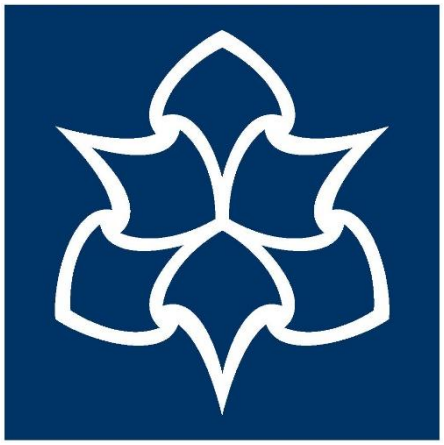
Energy Consumption Forecasting Using Hybrid Deep Learning Frameworks

A DISSERTATION SUBMITTED TO MANCHESTER METROPOLITAN UNIVERSITY
FOR THE DEGREE OF MASTER OF SCIENCE
IN THE FACULTY OF SCIENCE AND ENGINEERING



2022

By
Simon Wilson
Department of Computing and Mathematics

Table of Contents

List of Tables

List of Figures

Abstract

Throughout 2022, energy prices have been continuously rising throughout the UK, due to an increase of wholesale gas prices worldwide. As such, it is of mounting value to both suppliers and consumers to be able to manage their finances, as the both the cost of living and the cost of business increase.

This dissertation proposes the use of a hybrid deep learning framework in order to forecast energy consumption using smart meter data from a variety of differing households. Using this forecast, energy suppliers will then be able to prepare for future energy expenditures, in order to reduce waste, and in turn costs. This forecast will also enable consumers to make more educated decisions with regards to their energy consumption and finances, as the forecast can be displayed as a cost value on their in-home smart meter displays.

This paper both experiments with and discusses multiple approaches to solving this problem, through the use of both univariate and multivariate datasets, in addition to a variety of deep learning techniques, such as CNN-LSTMs, Encoder-Decoder LSTMs, and more. Justifications as to how data was cleaned and prepared so that it may be best suited to the problem at hand are also included.

Prior to experimentation, a literature review was conducted that discusses other pieces of work that also tackle similar time-series forecasting problems, in addition to a justification of the methodological approach taken to framework construction, and finally, an evaluation of the success of this dissertation, with suggestions for further study.

Declaration

No part of this project has been submitted in support of an application for any other degree or qualification at this or any other institute of learning. Apart from those parts of the project containing citations to the work of others, this project is my own unaided work. This work has been carried out in accordance with the Manchester Metropolitan University research ethics procedures, and has received ethical approval number 46079.

Signed:

X 

Simon Wilson

Date: 28 / 09 / 2022

Acknowledgements

With special thanks to my supervisor Olamide Jogunola, for providing me with many vital resources,

to Amy Wilson, for pushing me to be my best self,

to Daniel Whitehead and Jan Clare, for being excellent coursemates,

and to James Davies, for his kindness and hospitality.

Abbreviations

IHD	In Home Display
SM	Smart Meter
DL	Deep Learning
ML	Machine Learning
SVM	Support Vector Machine
ANN	Artificial Neural Network
RNN	Recurrent Neural Network
DCC	Data Communications Company
EEC	Embedded Edge Computing
IoT	Internet of Things
ARIMA	Auto Regressive Moving Average
SVR	Support Vector Regression
KNN	K Nearest Neighbours
AE	Autoencoder
SOM	Self-Organised Maps
LSTM	Long Short-Term Memory
BLSTM	Bidirectional Long Short-Term Memory
SWT	Stationary Wavelet Transform
DPA	Data Protection Act
MAE	Mean Absolute Error
RMSE	Root Mean Squared Error
FGSM	Fast Gradient Sign Method
BIM	Basic Iterative Method

1.0 - Introduction

1.1 - Background

Following the recent 54% (£1,277 per year to £1971 per year) increase of UK energy prices during April 2022 (Stewart and Bolton, 2022), in addition to the predicted 40% further rise in October of 2022 (Cornwall Insight, 2022), there is an increasing level of importance that correct energy meter readings are being produced. Wholesale gas prices are currently four times as expensive as they were in the beginning of 2021, meaning it is important for providers to receive correct meter readings, so as to not undercharge for their services and still make meaningful profits(Office for National Statistics, 2022). In comparison, the cost of living has been consistently rising for consumers, with an increase of 4.8% between December 2020 and December 2021 alone; according to the 'Opinions and Lifestyle Survey' from January 2022, 79% of respondents who reported an increase in their living costs attributed gas and electricity prices as a cause (Harris et al., 2022).

Across recent years, energy suppliers across multiple provinces of the United Kingdom have been required by their respective governments to provide devices called 'smart meters' (SMs) to their customers. SMs come with an 'In-Home Display' (IHD), which shows households their energy usage in near real-time, expressed in pounds and pence. The usage cost is calculated by measuring voltage and electrical current flow every 30 minutes, sending the data to a "nationwide secure smart network" called the 'Data and Communications Company' (DCC), who distribute the data to various users of their services, including the household's energy supplier, who will then calculate the cost, and update the value on the household's IHD (Bulb, 2022).

Using these IHDs and their associated SM data, both parties involved are already enabled to make educated, data-driven decisions based on the current energy usage cost that is displayed at present (Neto et al., 2021). This is due to consumers being able to plan their finances in advance due to the information, and providers using it to gain knowledge of

expected income. In order to further the value that both consumers and providers are able to gain through these IHDs, there have been many attempts to forecast future energy consumption using a variety of machine learning (ML) techniques. These techniques have been used with the intention of implementing accurate forecasting models into IHDs using SM data. Provided these accurate predictions can be achieved, it may be possible for households to receive predictive energy costs for multiple weeks in advance, as opposed to just their current expenditures for the month. This would be especially useful for consumers as it would allow them further agency when looking to better plan their finances and manage their energy usage, which is of increasing importance as the cost-of-living increases. Similarly, providers would be able to better understand the amount of energy they may need to produce within a given time frame, helping them reduce waste and optimise their costs.

1.2 – Project Focus & Aims

Specifically, this paper will primarily focus on the construction of an accurate machine learning or deep learning model with which to predict future energy consumption across a medium to long term period (2-4 weeks).

This shall be done with the intention of helping to create financial agency for both providers and consumers, through the providing of expected energy expenditures. With the expectation that these expenditures will be shown to consumers through IHDs, so that they may better plan their finances, and will be determined internally by providers, through usage of the predicted consumption output.

As such, the specific aims for this project shall be as follows:

- Build a robust ML/DL framework that is able to accurately predict energy consumption for a variety of households.
- Explore the effect of non-energy data being used to aid in the prediction of energy consumption (such as weather).

- Keep the computational cost of the framework appropriate, so that the model may be potentially deployed on an embedded device.

1.3 – Dissertation Objectives

Similarly, the objectives of this dissertation shall be to:

1. Conduct a literature review
2. Gather and prepare data necessary for training and testing a deep learning/machine learning framework.
3. Build an initial framework, training and testing the prepared dataset on it for comparison.
4. Carry out a comparative analysis of my framework versus other similar frameworks.
5. Conduct an experimental study in order to attempt to optimise the framework.
6. Write a complete dissertation report documenting the project.
7. Produce a video to present the project for the viva.

2.0 - Literature Review

Following the initial justification for the project, and when looking to achieve the goals laid out within the introduction, appropriate and relevant literature shall first be reviewed and critically assessed in order to gain a more diverse understanding of how to create a robust energy forecasting model. After gaining a deeper understanding of how others have approached similar forecasting problems, limitations shall be assessed in order to recognise the restrictions of the approaches, so that one may be more cognizant of what is realistically achievable. In addition to this, relevant ethical considerations shall also be addressed, in order to avoid a breach of ethics when selecting a methodology with which to solve this energy forecasting problem.

2.1 – Forecasting online vs offline

Sirojan et al. (2019) recognise the potential for running processes locally (offline) within SMs, defining it as a form of 'embedded edge computing', and stating that it will aid in removing complex issues such as "location sensitivity and privacy concerns in the smart meter data analytics", in addition to "reducing bandwidth and response time". The location sensitivity and privacy concern reductions are undeniably of high value, with these benefits being caused primarily due to a reduction in the number of entities required to process the data - this specific benefit shall be addressed more directly during the 'ethical considerations' section. It is debatable whether the bandwidth reduction is of any meaningful value, as the speed with which the information is garnered from the data does not immediately appear to be of particular importance, although it is possible that the small reduction compounds into a greater time-save when processed in large quantities, as it will reduce network congestion.

In order to understand the concept of 'Embedded Edge Computing' (EEC), it is first important to understand the 'Internet of Things' (IoT). According to Rose et al. (2015), the IoT generally refers to "scenarios where network connectivity and computing capability extends to objects, sensors, and everyday items not normally considered computers,

allowing these devices to generate, exchange, and consume data with minimal human intervention". Using this general definition, NXP (2022) define EEC as "a concept that has been developed… to support IoT deployments", as it "selectively moves cloud-compute processing from cloud data centres to edge and end node processing platforms". In short, some portion of the storage and/or computational resources are moved out of a central data centre. This definition further supports the privacy benefits outlined by Sirojan et al. (2019), as, when applied to the works of this dissertation, it means the DCC may not have to store as much data, reducing privacy concerns. It additionally supports the reduced bandwidth claim, although it is debatable as to whether reduced bandwidth is of value to this dissertation being it is unknown what the networks bandwidth limitations are.

With regards to this dissertation, EEC will enable energy providers to keep data at relevant locations where it is generated (such as individual households). As such, if an accurate ML/DL model is built and deployed into household SMs (the embedded edge device in this scenario), an accurate prediction of energy usage for a given month may be achieved locally, to then be sent off to the relevant supplier directly, reducing the number of involved parties in the process. Whilst it would in theory be possible for SMs to produce a cost value locally (instead of just the amount of energy), this would be ill-advised due to fluctuating energy prices mentioned within the introduction. This occurrence is known as "concept drift", an idea that Chollet (2021) defines as being when the properties of the production data change over time, resulting in a gradual decay of model accuracy. For example, equipment that is more energy efficient may become more common over time, gradually reducing energy consumption, resulting in the model overpredicting total consumption.

In contrast to Sirojan, Fekri et al. (2021) propose the use of an 'Online Adaptive RNN' (Recurrent Neural Network) when approaching an energy consumption forecasting problem, creating a DL model that "learns from data as they arrive". Their approach uses online pre-processing techniques to prepare the data for their model, which then tracks performance, and, if said performance is deteriorating, activates a tuning mechanism in order to adjust the

learning rates and improve accuracy. This approach to load forecasting has multiple benefits in comparison to the offline approach, such as potentially avoiding issues with concept drift, as it adapts to changes in energy consumption patterns. Additionally, as it does not need periodical retraining, meaning total computational time is reduced in comparison to batch learning. The benefit of this a result of no longer needing periodical retraining meaning the labour cost for maintenance of the model is lowered. However, it is possible that an improvement in the computational capabilities of SMs, in addition to an increase of efficiency in DL libraries would result in the model being easily improved upon within a few years' time, devaluing this benefit, as a new model would be developed. It could be argued that if the increase in accuracy is marginal, it would not be worth revisiting and rebuilding the model, however, Notton and Voyant (2018) show in their study that a 1% improvement when forecasting wind generation over a six-hour time period leads to savings of nearly two million dollars over the course of a year. This is not necessarily going to be a direct correlation to the results of this study; it is worth keeping this in mind when deciding whether to build either an online or an offline model.

2.2 - Machine Learning vs Deep Learning

Within this section, the benefits and drawbacks of ML and DL shall be assessed in order to gain an understanding of which approach may be better suited for this dissertation project.

