMs prediction of the 24 hours beyond the test dataset. The prediction fits well with the most recent data from the test dataset. As can be seen there is a high level of volatility in this eleven-day period, likely because this is during the middle of winter which is the most volatile time of year for carbon intensity so prediction can be challenging.

The final step of the evaluation was to calculate a performance metric to allow for comparison between the models. I chose to calculate the Mean Absolute Percentage Error (MAPE) between the test dataset and the models' predictions for the test dataset. I decided to use MAPE because it is a very common performance metric for time series data and quite simple to implement. Calculating the MAPE yielded a percentage of 24.82% for my LSTM.

5.2. *Extreme Learning Machine Evaluation*

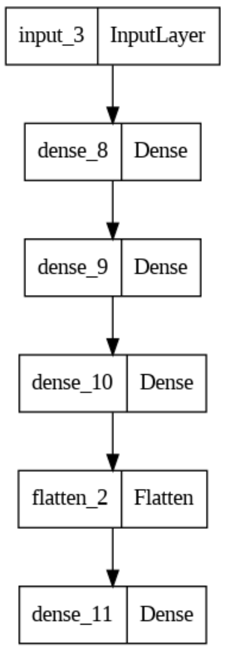


Figure 21 Layers of final ELM

Figure 21 shows the layers of the final version of my ELM. This first dense layer is the ELM layer which contains 100 nodes and has trainable set to false and the node weights assigned randomly. The next two dense layers both contain 24 nodes, are trainable and use the ReLU activation function. The main purpose of these two layers was to improve the accuracy of the model. Finally, the output is flattened and passed through a dense decision layer which contains 24 nodes, which is one node for each hour of the day to predict.

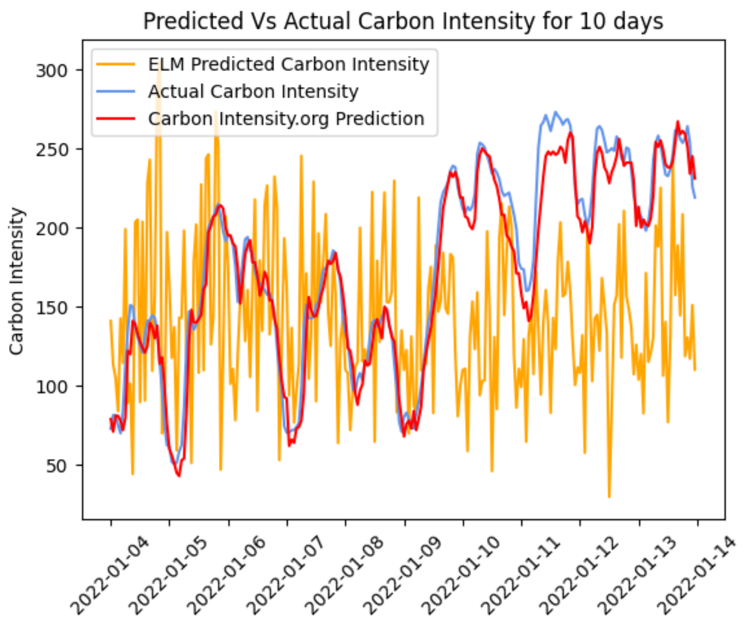


Figure 22 ELM next-step prediction for 10 days

Figure 22 is the same as Figure 19, but uses predicted data from the ELM instead of the LSTM. As can be seen the predicted carbon intensity values from the ELM are very far from the actual data and from the predictions made by the Carbon Intensity API. The predictions are also far from the valued predicted by the LSTM.