Fekri et al. (2021) assessed multiple ML approaches to sensor-based forecasting, such as Auto Regressive Moving Average (ARIMA), Bayesian Approaches, and Support Vector Regression (SVR). They concluded that the performance of such techniques degraded heavily in the presence of non-smooth target signals with high variation – which are commonly present within load forecasting. The investigation conducted into the applicability of these approaches seems to be relatively limited, only seeming to show insight into the performance of ARIMA in comparison to an artificial neural network (ANN), so investigation of further studies shall be conducted in order to gain a more well-rounded understanding of the performance of the individual techniques. However, it shall be noted that this study did

find noticeable performance improvements when applying a DL approach instead of a more traditional ML approach to their problem. When Fekri et al., draw direct comparisons to approaches such as linear regression and k nearest-neighbours (KNN) regression, it is empirically noticeable that their RNN model begins to consistently outperform the other approaches when predicting over longer time periods.

Gajowniczek and Zabkowski (2014) also explored numerous methods used for load forecasting as they built a model designed to forecast up to one day ahead, again finding that approaches such as regression models and ARIMA would underperform in comparison to techniques such as ANNs. They state that there are downsides to ANNs, as they lack general explainability, however, this is overshadowed by their many positives. Said positives including the ability to generalise from noisy and incomplete data, and that they are strong approximators that are able to model any continuous function to any desired accuracy. Due to multiple studies supporting the use of ANNs as the most effective approach to load forecasting, they shall be integrated when looking to solve the problem at hand.

2.2.1 – Deep Learning Approaches

When constructing an ANN model, there are numerous different architectures that can be used. These architectures include, but are not limited to, Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Autoencoders (AEs), and Self-Organized Maps (SOMs). Depending on the problem at hand, different architectures may more applicable than others, with the opportunity to further enhance their performance via tuning of their hyperparameters. A “hybrid deep learning framework” may also be created, in which multiple of these architectures are strung together and implemented into a single model. When looking to educate the initial model, studies implementing both approaches shall be reviewed briefly within this section, in order to select the more appropriate method, with deeper study being carried out during the methodology and ablation study sections in order to appropriately tune model hyperparameters.

Bouktif et al. (2018) explored the usage of an RNN type network in the form of a long short-term memory (LSTM)-based network. In their study they found that they were able to accurately forecast energy consumption accurately over both short and long periods of time (a few days up to a few months). It should be noted however, that this study was more focused on forecasting total average energy consumption so that providers may make better business decisions, with little thought towards forecasting for individual households. This gives their model much more flexibility, and this should be kept in mind when constructing an initial ANN.

Dang et al. (2021) looked at combining LSTM and CNN networks with a support vector machine (SVM) when tackling a sentiment analysis problem. They found that combining two or more architectures together resulted in the advantages of both being incorporated, helping to increase the versatility of the model, and cover more potential weaknesses. Even though the problem at hand is different in the study of Dang et al., it is likely that the outcome will be the same when trying to implement a similar solution into this dissertation.

Jogunola et al. (2022) also explored creating a hybrid framework. Within their study they experimented with various combinations of architectures when creating a hybrid model. Ultimately, they decided to combine a CNN, a BLSTM (bidirectional long short-term memory) paired with an AE to encode data, and an LSTM paired with an AE to decode data. It is also shown that the framework created within this study performed significantly stronger than vanilla LSTM models. An important note with regards to this study is that particular consideration was taken towards the computational cost of the model. It was found that their final model required roughly 75.2% less computation time than a vanilla LSTM. This is important for this dissertation project, as it is looking to keep computational cost to a minimum, so that the end product may potentially be deployed to an embedded smart meter device. As such, the knowledge that increasing the amount of DL architectures within a given framework does not necessarily increase computational cost shall be carried forwards.

Yan et al. (2019) explore combining an LSTM network with a stationary wavelet transform (SWT) when creating a hybrid framework that will be able to forecast for individual households. The use of the SWT helps to increase the dimensionality of the data, whilst also reducing the volatility that is frequent when predicting for individual households due to various day-to-day inconsistencies in external energy usage factors. Through this approach, their results show that the proposed method can outperform other methods such as support vector regression (SVR), LSTM networks, and CNN-LSTMs when looking to predict short-term energy consumption. Though this approach seems to be largely successful when looking to perform strongly with respect to individual households, it is possible that the increase in dimensionality, and resulting increase in computational cost, may be too computationally expensive for an embedded device to process.

Kim and Cho (2019), propose the usage of a CNN-LSTM network in an attempt to extract spatial-temporal features to effectively predict energy consumption. Similar to the other hybrid approaches, the CNN layer is used to extract important features from the data, so that these features may then be fed into a form of RNN (in this case an LSTM). Additionally, it is confirmed that the CNN layer has, in this case, reduced the data spectrum, meaning that there is a reduction in computational cost.

2.2.2 - The Limitations of Deep Learning

Whilst there are many benefits to using a deep learning approach when approaching a time-series forecasting problem, it is also crucial to understand their limitations and restrictions. These limitations shall be addressed and discussed during this section so that awareness may be raised, in order to try and avoid these issues as the model is being constructed.

A key issue that DL approaches may face is that they may learn to predict based on the wrong features, resulting in the model struggling when being applied to new data. Mitchell (2019) gives an example of this in relation to a CNN, in which a model learned to recognise

photographs of birds by identifying blurry backgrounds. Whilst this specific example is in relation to an image recognition problem, it is still possible that the same phenomenon may occur when building a model that is being applied to a time-series forecasting problem. There is likely no definitive way of avoiding this, although a general mindfulness of this issue must be maintained whilst preparing and training the dataset, in order to try and notice the potential for this issue to occur.

In a similar vein to the prior potential issue, Papernot et al. (2016) investigate how to defend against a key issue deep learning approaches have that can cause them to misclassify or even improperly train – adversarial samples. Adversarial samples are defined as “an input to cause deep learning algorithms to misclassify”, these samples are created so that the intentional output is clearly visible to the human eye, but due to slight modifications in the input, the model incorrectly predicts the output. The example that Papernot et al. give is in relation to an image recognition problem, and as such a debate could be made that the study being assessed and the study at hand differ too heavily to be applicable. However, Mode & Hoque (2020) discuss tackling adversarial examples in a multivariate time-series regression scenario. When adding adversarial samples into the training data by using the “fast gradient sign method” (FGSM), and the “basic iterative method” (BIM). The application of these adversarial attack methods resulted in accuracy increases across all tested models, especially in the CNN and LSTM models (25.9% and 22.9%, respectively). It is noted within the study that BIM “shows more effectiveness in crafting a stealthier adversarial example”, however, it also comes with a higher computational cost. Whilst it notes that BIM is a higher computational cost, there does not appear to be any mention of how computationally expensive these methods are at a base level, but they may still be worth exploring when looking to increase model resilience and versatility.

A consistent historical issue within DL models is their explainability. Ras et al. (2021) address these issues, and offer multiple approaches when looking to make DL more accessible and understandable to stakeholders, these approaches include:

- **Visualisation methods:** highlighting input characteristics that strongly influence output
- **Model distillation:** Creating a separate, explainable “white box” machine learning model that can mimic the behaviour of the network.
- **Intrinsic methods:** Networks that have been created with the intention of rendering an explanation alongside its output.

The study of Ras et al. would have very strong prospects when being applied to an online model, however it is debatable as to whether or not they would be worth implementing into the solution of the problem at hand. The implementation of these methods would likely result in greatly increased computational cost, which is a limited resource when looking to implement a model into an embedded device. As a possible further study it would be intriguing to see if there would be a method to implement such ideas into this study, however, in the interest of keeping the scope a reasonable size, it is not likely that this will be explored any further.

One final limitation that has been mentioned prior within this dissertation is the idea of concept drift. Whilst it has already been explained in section 2.1 (Chollet, 2021), and appropriate actions that may be taken in attempt to combat such an issue have already been defined, it is being reiterated within this section as it is extremely important to consider when looking to understand the limitations of a model’s long-term relevance and validity.

2.3 - Ethical Considerations

General nod to the moral and ethical issues that must be considered when building such a model, from data privacy, to how the model may be exploited. Provider/consumer relationship, current record profits from providers, although they also faced massive losses in Q1 2022.

In this section, consideration shall be given towards the potential ethical issues involved with the usage of user data, in addition to the potential ethical misuse of DL models in a business setting.

In any data-related project, it is imperative to maintain a strong level of cognizance with regards to any potential ethical and moral dilemmas, so that user information is not misused, and furthermore, is kept secure. The data protection act (DPA) is the UK's version of the general data protection regulation (GDPR), and is enforced with the aim of laying out clear protocol as to how data must be handled and kept secure in order to avoid potential misuse of data, and breach of trust by making sure information is:

- Used fairly, lawfully, and transparently.
- Used for specified, explicit purposes.
- Used in a way that is adequate, relevant, and limited to only what is necessary.
- Accurate and, where necessary kept up to date.
- Kept for no longer than is necessary.
- Handled in a way that ensures appropriate security, including protection against unlawful or unauthorised processing, access, loss, destruction, or damage (GOV.UK, 2018).

Throughout each stage of this project, all data used shall be handled following the above legislation, so that no ethical concerns are raised.

One of the key ethical points to consider as this dissertation is carried out is the potential for users to be identifiable through their energy consumption data. In the case of a dataset within this study containing information that may possibly be used to extrapolate personal user information, it shall be cleaned from the dataset, in order to ensure compliance with the DPA.

It is possible that the model created as a result of this study may be unethical, or even immoral, in ways that do not directly involve a breach of data privacy. For example, it is

possible that the model may overpredict or underpredict energy usage, which could then in turn result in consumers receiving increased prices, or providers losing profit. There has been recent uproar within the media, as many energy providers have been receiving record-breaking profits. Following this uproar, one could say that it would be more moral to favour the consumer, as providers are mindlessly profiteering. However, upon inspection of a recent sales report from British Power (BP, 2022), it is stated that they faced a loss of $20.4 billion during the first quarter of the year, with them reporting $9.3 billion in profit during the second quarter. As these second quarter profits must also be used in order to recoup their lack of first quarter profits, it is thus debatable as to whether or not these record profits are a form of corporate greed, or a necessity due to massive losses. Due to this ambiguity towards the reality of the situation, it would be difficult to assume a definitive moral stance. Regardless of whether or not a stance could be assumed to be moral, it is important to remain objective, with an awareness of potential model biases being noted, so that actions may be taken to avoid unfair treatment of any parties directly affected by implementation of the model.

3.0 - Methodology

The following section shall discuss the decision-making process when looking to create the model, with justification where appropriate.

3.1 – Selecting an Approach

Following the evaluation of prior studies that has been carried out across the literature review, this section shall summarise and evaluate said studies, so that a foundation may be laid out when looking to create an initial model for this dissertation.

3.1.1 - Online vs Offline

When assessing prior works that contained models running either online or offline, two main studies were reviewed. Sirojan et al. (2019) implemented an offline model into their forecasting solution, building it once, and revisiting it for maintenance annually. Whereas, in contrast, Fekri et al. (2021) proposed the use of an online model that learned from new data as they arrived.

There were several benefits and drawbacks to each method, however, ultimately, for the project at hand, an offline model shall be built. Despite there being great benefits to building an online model that can retrain itself, with lack of a need for maintenance, it is not unlikely that in a few years it will be possible to build a stronger model, meaning the lack of a need for maintenance holds little value. Similarly, the privacy benefits of being able to keep user data mostly local using the offline model will likely be of great benefit to the average consumer, and it does not seem as though there will be any alteration to this benefit over the coming years.

3.1.2 - ML vs DL

Now that it has been decided on whether the model shall operate online or offline, the technical approach must be decided upon. When selecting the technical approach, particular

respect shall be paid to the fact that the goal of this project is to create a model that is able to operate on an offline embedded device, and not from a separate network and/or location.

Both Fekri et al. (2021) and Gajowniczek and Zabkowski (2014) explored ML approaches to model building within their studies, although, as was discussed during section 2.2, found that there were glaring potential weaknesses that such approaches struggle with. These models' lack of ability to handle noisy and/or incomplete data may have significant repercussions upon accuracy if applied to this project.

Fekri et al. and Gajowniczek and Zabkowski also explored DL approaches to their respective problems, finding that accuracy metrics were consistently stronger for DL models. This is likely due to the ability of ANNs and RNNs to be able to generalise from noisy data, albeit at the cost of explainability, giving them an edge over the ML approaches.

As both the studies reviewed ultimately found that DL models provided consistently better accuracy metrics, the same route shall be explored for this dissertation project. Furthermore, the fact that ML approaches struggle with noisy data, when paired with the knowledge that this project will likely use a large amount of data, further dissuades the usage of an ML approach. This is because the large amount of data has the potential to end up containing a lot of noise, resulting in very poor accuracy.

As a final note, it will be important to maintain an awareness of the computational cost of a DL approach, as they have the potential to become very computationally expensive. This awareness will ensure that the model can successfully fulfil the aim of this dissertation, in which the model may be implemented into a smart meter.

3.1.3 - Deep Learning Approach

Now that DL has been selected as the modelling approach, an architecture/framework must be decided upon so that the model may be built. As was discussed in section 2.2.1, there are numerous architectures that may be used to build a DL model, with these architectures potentially being strung together into what is called a "hybrid framework".

Initially, the study of Dang et al. (2021) was assessed in which a hybrid DL approach was applied to a sentiment analysis problem. In their study they found success through the combination of a CNN, LSTM, and SVM, noticing that the models appeared to aid in covering the weaknesses of the other models. As it has already been established that DL approaches tend to be computationally expensive, it would be fair to assume that this approach may result in the computational cost of the model being too high for an embedded device. However, in the study of Jogunola et al. (2022), it was found that the combination of CNN and LSTM architectures resulted in a reduction of computational cost when compared to the integration of solely a vanilla LSTM architecture, with further reduction in computational cost for their proposed model, which combines a CNN, BLSTM, and an AE. The approach chosen within the study of Jogunola et al., in which computational cost of embedded devices is considered whilst an accurate model is also created, strongly suggests that a hybrid approach should be used for this dissertation.

When looking to continuously optimise accuracy for individual households, the study of Yan et al. (2019) in which a SWT was implemented into their hybrid model may be considered, although due to the offline nature of the model, an assessment will be made during the data preparation section of this dissertation (section 3.4), so that the dimensionality and size of the data may be accounted for when applying this technique. It is also worth noting that the scope of study for Yan et al. is different to the intended scope of this dissertation, as Yan et al. aim to predict within a short period of time, whereas this dissertation will aim to predict across medium to long periods of time. As such, the implementation of SWTs is not explored within this dissertation, however it is a potential recommendation for further work.

Following the review of these studies, this paper shall implement a hybrid deep learning framework when attempting to arrive at a well-optimised solution to the energy forecasting problem at hand.

3.1.4 - Ethical Awareness

In order to comply with the ethical standards laid out within section 2.3, the following measures will be taken:

- Data will be collected from the public domain, in order to avoid any risk related to breaching of data privacy regulations.
- An objective stance shall be maintained when constructing the model, so as not to favour either consumers or providers.
- Any data that may be potentially used to identify specific household locations, or specific individuals, shall be either removed or encrypted so that there is no risk that the data involved is used with malicious intent.

3.2 - Measuring Success

Before creating the model, it is important to define how success shall be measured. Typically, studies that have been assessed within the literature review have used common accuracy and loss metrics such as root mean squared error (RMSE), and mean absolute error (MAE). However, many of these studies have not specifically been working to create offline models that have key computational cost restrictions. Due to the restrictions at hand when creating the model for this study, a method of tracking computational usage would be greatly beneficial. Packages such as ‘tflite’ in Python allow for the deployment of ML/DL solutions to embedded devices, creating a compressed file that contains the model, that may then be deployed to an embedded device. This would be the ideal solution, however, due to a lack of immediate access to a device that the files may be deployed on, it is uncertain how feasible it will be to accurately measure computational cost using this package.

Development frameworks such as ‘renode’ allow for simulation of embedded systems, sothis was considered when trying to generate computational cost as a metric of success when creating the framework. However, it was deemed inappropriate to attempt to use this, as there did not appear to be a simulation available for a smart meter. Additionally, there are

multiple versions of smart meters that have been distributed throughout the UK (Mostyn, 2021), and there is a high likelihood that they carry different hardware capabilities.

The primary accuracy metric chosen for this study shall be RMSE, as MSE penalises outliers to a greater extent than RMSE does, so RMSE should provide a more balanced accuracy metric to base model performance off. Furthermore, attention shall be paid to model build time as more complex models are built, so that it may be used a rough guideline for computational cost, in place of simulating it. However, it is important to note that build time is not always a strong indicator of the computational cost of a model, being some model hyperparameters may inflate training times without increasing model complexity.

3.3 - Data Collection

The data in this study was collected from the London Datastore (UK Power Networks, 2022) and Kaggle (Jean Michel, 2022). The former source is a publicly available, balanced, and representative dataset containing electrical energy usage data for 5,567 households across London, between November 2011 and February 2014. Each household has a unique location identifier, which does not give any leading personal information that may aid in identifying the household's address, or the occupants' personal data. The latter source was used to retrieve daily weather data for London across the timestamps of the former dataset, this data was retrieved using the darksky application programming interface, and does not contain any personal information of any kind.

As such, the data within this study appears to be ethically sound, with no risk of leaking identifiable personal information. This, in tandem with the intended usage of the data within this paper, means that there are no immediately foreseeable ethical dilemmas across the course of the project.

It should be noted that the London Datastore dataset was split into over one hundred blocks of data, each with one million entries. In the interest of computational cost for training, the

first twenty blocks were selected, providing twenty million samples to create a model from. There were attempts to incorporate more, although this caused frequent crashes of the notebook environment in which the data was processed in.

3.4 - Data Preparation and Cleaning

In order to begin forecasting from the time-series data, it must first be cleaned and prepared so that it is in an acceptable format for the specified model to be able to train and test on it. Famili et al. (1997) discuss the importance of maintaining a proper approach to data preprocessing. Assessing common issues that are prevalent within many datasets, in addition to potential strategies that may be implemented in order to work around these common issues. Whilst the work of Famili et al. is relatively dated, it should still maintain a strong level of relevance and importance when considered in relation to the work being carried out within this paper. This is due to the fact that when compared to more modern works that assess data preprocessing, such as that of Sukumar et al. (2016), the foundational principles appear to be innately similar.

3.4.1 – Household Energy Consumption Dataset

Upon initial inspection of the London energy consumption dataset, there were four included features:

- ‘LCLid’ – Unique household identifiers (e.g., MAC000002)
- ‘stdorToU’ – Type of energy prices (‘Std’ or ‘ToU’)
- ‘DateTime’ – Date + half hourly timestamp (e.g., ‘2012-10-12 00:30:00.000000)
- ‘KWH/hh (per half hour)’ – Energy used in the sample space (e.g., ‘0.27’)

‘LCLid’ was retained, as it may be used to select individual households to train and test upon, additionally, when models are constructed, households that have not been trained or tested upon may be used as a validation set. Furthermore, due to the ambiguity of the

identifiers used, there is no risk of a personal data leak, meaning there are no ethical issues related to keeping this feature in the data.

‘stdorToU’ was omitted, the feature contained information as to how customers were informed of their current energy prices, through either standard methods, or a dynamic time of use system. The dynamic system may potentially result in reduced energy usage as customers are more aware of their consumption habits in relation to current energy prices. However, it is debatable as to how significant this effect may be. As such, the decision was made to exclude this feature.

‘DateTime’ was retained, albeit renamed to ‘datetime’ in order to follow common naming conventions. This is the time-series component of the dataset, and will be crucial when looking to forecast energy consumption, in its initial state, it is in half hourly timestamps. As it is in this continuous format, and in order to eliminate unnecessary components of the feature, the milliseconds were removed.

‘KWH/hh (per half hour)’ is a vital component to the problem at hand, as it is the target feature that will be actively predicted by the model, and as such, was also retained. In order to neaten the feature, and reduce redundancy in its name, it was renamed to ‘energy (KWH/hh)’.

Upon continued inspection, it was found that there were null values within the ‘energy (KWH/hh)’ column were inputted as the string ‘Null’. These strings were replaced with numpy’s ‘NaN’ value, so they may be caught by a null checking function. When looking to fix these null values, the works of Pratama et al. (2016) and Fortuin et al. (2020) were both reviewed. Pratama et al. propose multiple methods of dealing with null values in time-series data that range in complexity, such as mean imputation, autoregressive imputation, and incorporating an SVM that creates a “local time index” to use in training patterns. Fortuin et al., implement multiple imputation methods into the ‘Healing MNIST’ time-series dataset, and test accuracy of models built using each approach. They found that the MSE is very similar

across many of their explored methods of imputation when applied to a time-series problem, with all metrics being within roughly 0.1 MSE of each other. In the interest of simplicity, “Forward Imputation” was selected, as there were only 584 null values in a dataset of two million samples, meaning it is likely any skew caused would be insignificant in relation to the overall scope. If this study was to increase in complexity, with more null data, then one of the more complex methods proposed by Pratama et al. would likely have been selected to impute data.

In order to ensure that the households being used to train the model covered a fair and representative sample of ‘datetime’, a filter was applied to household identifiers in the dataset, so that ten households were left, with at least 39719 samples each (roughly 827 days of data). To then ensure that these ‘datetime’ values were continuous, a filter was applied to ‘datetime’, filtering out any ‘datetime’ samples with less than ten unique entries. It was found after executing this function that there were duplicate entries within the dataset, which were then removed using a drop duplicates function.

In order to double check each unique household id had the same range of dates, the minimum and maximum ‘datetime’ values were printed for each id. The final date range for the dataset is from the 27th November 2011, up until the 1st December 2013 – this covers 105 standard weeks (Sunday-Saturday) in total, and allows the dataset to be trained on the first year of data, and then tested on the second.

To match the scope of the study, which is looking to predict energy consumption up to a month in advance, the half hourly data was then aggregated into daily data. It should be noted that, whilst this aggregation could have been carried out earlier in the preparation process to reduce computational time of further data cleaning, the duplicates would have then been aggregated, resulting in a skew of the data.

3.4.2 – Weather Data

As the London dataset is purely univariate, the relevant weather data to the London dataset was sourced in order to create a new, multivariate dataset containing energy consumption information, in addition to information about the daily weather. These extra features are intended to result in an improvement of accuracy metrics for energy forecasting in comparison to a univariate model. Of the 32 features in the weather data, only six were kept in order to keep dimensionality to a reasonable size. The six remaining features are as follows:

- 'cloudCover' – A float between 0 and 1 stating what percentage of the sky is covered by clouds (e.g., 0.32)
- 'windSpeed' – A float stating wind speed in meter/sec (e.g., 3.77)
- 'humidity' – A float between 0 and 1 stating the percentage of humidity (e.g., 0.95)
- 'temperatureLow' – A float in degrees celcius (e.g., 10.87)
- 'temperatureHigh' – A float in degrees celcius (e.g., 3.09)
- 'date' – In the format 'YYYY-MM-DD'

'cloudCover', 'humidity' and 'windSpeed' were selected in tandem, with the intention that the three features together may be an indicator of rain, which may then in turn increase energy consumption, as residents are less likely to leave their households.

'temperatureLow' and 'temperatureHigh' were also selected with the intention that residents are more likely to spend time outside of their household if the temperature is on the mid-high end. There is also the hope that a CNN-LSTM network will be able to extract some form of seasonality trend, with residents more likely to go outside on cool summer days, and warmer autumn and winter days, though, as has been mentioned in section 2.2.2, the inexplainable nature of DL models will make it difficult to know if this is actively occurring.