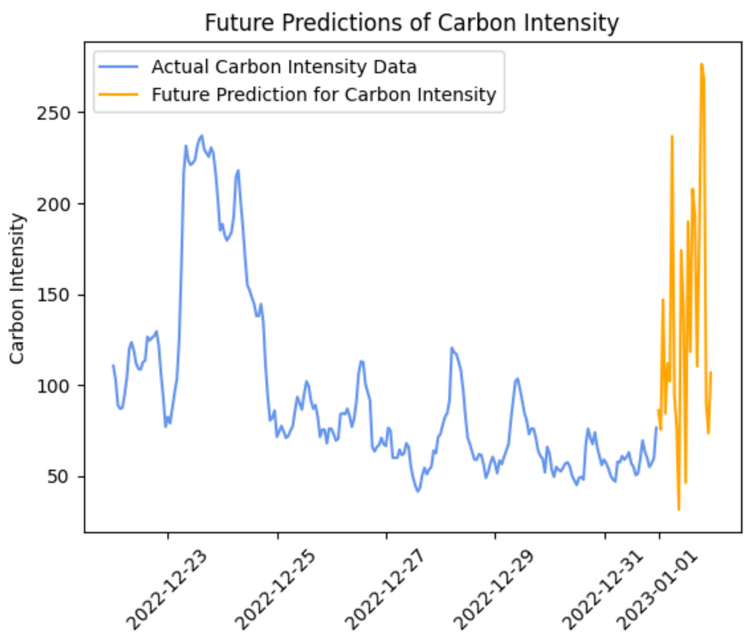


Figure 23 ELM prediction for 24 hours beyond dataset

Figure 23 is the same as Figure 20, but the predicted data from the ELM is used instead of the LSTM. Like Figure 22 the predictions made by the ELM are very poor, when compared to the LSTM and the predictions made by the Carbon Intensity API.

The final step was to calculate the MAPE from the ELM. The MAPE value I calculated was 43.17% which is significantly higher than the 24.82% from the LSTM. From the graphs produced and performance metric calculated the LSTM is much better than the ELM for carbon intensity prediction.

6. Conclusion

6.1. Aim and Objectives

6.1.1. Aim

The original aim of this dissertation was to develop a machine learning model capable of accurately predicting carbon intensity values 24 to 48 hours into the future. This was partly achieved because the LSTM developed can predict the carbon intensity values, however it is not as accurate as the predictions made the by Carbon Intensity API.

6.1.2. Objectives

1. Collect the necessary data needed for carbon intensity prediction.

   This objective was achieved successfully as a batch script was written to download the five years of carbon intensity data required for this project.

2. Develop multiple machine learning models to predict carbon intensity values.

   This objective was also achieved because a LSTM and an ELM were developed for carbon intensity prediction. While two models were developed, only one could predict carbon intensity values with some level of accuracy. This proved how important this objective was as a working model was produced which may not have been the case if only one machine learning model was developed.

3. Evaluate the machine learning models against each other, to select the best solution.

   This was also achieved, but due to the poor performance of the ELM there was not a need for extensive evaluation to select the best model. Because of this it would have been advantageous to have more models to compare, so that a more detailed evaluation could be carried out.

6.2. Future Work

This project only focused on univariate time series prediction, any future work should focus on multivariate time series prediction and investigate the best additional data to use for this application. Multivariate prediction should lead to improvements in results if the correct additional data is chosen. Some promising options include; temperature data, weather data, and oil price data.

Other machine learning models could also be investigated by future work, this dissertation investigated several possible options, and the most promising model not chosen for development would be deep learning. Deep learning uses many layers which can improve the learning of complex patterns in the data which will be very important if a multivariate approach is chosen.

This project focused on predicting the next 24 hours' worth of carbon intensity data. A future project could evaluate this projects models at predicting more than 24 hours ahead or develop completely new models for this purpose.

6.3. Final Conclusions

In conclusion this dissertation has managed to achieve the aim and objectives set out in introduction, but an accuracy comparable to the Carbon Intensity API's predictions was not achieved. Matching or beating their accuracy is very difficult in a short project and the lack of information on their prediction methodology makes this even more difficult.

The LSTM developed by this project could predict the carbon intensity values 24 hours into the future with a reasonable accuracy. The ELM on the other hand was not able to accomplish this, but likely could achieve similar results with significantly more training data.

References

[1] National Grid, “What is carbon intensity?,” [Online]. Available: https://www.nationalgrid.com/stories/energy-explained/what-is-carbon-intensity. [Accessed 23 04 2023].

[2] Our World in Data, “Emissions by sector,” [Online]. Available: https://ourworldindata.org/emissions-by-sector. [Accessed 23 04 2023].

[3] Gov.uk, “ UK's path to net zero set out in landmark strategy,” 19 11 2021. [Online]. Available: https://www.gov.uk/government/news/uks-path-to-net-zero-set-out-in-landmark-strategy. [Accessed 23 04 2023].

[4] K. Anderson and A. Bows, “Beyond 'dangerous' climate change: emission scenarios for a new world,” 13 01 2011. [Online]. Available: https://royalsocietypublishing.org/doi/full/10.1098/rsta.2010.0290. [Accessed 25 04 2023].

[5] H. Chen, S. Qi and X. Tan, “Decomposition and prediction of China's carbon emission intensity towards,” 15 06 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0048969722009317?casa_token=AMlA2m7Pn7YAAAAA:YJaq8rdTi-lKFVeRE_rcVIaw7eL96cj9s6JCr1EQhE0nUQhFcIhW2iCvXMGqyBxFdVEpiY1oh3w. [Accessed 01 05 2023].