‘date’ was selected so that this dataset may be joined onto the aggregated energy consumption data, so that the weather data is appended to the relevant day in the main dataset.

Upon inspection of the contents of the dataset, it was found that there was a single null value for ‘cloudCover’, this null value falls on the 1st January 2014, meaning the sample will be omitted when joined onto the energy consumption data, as such, the ‘cloudCover’ value will be forward filled for this sample, as it will have no effect on the model.

This dataset was then joined onto the already prepared energy consumption data using the ‘date’ and ‘datetime’ columns respectively, resulting in the daily weather data being added to the appropriate dates of the other dataset. This multivariate dataset was saved to a new file, so that models may be built off both the univariate and multivariate data, in order to compare model performance between the two as model complexity changes.

It should also be noted that the decision was made to not standardise this dataset, in order to aid with interpretability of the accuracy results, and so that stronger comparisons may be against the univariate dataset. An awareness shall be maintained that there may be a drop in the accuracy of the multivariate model as a result of this decision.

Now that an understanding of the dimensionality of the dataset has been gained, the decision was made to not incorporate the usage of a stationary wavelet transform, as per the concerns raised within section 3.1.3.

4.0 – Experimental Studies

4.1 – Architectural Components

Before beginning to build initial frameworks, common components of the architectures shall be explained, so that the architectures created throughout the whole of section 4 may be more clearly understood.

4.1.1 - Activation Functions

Activation functions are implemented within layers of neural networks in order to help them learn complex patterns. There are multiple activation functions that may be implemented within deep learning (DL) models, such as:

- Sigmoid – A sigmoid activation function takes a value as an input, and then outputs values in the range of 0 to 1. The sigmoid function primarily suffers from a vanishing gradient problem, which can cause issues when training models.

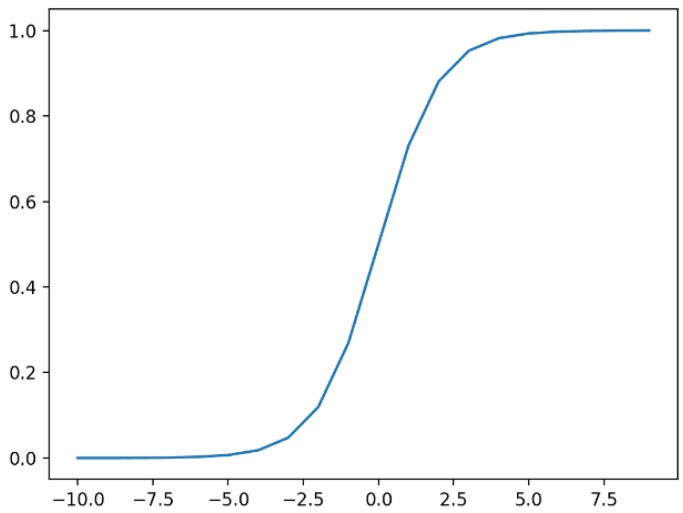


Figure 4.1 – Sigmoid function visualisation

- Tanh (Hyperbolic Tangent) – Very similar to the sigmoid function in that it returns values in the range of 0 to 1, except it takes values within the range -1 to 1. Tanh also suffers from a vanishing gradient problem, which can cause issues if used extensively within DL models.

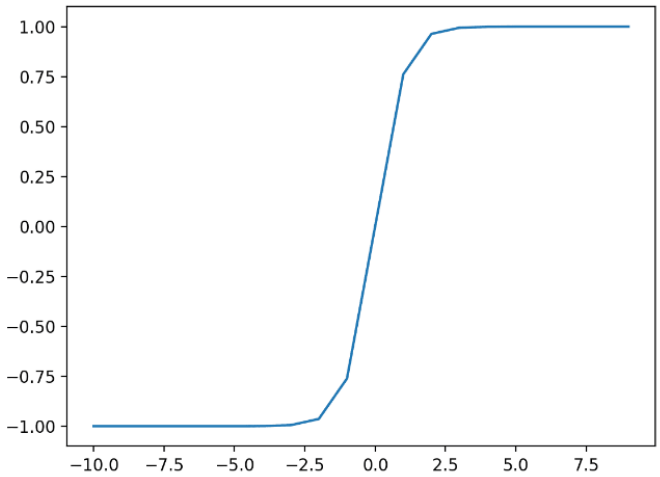


Figure 4.2 – Tanh activation function visualisation

- ReLU (Rectified Linear Unit) – This is the most popular activation function, as it is extremely simple, in addition to being very effective. The ReLU function returns 0 if a value is negative, and will otherwise return the initial input value. Additionally, the ReLU function does not suffer from the same vanishing gradient issue that both sigmoid and tanh struggle with.

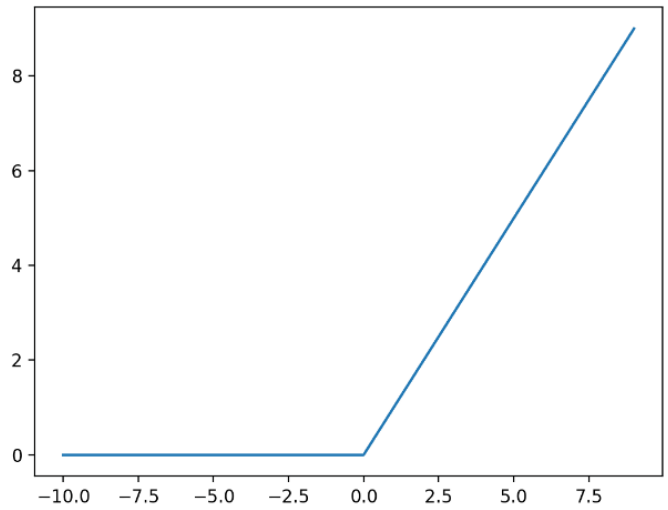


Figure 4.3 – ReLU activation function visualisation

As such, ReLU will be used as the primary activation function throughout every model created, if not used as the sole activation function, due to its effectiveness, simplicity, and ability to avoid training issues that sigmoid and tanh may struggle with.

4.1.2 - Epochs

The ‘epochs’ of a model is the number of times that the learning algorithm will operate on the entire training dataset. When measuring the computational cost of a model using the training time, a high number of epochs will result in a longer training time, despite model complexity not necessarily increasing in relation. With this said, however, it is possible that more complex models will require more epochs in order to properly optimise themselves.

4.1.3 - Batch Size

Batch size is used to pass a specific number of samples to the model, one batch at a time, so that the model may update its weights more frequently.

4.1.4 - LSTM Layer

A type of RNN layer that can learn the dependencies in sequential time-series data. The main parameter of an LSTM layer is the ‘units’, which is the number of scalars the function will return to feed into the next layer. There is an adaptation of the LSTM called a BLSTM, which can read sequential time-series data both backwards and forwards, helping to improve model accuracy.

4.1.5 - RepeatVector Layer

Used to repeat sequences n times (for example, 7 times when trying to access daily data that has been aggregated into week-long arrays), this is useful for when trying to predict for multiple timesteps within a sequence.

4.1.6 - Flatten Layer

Can be used to vectorise sequential data (for example, a two-dimensional matrix). This can be useful as Dense layers are only able to process vectors.

4.1.7 - Dense Layer

Dense layers (also known as fully-connected layers) are layers in which every neuron is linked to every neuron from the preceding layer. Dense layers are primarily used at the end of the network, in order to produce an output prediction.

4.1.8 - Conv1D Layer

Creates a convolution kernel involved with the layer output over a single dimension, to then produce a tensor of outputs.

4.1.9 - MaxPooling1D Layer

A MaxPooling1D layer is used to simplify feature maps by keeping 25% of the values, specifically those with the largest signal. This helps with reducing the chance of the model overfitting, as well as reducing model training time.

4.1.10 – ConvLSTM2D Layer

Takes an array input with the following elements, in order:

1. Samples
2. Timesteps
3. Rows
4. Columns
5. Channels

This allows the model to split time periods into further subsequences, allowing for stronger learning of key features within the data. This input is then converted into convolutions, so that they may be inputted into the LSTM units.

4.2 – Creating a baseline

According to Brownlee (2019) when facing any new prediction problem, a simple naïve baseline model must first be created in order to gain a “quantitative idea of the initial difficulty of the problem”, and to also provide base metrics which more advanced architectures and methods may be evaluated against. Naïve baseline models have been created for this dissertation that make a prediction by carrying over the data from the previous day, the previous week, the previous month, the same week from one year ago, and the same month from one year ago. These baselines have been constructed using a single household from the dataset. The results can be seen in figure 4.1 and table 4.1.

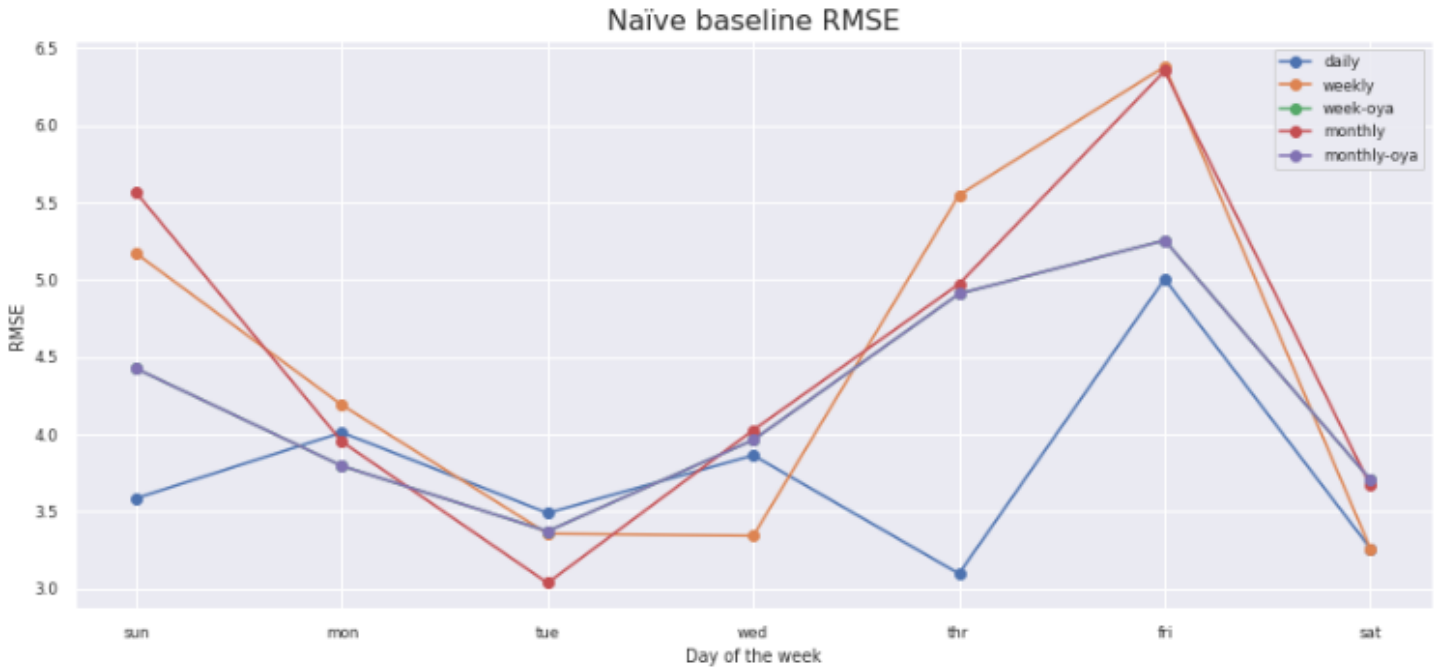


Figure 4.4 – Naïve baseline RMSE score plot

Naive baseline RMSE scores

	Average	Sun	Mon	Tue	Wed	Thur	Fri	Sat
Daily	3.805	3.6	4.0	3.5	3.9	3.1	5.0	3.3
Weekly	4.612	5.2	4.2	3.4	3.3	5.5	6.4	3.3
Weekly – o/y/a	4.252	4.4	3.8	3.4	4.0	4.9	5.3	3.7
Monthly	4.638	5.6	4.0	3.0	4.0	5.0	6.4	3.7
Monthly – o/y/a	4.252	4.4	3.8	3.4	4.0	4.9	5.3	3.7

Table 4.1 – Naïve baseline RMSE score table

Upon initial assessment of this baseline model, it appears as though energy consumption has less variance on Mondays, Tuesdays, Wednesdays, and Saturdays, than on Sundays, Thursdays, and Fridays. This trend should be expected to continue as future models are developed, as the data naturally appears to trend this way, as such, if this trend continues, it shall not be treated as a form of overfitting. Initially, it also seems apparent that carrying forward data from the previous day provides the most accurate baseline prediction results, although it is expected that this will be subject to change as models are iterated and developed upon, as DL models will likely be able to learn complex trends in longer time periods of data.

4.3– Constructing initial frameworks

As there are two separate datasets, so that univariate and multivariate approaches may be explored (as discussed in section 3.4), the framework construction for each dataset shall be separated into differing subsections.

4.3.1 – Initial Univariate Framework

The initial architecture constructed for the univariate dataset was an “Encoder-Decoder LSTM”, this was composed of a subset of two models.

Firstly, an encoder comprised of an ‘LSTM’ layer and a ‘RepeatVector’ layer was used to read and encode the input data.

Secondly a decoder was constructed to read the encoded input using a second ‘LSTM’ layer, making one-step predictions for the elements of the output vector.

After the output was decoded into a sequential format, the sequence was parsed through a fully-connected ‘Dense’ layer in order to interpret each individual time step (each day), before producing an output prediction for each step. It is important to note that a ‘TimeDistributed’ wrapper was used in the fully-connected layer so that wrapped layers could be used for each time step in the output from the decoder, as opposed to a single scalar output for the week.

The layers and their hyperparameters are as such for these initial frameworks:

- Epochs = 20
- Batch Size = 16
- LSTM (Encoder): units = 200, activation = ReLU
- RepeatVector Layer
- LSTM(Decoder): units = 200, activation = ReLU
- Dense Layer (TimeDistributed): units = 100, activation = ReLU
- Dense Output Layer (TimeDistributed): units = 1

The aim of this initial framework was primarily to provide a simple model that may also be used as a baseline, in order to measure improvement specifically between different DL framework approaches. More advanced models that are constructed using guidance from the research assessed within section 2.2.1 shall be created during sections 4.4 and 4.5, as comparative analysis and experimental studies are carried out to improve upon these initial models.

The RMSE scores from the univariate model can be seen in figure 4.5 and table 4.2. As expected, there is a lower average RMSE score for predictions on Monday, Tuesday, Wednesday, and Saturday, in comparison to Sunday, Thursday, and Friday. Furthermore, as predicted, the DL model has started to achieve stronger accuracy metrics when using longer time periods of data in order to forecast energy consumption (with 7 and 28 days being the strongest). This model, whilst simple, is already showing forms of improvement over the naïve baseline.

The improvement of this univariate model over the baseline is likely due to the low dimensionality of the dataset used, making it easier for the DL model to interpret patterns, allowing it to optimise quickly.

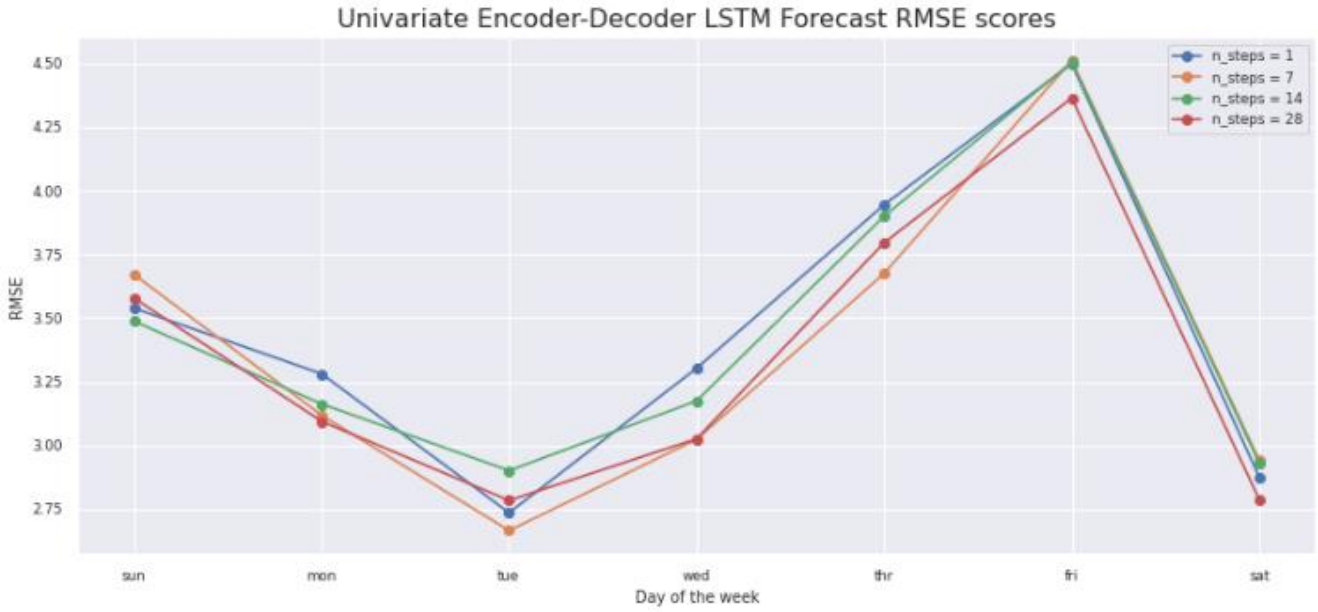


Figure 4.5 – Univariate Encoder-Decoder LSTM Forecast RMSE score plot

Univariate Encoder-Decoder LSTM RMSE scores

Num_of_steps	Average	Sun	Mon	Tue	Wed	Thur	Fri	Sat
N = 1	3.500	3.5	3.3	2.7	3.3	3.9	4.5	2.9
N = 7	3.421	3.7	3.1	2.7	3.0	3.7	4.5	2.9
N = 14	3.479	3.5	3.2	2.9	3.2	3.9	4.5	2.9
N = 28	3.391	3.6	3.1	2.8	3.0	3.8	4.4	2.8

Table 4.2 – Univariate Encoder-Decoder LSTM Forecast RMSE score table

4.3.2 – Initial Multivariate Framework

The initial architecture for the multivariate dataset was identical to that of the univariate dataset, as such it shall not be re-explained. Information on the architecture can be found in section 4.3.1.

In contrast to the univariate model, it is doubtful that the multivariate model will show immediate improvement over the baseline, although it is possible. The latter model is expected to have stronger accuracy scores as model complexity and epochs begin to increase, being the model will have more neurons available to identify features with, in addition to the ability to learn more thoroughly via a greater number of iterations.

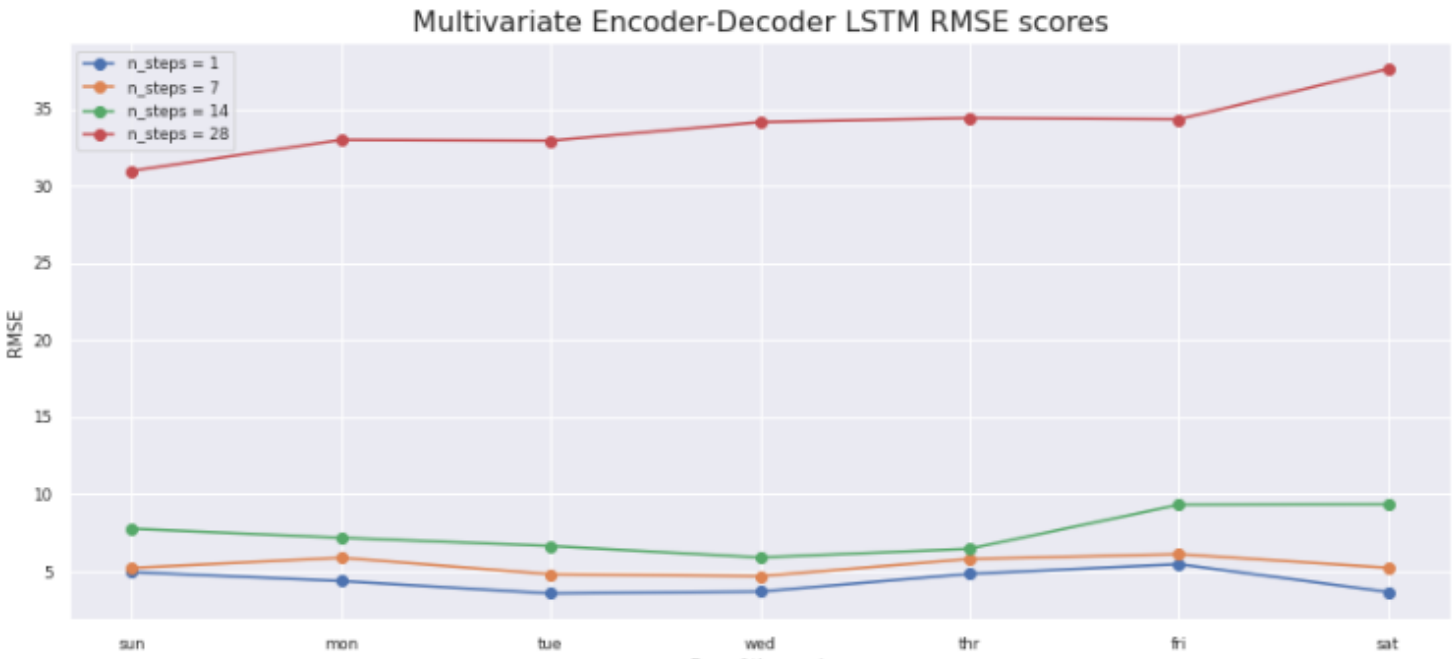


Figure 4.6 – Multivariate Encoder-Decoder LSTM Forecast RMSE score plot

Multivariate Encoder-Decoder LSTM RMSE scores

Num_of_steps	Average	Sun	Mon	Tue	Wed	Thur	Fri	Sat
N=1	4.422	5.0	4.4	3.6	3.7	4.8	5.5	3.7
N = 7	5.420	5.2	5.9	4.8	4.7	5.8	6.1	5.2
N = 14	7.627	7.8	7.2	6.6	5.9	6.5	9.3	9.3
N = 28	33.979	31.0	33.0	33.0	34.2	34.4	34.3	37.6

Table 4.3 – Multivariate Encoder-Decoder LSTM Forecast RMSE score table

As can be seen within figure 4.6 and table 4.3, the multivariate model is performing very poorly in comparison to the baseline. It is likely that 20 epochs are not enough iterations for the model's weights to adjust. Other possibilities for the lack of performance could be the complexity of the model, or poorly optimised hyperparameters.