[6] W. Sun and C. Huang, “Predictions of carbon emission intensity based on factor analysis and an improved extreme learning machine from the perspective of carbon emission efficiency,” 01 03 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0959652622000609?casa_token=HSO4SfzLRkMAAAAA:4ID_bogfM_g3vcowQW-ufY8E5ILHxf2OjXVs-ck4x3jLaA8HNwkvVoybOUGIsWmclMzEiktK8bM. [Accessed 02 05 2023].

[7] M. Jain, W. Andreopoulos and M. Stamp, “CNN vs ELM for Image-Based Malware Classification,” 24 03 2021. [Online]. Available: https://arxiv.org/pdf/2103.13820v1.pdf. [Accessed 02 05 2023].

[8] D. Maji, R. K. Sitaraman and P. Shenoy, “DACF: day-ahead carbon intensity forecasting of power grids using machine learning,” 28 06 2022. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3538637.3538849?casa_token=YnGF3lh

niesAAAAA:kN-3YU0Vaahlytj_kFRVTHuuZCMdSH52gZkd_HLW1bMikRHx1YQsI2uAL7mbHGpKP2qTQW81qXeexA. [Accessed 03 05 2023].

[9] F. C. e. al., “Keras,” 2015. [Online]. Available: https://keras.io/. [Accessed 03 05 2023].

[10] Carbon Intensity API, “Carbon Intensity API,” National Grid, [Online]. Available: https://carbonintensity.org.uk/. [Accessed 02 05 2023].

[11] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” 15 11 1997. [Online]. Available: https://papers.baulab.info/Hochreiter-1997.pdf. [Accessed 03 05 2023].

[12] DeepAI, “Multilayer Perceptron,” [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/multilayer-perceptron. [Accessed 04 05 2023].

[13] M. W. Gardner and S. R. Dorling, “Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences,” 01 08 1998. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1352231097004470?casa_token=NFwLOBMCSWIAAAAA:NxaIzHWe546WAjG79KHIxKbRgEISL_RMQ4lz8uq_kBPx_lR2OCSjWkzNQ-EP3RWp34XWS4NuT30. [Accessed 04 05 2023].

[14] K. O'Shea and R. Nash, “An Introduction to Convolutional Neural Networks,” 26 11 2015. [Online]. Available: https://arxiv.org/abs/1511.08458. [Accessed 04 05 2023].

[15] Y. LeCun, Y. Bengio and G. Hinton, “Deep Learning,” 27 05 2015. [Online]. Available: https://www.nature.com/articles/nature14539. [Accessed 04 05 2023].

[16] Learn Statistics, “Autoregressive (AR) Model: Understanding the Basics,” [Online]. Available: http://www.learn-stat.com/what-is-autoregressive-ar-model/. [Accessed 04 05 2023].

[17] G.-B. Huang, Q.-Y. Zhu and C.-K. Siew, “Extreme learning machine: Theory and applications,” 12 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231206000385?casa_token=zc4oISZQ3WEAAAAA:DbZnwY300q22KkQDX-

iXzVa3QMLXaCtylaj925DFXm-FHaydVzgFPCA4q4ekaeTqItqSAkt8PGY. [Accessed 04 05 2023].

[18] G. Van Houdt, C. Mosquera and G. Nápoles, “A review on the long short-term memory model,” 13 05 2020. [Online]. Available: https://link.springer.com/article/10.1007/s10462-020-09838-1. [Accessed 04 05 2023].

[19] F. Gers, J. Schmidhuber and F. Cummins, “Learning to forget: continual prediction with LSTM,” 07 09 1999. [Online]. Available: https://ieeexplore.ieee.org/document/818041. [Accessed 04 05 2023].

[20] M. Abbas, A. Albadr, S. Tiun and M. A. M. Sammour, “Spoken language identification based on the enhanced self-adjusting extreme learning machine approach,” 19 04 2018. [Online]. Available: https://www.researchgate.net/publication/324614872_Spoken_language_identification_based_on_the_enhanced_self-adjusting_extreme_learning_machine_approach. [Accessed 04 05 2023].

[21] Google, “Google Colab,” [Online]. Available: https://colab.research.google.com/. [Accessed 05 07 2023].

[22] Jupyter, “Jupyter,” [Online]. Available: https://jupyter.org/. [Accessed 05 07 2023].

[23] Pandas, “Pandas,” [Online]. Available: https://pandas.pydata.org/. [Accessed 24 05 2023].

[24] Google, “Normalization,” [Online]. Available: https://developers.google.com/machine-learning/data-prep/transform/normalization. [Accessed 08 05 2023].

[25] D. R. C. G. E. P. Box, “An Analysis of Transformations,” *Journal of the Royal Statistical Society: Series B (Methodological),* vol. 26, no. 2, pp. 211-243, 07 1964.