4.3.1 – Thoughts for iteration #2

As the prior models contained within sections 4.3.1 and 4.3.2 are relatively basic, with a lot of room for iteration and improvement, inspiration shall be taken from the studies assessed within section 2.2.1. The studies of Bouktif et al. (2018), Dang et al., (2021) Jogunola et al. (2022), Yan et al. (2019), and Kim and Cho (2019) all implemented models comprising of at least a 'CNN-LSTM' based model when creating DL solutions to their respective problems. The term 'CNN-LSTM' specifically refers to a model in which a one-dimensional CNN is used to read across a sequential input, so that it may learn key features. This shall be the primary change made for the second iteration of both frameworks, alongside some separate hyperparameter changes with regards to the two models, which shall be noted within their respective sections.

4.4 – Iteration #2

This second iteration of both frameworks primarily aims to incorporate the foundations of the studies within section 2.2.2, as many of them at least incorporate the usage of what is known as a 'CNN-LSTM' architecture. As such, this shall be implemented in the second iteration of both the univariate and multivariate models.

4.4.1 – Univariate Iteration #2

The subsequent architecture constructed for the univariate dataset was an "Encoder-Decoder CNN-LSTM", this was composed of a subset of two models.

Firstly, an encoder comprised of two 'Conv1D' layers, a 'MaxPooling1D' layer, a 'Flatten' layer, and a 'RepeatVector' layer was used to read and encode the input data.

Secondly a decoder using a second 'LSTM' layer to read the encoded output was implemented, in order to make one-step predictions for the elements of the output vector.

Similar to the first iteration, the sequence was parsed through a fully-connected 'Dense' layer in order to interpret each individual time step (each day), before producing an output prediction for each step.

Notable alterations to the hyperparameters from the first iteration include:

- Epochs were increased to 50, to measure if the model was underfitting when training for 20 epochs.
- The learning rate for the Adagrad optimizer was declared to be 0.01, different from the default learning rate of 0.001. This was done in an attempt to encircle upon an 'optimal' learning rate for the framework.
- Units of the first 'Dense' layer in the fully-connected layer of the model were increased to 128. As there may be many features to learn from the data, an increase from 100 seemed appropriate, so 128 was selected as an arbitrary number.

As such, the layers and their hyperparameters for the second iteration of the univariate framework are as follows:

- Epochs = 50
- Batch Size = 16
- Conv1D (Encoder): units = 128, strides = 3
- Conv1D (Encoder): units = 64, strides = 3
- MaxPooling1D(Encoder)
- Flatten(Encoder)
- RepeatVector Layer
- LSTM(Decoder): units = 200, activation = ReLU
- Dense Layer (TimeDistributed): units = 128, activation = ReLU
- Dense Output Layer (TimeDistributed): units = 1

It was expected that this iteration of the univariate model would have a noticeable performance improvement over the previous iteration, as model complexity has increased, allowing for more thorough learning.

The RMSE scores from the second iteration of the univariate model can be seen in figure 4.7 and table 4.4. As expected, there is a lower average RMSE score for predictions on Monday,

Tuesday, Wednesday, and Saturday, in comparison to Sunday, Thursday, and Friday. However, contrary to the prediction made that performance will increase, this DL model has maintained relatively the same accuracy in relation to the preceding univariate model.

It is possible that this framework has reached a state of optimisation for the problem, however, this seems unlikely, as little attempt to optimise has been made so far. Additionally, Olamide et al. (2022) were able to achieve a much higher accuracy score than those achieved here, as such, further experimentation of framework architecture shall be carried out.

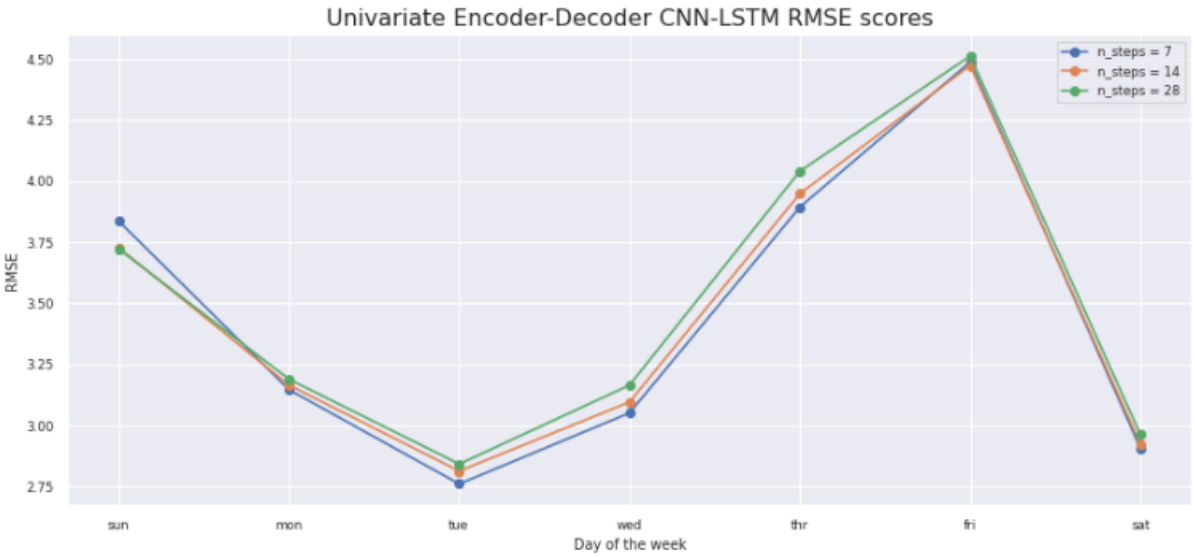


Figure 4.7 – Univariate Encoder-Decoder LSTM-CNN RMSE score plot

Univariate Encoder-Decoder CNN-LSTM RMSE scores

Num_of_steps	Average	Sun	Mon	Tue	Wed	Thur	Fri	Sat
N = 7	3.491	3.8	3.1	2.8	3.1	3.9	4.5	2.9
N = 14	3.497	3.7	3.2	2.8	3.1	3.9	4.5	2.9
N = 28	3.538	3.7	3.2	2.8	3.2	4.0	4.5	3.0

Table 4.4 – Univariate Encoder-Decoder LSTM-CNN RMSE score table

As a note upon computational cost, the run time for each of these models was about 30 seconds, suggesting there should still be room for an increase in model complexity.

4.4.2 – Multivariate Iteration #2

The subsequent architecture constructed for the multivariate dataset was also an “Encoder-Decoder CNN-LSTM”, composed of the same foundational structure as the univariate model in section 4.4.1. However, there were multiple alterations to the hyperparameters which differed to that of the univariate model. These alterations to the hyperparameters from the first iteration include:

- Epochs were increased to 100, due to there being more data to work with, it is more likely that 20, or even 50 epochs, is not enough training time for this framework.
- The Conv1D layers had their units increased to 512 and 256, respectively. This was done with the intention of allowing more room for the model to learn features of the complex data, in the hopes of improving accuracy.

In contrast, the following hyperparameters were altered from the first multivariate framework iteration, but to the same values as stated in the second univariate framework iteration:

- Units of the first ‘Dense’ layer in the fully-connected layer of the model were also increased to 128.
- The learning rate for the Adagrad optimizer was declared to be 0.01, different from the default learning rate of 0.001, in an attempt to encircle upon an ‘optimal’ learning rate for the framework.

As such, the layers and their hyperparameters for the second iteration of the multivariate framework are as follows:

- Epochs = 100
- Batch Size = 16

- Conv1D (Encoder): units = 256, strides = 3
- Conv1D (Encoder): units = 128, strides = 3
- MaxPooling1D(Encoder)
- Flatten (Encoder)
- RepeatVector Layer
- LSTM(Decoder): units = 200, activation = ReLU
- Dense Layer (TimeDistributed): units = 128, activation = ReLU
- Dense Output Layer (TimeDistributed): units = 1

The expectation for the second iteration of the multivariate framework was that there would be a slight improvement in model performance, as the increased model complexity should aid in the identification of complex patterns in the features. However, it is also expected there will still be much room for improvement due to the complexity of the multivariate data.

The RMSE scores from the second iteration of the multivariate model can be seen in figure 4.8 and table 4.5. Whilst there is some increase in model performance, the RMSE metrics are still below the baseline, most noticeably when trying to forecast for 28 days.

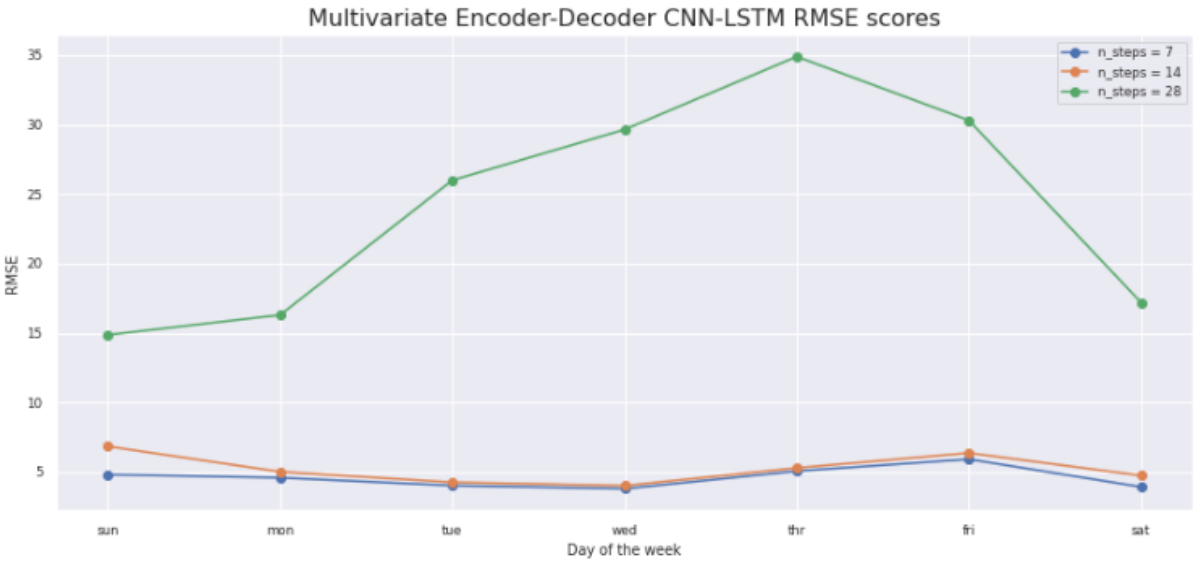


Figure 4.8– Multivariate Encoder-Decoder LSTM-CNN RMSE score plot

Multivariate Encoder-Decoder CNN-LSTM RMSE scores

Num_of_steps	Average	Sun	Mon	Tue	Wed	Thur	Fri	Sat
N = 7	4.638	4.8	4.6	4.0	3.8	5.1	5.9	3.9
N = 14	5.302	6.8	5.0	4.3	4.0	5.3	6.4	4.7
N = 28	25.277	14.9	16.3	26.0	29.6	34.9	30.3	17.2

Table 4.5 – Multivariate Encoder-Decoder LSTM-CNN RMSE score table

The above models took an average of 1 minute and 35 seconds each to run, suggesting that there is still likely room for improvement via an increase of complexity in the framework. However, if the subsequent iteration of this model shows an insignificant increase in accuracy, no further experimentation shall be carried out upon this multivariate dataset, as it is struggling to beat the baseline, and is already more costly than the stronger univariate framework.

4.4.3 - Thoughts for iteration #3

An extension of the ‘CNN-LSTM’ model is a ‘ConvLSTM’ model, which differs from the currently implemented ‘CNN-LSTM’ architecture, as it uses the convolutions of the CNN as a direct input to LSTM units, potentially allowing for deeper salient features to be learned.

It is possible that the ‘Conv-LSTM’ model may be very computationally expensive in contrast to the ‘CNN-LSTM’, as it is simply a combination of a convolution and an LSTM output, however, this shall be quantitively assessed based upon the results of the third iteration of each framework.

4.5 - Iteration #3

This third iteration of both frameworks primarily aims to incorporate the foundations of the studies within section 2.2.2, as many of them at least incorporate the usage of what is known as a ‘CNN-LSTM’ architecture. As such, this shall be implemented in the second iteration of both the univariate and multivariate models.

For the third iteration of both frameworks, a variation of the ‘CNN-LSTM’ architecture is to be incorporated, the ‘ConvLSTM’. Brownlee (2019) recommends this as an extension to the former model type, and as such it shall be experimented with in the next iteration, in order to understand the effect it has on model performance and computation time.

It should be noted that within this section, models were only constructed to forecast a 14-day time period, using the prior 14 days. This decision was made as the scope of the project lined out within section 1.2 aims to forecast energy consumption over at least a 14-day period. As such, due to 28-day forecasts being generally more inaccurate, a 14-day forecast has been selected in the interest of achieving stronger accuracy metrics.

4.5.1 - Univariate Iteration #3

Like the prior architectures, the ‘ConvLSTM’ model comprises of a subset of two models.

Firstly, an encoder comprised of two ‘ConvLSTM2D’ layers, a ‘Flatten’ layer, and a ‘RepeatVector’ layer was used to read and encode the input data.

Secondly a decoder using a standard ‘LSTM’ layer to read the encoded output was implemented, in order to make one-step predictions for the elements of the output vector.

Finally, the sequence was again parsed through a fully-connected ‘Dense’ layer in order to interpret each individual time step (each day), before producing an output prediction for each step.

No notable alterations were made to generic hyperparameters such as epochs, so that any difference in model performance can be attributed solely to the change in the hybrid architecture.

As such, the layers and their hyperparameters for the second iteration of the univariate framework are as follows:

- Epochs = 50
- Batch Size = 16

- Conv2D (Encoder): units = 128, strides = 1,3
- Conv2D (Encoder): units = 64, strides = 1,3
- Flatten (Encoder)
- RepeatVector Layer
- LSTM(Decoder): units = 200, activation = ReLU
- Dense Layer (TimeDistributed): units = 128, activation = ReLU
- Dense Output Layer (TimeDistributed): units = 1

Due to the nature of the ConvLSTM, in which convolutions are fed directly into LSTM units as inputs, it was expected that this iteration of the univariate model would again demonstrate an increase in performance, although possibly at high computational cost due to the increase in calculations the model must perform.

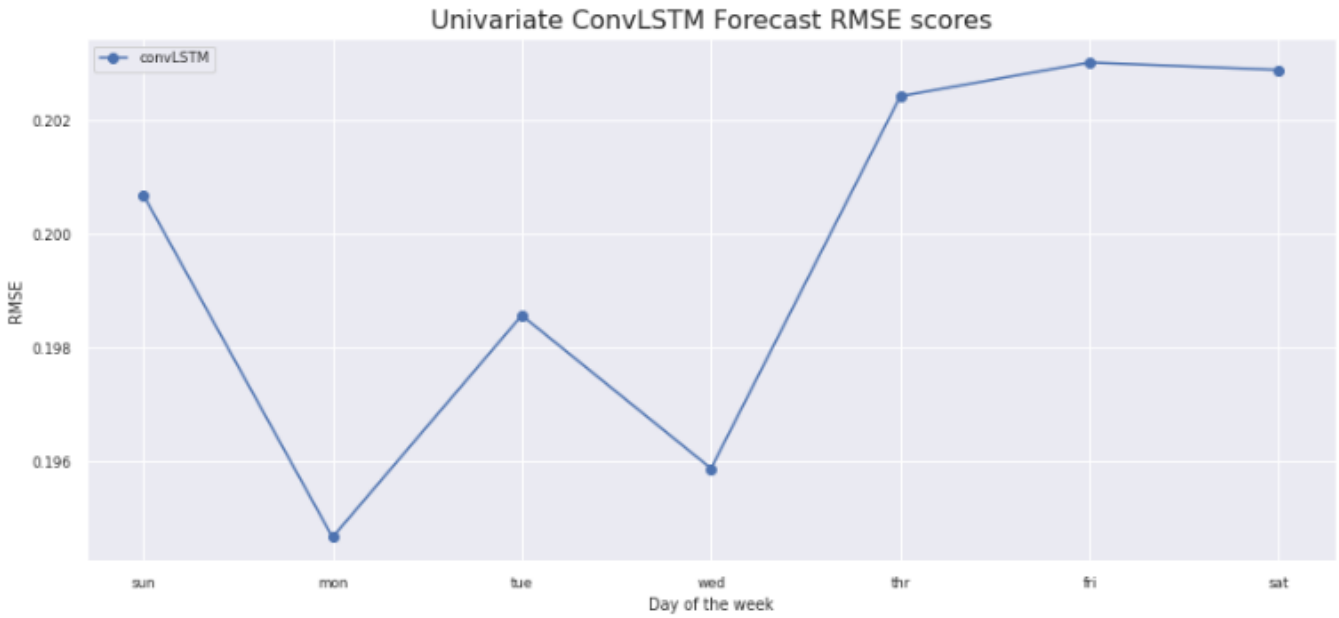


Figure 4.9 – Univariate Encoder-Decoder ConvLSTM RMSE score plot

The RMSE scores from this third iteration of the univariate model can be seen in figure 4.9 and table 4.6. The increase in performance from the prior framework is drastic, as the average RMSE has improved to 0.2, outperforming every prior model by a large amount. Although, as predicted, this increase in performance comes at a much higher computational cost, with the model taking roughly 5 minutes to train, even with a low number of epochs.

Univariate Encoder-Decoder ConvLSTM RMSE scores

n_input	Average	Sun	Mon	Tue	Wed	Thur	Fri	Sat
Steps = 2, len = 7	0.200	0.2	0.2	0.2	0.2	0.2	0.2	0.2

Table 4.6 – Univariate Encoder-Decoder ConvLSTM RMSE score table

4.5.2 - Multivariate iteration #3

Similarly to the third univariate framework iteration, there is an encoder, however it is comprised of four ‘ConvLSTM2D’ layers, a ‘Flatten’ layer, and a ‘RepeatVector’ layer was used to read and encode the input data.

Secondly a decoder using a standard ‘LSTM’ layer to read the encoded output was implemented, in order to make one-step predictions for the elements of the output vector.

Finally, the sequence was again parsed through a fully-connected ‘Dense’ layer in order to interpret each individual time step (each day), before producing an output prediction for each step.

Epochs were increased to 100, with the intention that the model may have an appropriate amount of training time to optimise as the ConvLSTM model framework is more complex than the previous CNN-LSTM framework.

As such, the layers and their hyperparameters for the second iteration of the univariate framework are as follows:

- Epochs = 100
- Batch Size = 16
- Conv2D (Encoder): units = 256, strides = 1,3
- Conv2D (Encoder): units = 128, strides = 1,3
- Conv2D (Encoder): units = 64, strides = 1,3
- Conv2D (Encoder): units = 32, strides = 1,3

- Flatten (Encoder)
- RepeatVector Layer
- LSTM(Decoder): units = 200, activation = ReLU
- Dense Layer (TimeDistributed): units = 128, activation = ReLU
- Dense Output Layer (TimeDistributed): units = 1

Whilst the RMSE scores seen in figure 4.10 and table 4.7 are a further increase upon the previous iterations of the multivariate model, the model is still struggling to beat the naïve baseline by a significant amount. This, when paired with the fact that computational cost of this model increased drastically (a 300% increase in training time over the prior multivariate iteration), suggests that whilst a possible solution found through this may be more accurate when fully optimised, it will very likely be far too computationally expensive to run on an embedded device. As such, the model would not fulfil the initial objectives of this dissertation project. Following this conclusion, the decision was made to not iterate upon this multivariate framework further.

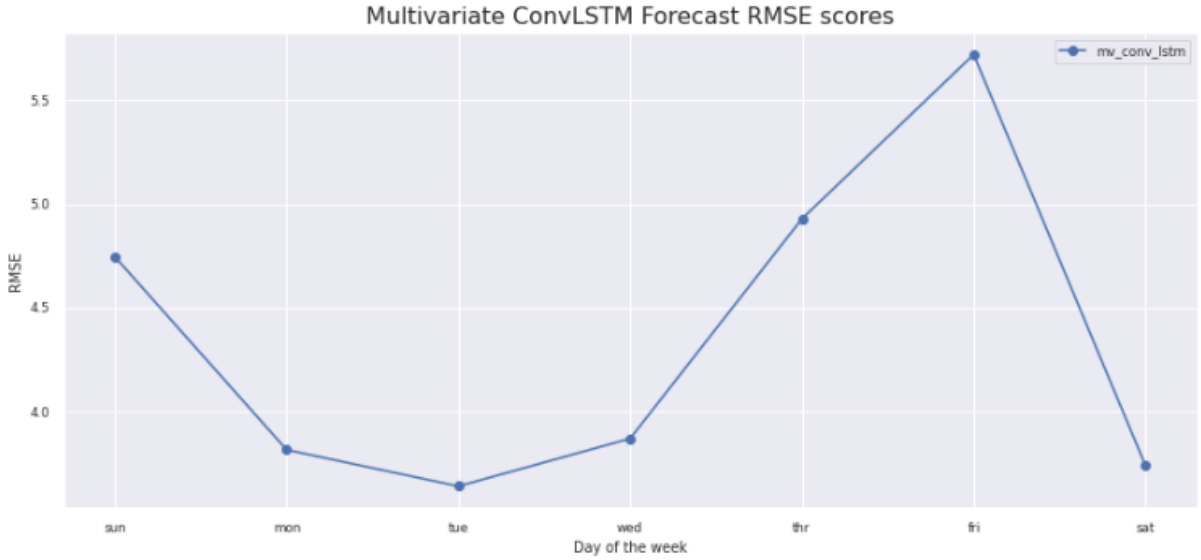


Figure 4.10 – Multivariate Encoder-Decoder ConvLSTM RMSE score plot

Univariate Encoder-Decoder ConvLSTM RMSE scores

n_input	Average	Sun	Mon	Tue	Wed	Thur	Fri	Sat
Steps = 2, len = 7	0.200	0.2	0.2	0.2	0.2	0.2	0.2	0.2

Table 4.7 – Multivariate Encoder-Decoder ConvLSTM RMSE score table

4.5.3 - Thoughts for iteration #4

Moving forwards, the univariate model shall be trained and tested upon data across the same time period from other households in the energy consumption dataset, in order to see if the model is as effective at learning trends in other time-series energy data, as opposed to just data from the single household used so far.

4.6 - Model performance on other households

In order to test how the selected model performs on energy data from other households, the selected model shall be trained and tested on data from other households from the same dataset, in order to see if it maintains the same level of effectiveness.

As can be observed in figure 4.11, in addition to table 4.8, the model demonstrates strong performance when trained and tested upon individual data from a variety of other households from the initial dataset, with RMSE scores ranging from 0.05 to 0.25.

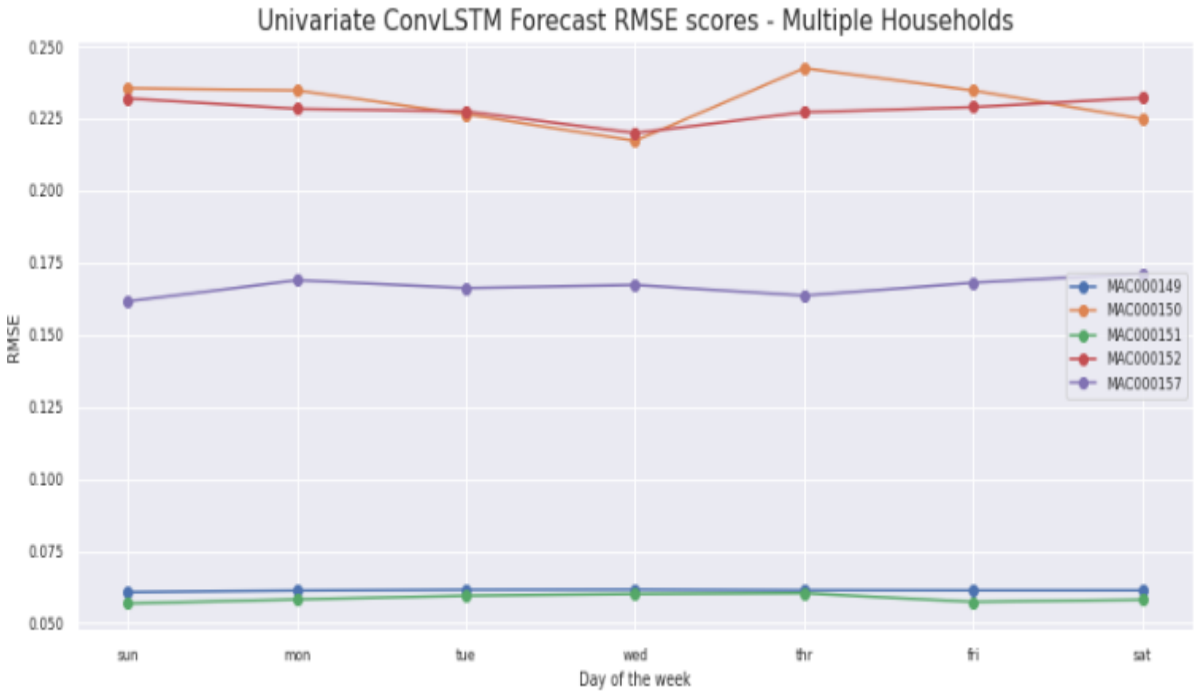


Figure 4.11 – Univariate Encoder-Decoder ConvLSTM RMSE score plot – Multiple Households

Univariate Encoder-Decoder ConvLSTM RMSE scores – Multiple Households

Household	Average	Sun	Mon	Tue	Wed	Thur	Fri	Sat
MAC000149	0.063	0.1	0.1	0.1	0.1	0.1	0.1	0.1
MAC000150	0.231	0.2	0.2	0.2	0.2	0.2	0.2	0.2
MAC000151	0.058	0.1	0.1	0.1	0.1	0.1	0.1	0.1
MAC000152	0.228	0.2	0.2	0.2	0.2	0.2	0.2	0.2
MAC000157	0.167	0.2	0.2	0.2	0.2	0.2	0.2	0.2

Table 4.8 – Univariate Encoder-Decoder ConvLSTM RMSE score for multiple households table

As such, this shall conclude the experimental studies section of this dissertation, as a DL framework has been created that may be trained and tested upon data from a variety of individual households, that performs strongly in comparison to the naïve baseline created in section 4.2. Furthermore, the accuracy metrics achieved for these models are comparable to that of some of the studies discussed within section 2.2.1, such as Yan et al. (2019), suggesting that there is little room further optimisation.

5.0 – Conclusions & Recommendations

In this section, final conclusions shall be made as to the success of this dissertation project with regards to its initial aims and objectives, and recommendations shall be given as to how this study may wish to be taken further.

5.1 – Evaluation

This section shall evaluate the success of this dissertation, via direct reference to the aims and objectives outlined within sections 1.2 and 1.3 respectively.

5.1.1 - Aims of the Project

The aims of this project were as follows:

- Build a robust ML/DL framework that is able to accurately predict energy consumption for a variety of households.
- Explore the effect of non-energy data being used to aid in the prediction of energy consumption (such as weather).
- Keep the computational cost of the framework appropriate, so that the model may be potentially deployed on an embedded device.

The first aim of this paper has been confidently met, as a DL framework has been created that can accurately predict energy consumption for individual households when trained and tested on their individual data.

The second aim of this paper has also been undoubtedly met, as the usage of weather data in tandem with energy consumption data was explored when looking to forecast solely energy consumption.

The 5-minute training time of the final model indicates appropriate computational cost when looking to enable the framework to be potentially deployed on an embedded smart meter device. As such, the final aim of the paper also has been firmly met.

5.1.2 – Objectives of the dissertation

1. *"Conduct a literature review"*

This objective has been met, as the literature review mentioned is section 2 of this dissertation.

2. *"Gather and prepare data necessary for training and testing a deep learning/machine learning framework."*

This objective has been met, as data collection methods have been included in section 3.3 of this dissertation, and data preparation methodology has been discussed in section 3.4.

3. *"Build an initial framework, training and testing the prepared dataset on it for comparison."*

This objective has been met, as can be seen within section 4.3 of this dissertation.

4. *"Carry out a comparative analysis of my framework versus other similar frameworks."*

Sections 4.3, 4.4, and 4.5 involved the discussion of other studies and how their frameworks compared to the current iterations of the framework in this paper, as such, this objective is considered met.

5. *"Conduct an experimental study in order to attempt to optimise the framework."*

This objective has been met, as chapter 4 of this dissertation contains the approach to framework construction and experimentation, with the end goal of optimising the constructed framework.

6. *“Write a complete dissertation report documenting the project.”*

The dissertation document within which this objective is written is empirical proof that this objective has been met.

7. *“Produce a video to present the project for the viva.”*

Guidelines on the viva have been updated since creation of this objective, and as such this objective is no longer relevant, meaning it does not need to be met.

5.4 – Personal Reflections

On the whole, this project was an incredible experience. It provided a means to begin to explore deeper into the world of deep learning and time-series forecasting. Additionally, it provided insight into how these techniques may be applied within a real-world setting, to a relevant modern issue. There were many notable challenges throughout every step of the project, although an increasing understanding of the energy domain, in addition to an increasing awareness of the potential application of deep learning techniques to time-series forecasting problems allowed for strong technical growth.

The most difficult part of the dissertation was the configuration of the multivariate data when looking to process it through the ConvLSTM framework, as it required a specific 5-dimensional format which in turn required multiple arrays to be indexed to retrieve the relevant data.

If I was to redo this project, I would look to experiment with time-series forecasting methods earlier on, in addition to perhaps taking some complimentary courses that can aid with understanding of the domain. The knowledge required to be able to navigate a time-series forecasting problem can be extremely technical. And, as I had not previously explored approaching this type of deep learning, it would have been useful to have more formal

guidance in the form of a syllabus, so that a methodical approach to learning how to forecast time-series data may have been adopted.

5.5 - Future Recommendations

In order to further improve upon the robustness and versatility of the framework, it would be recommended to explore techniques such as hierarchical forecasting, which Khanuja et al. (2021) recommend the implementation of when working with time series data that follows a clear hierarchical aggregation structure. This would be relevant to the data used, as all the energy data is from London, so the households would be underneath London in the hierarchy.

Additional exploration may also be made with regards to developing a complex enough framework to be able to produce accurate predictions when applied to the multivariate dataset. The computational cost of the model would likely be very expensive, provided the trends in training times continue as they were doing in this study. As such it is recommended this is explored if looking to approach a similar problem to this dissertation, but with an online model solution in mind.

A final point of recommendation would be to thoroughly investigate the hardware capabilities of current commercial smart meters, so that a relevant simulation may be selected in order to test the model built within this paper upon, to then in turn gain a more accurate understanding of the computational cost of it.

Bibliography

Bouktif, S., Fiaz, A., Ouni, A. and Serhani, M. A. (2018) 'Optimal Deep Learning LSTM Model for Electric Load Forecasting using Feature Selection and Genetic Algorithm: Comparison with Machine Learning Approaches.' *Energies*, 11(7) pp. 1636–1636.

BP (2022) *Second quarter 2022 results*.

Brownlee, J. (2019) *Deep Learning for Time Series Forecasting*. Martin, S., Koshy, A., and Cheremskoy, A. (eds). 1.6.

Bulb (2022) *What is a smart meter and how does it work?* https://help.bulb.co.uk/hc/en-us/articles/360005132971-What-is-a-smart-meter-and-how-does-it-work-#:~:text=A%20smart%20meter%20measures%20your,and%20electricity%20meters%20any%20more. [Online] [Accessed on 23rd August 2022] https://help.bulb.co.uk/hc/en-us/articles/360005132971-What-is-a-smart-meter-and-how-does-it-work-#:~:text=A smart meter measures your,and electricity meters any more.

Chollet, F. (2021) *Deep Learning with Python*. 2nd ed., New York: Manning Publications Co.

Cornwall Insight (2022) *Cornwall Insight comments on its January Default Tariff Cap forecast rising to £3,000 for a typical user*. https://www.cornwall-insight.com/press/cornwall-insight-comments-on-its-january-default-tariff-cap-forecast-rising-to-3000-for-a-typical-user/.

Dang, C. N., Moreno-García, M. N. and De La Prieta, F. (2021) 'Hybrid Deep Learning Models for Sentiment Analysis.' *Complexity*, 2021, September.

FAMILI, A., SHEN, W., WEBER, R. and SIMOUDIS, E. (1997) 'Data preprocessing and intelligent data analysis.' *Intelligent Data Analysis*. No longer published by Elsevier, 1(1–4) pp. 3–23.

Fekri, M. N., Patel, H., Grolinger, K. and Sharma, V. (2021) 'Deep learning for load forecasting with smart meter data: Online Adaptive Recurrent Neural Network.' *Applied*

Energy. Elsevier, 282, January, p. 116177.

Fortuin, V., Baranchuk, D., Rätsch, G. and Mandt, S. (2020) 'GP-VAE: Deep Probabilistic Multivariate Time Series Imputation.' *In Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS) 2020*.

Gajowniczek, K. and Zabkowski, T. (2014) 'Short Term Electricity Forecasting Using Individual Smart Meter Data.' *Procedia Computer Science*. Elsevier, 35(C) pp. 589–597.

Genovese, S. (2020) 'Artificial Intelligence: A Guide for Thinking Humans.' *ORDO*. Pelican Books, 71(1) pp. 444–449.

GOV.UK (2018) *Data Protection Act 2018*.

Harris, B., Kerai, G. and Vizard, T. (2022) *Impact of increased cost of living on adults across Great Britain: November 2021 to March 2022*. https://www.ons.gov.uk/peoplepopulationandcommunity/personalandhouseholdfinances/expenditure/articles/impactofincreasedcostoflivingonadultsacrossgreatbritain/november2021tomarch2022.

Jean Michel, D. (2022) *London Smart Meter Analysis of Half-hourly Data*. [Online] [Accessed on 25th September 2022] https://www.kaggle.com/datasets/jeanmidev/smart-meters-in-london.

Jogunola, O., Adebisi, B., Van Hoang, K., Tsado, Y., Popoola, S. I., Hammoudeh, M. and Nawaz, R. (2022) 'CBLSTM-AE: A Hybrid Deep Learning Framework for Predicting Energy Consumption.' *Energies*. MDPI, 15(3).

Khanuja, M., Sabir, F. and Gupta, N. (2021) *Hierarchical Forecasting using Amazon SageMaker*. AWS Machine Learning Blog.

Kim, T. Y. and Cho, S. B. (2019) 'Predicting residential energy consumption using CNN-LSTM neural networks.' *Energy*. Pergamon, 182, September, pp. 72–81.

Mode, G. R. and Hoque, K. A. (2020) *Adversarial Examples in Deep Learning for Multivariate Time Series Regression*.

Mostyn, M. (2021) *SMETS1 and SMETS 2: Everything you need to know about different types of smart meters*.

Neto, P. S. G. de M., Oliveira, J. F. L. de, Bassetto, P., Siqueira, H. V., Barbosa, L., Alves, E. P., Marinho, M. H. N., Rissi, G. F. and Li, F. (2021) 'Energy Consumption Forecasting for Smart Meters Using Extreme Learning Machine Ensemble.' *Sensors*, 21(8096) pp. 1–3.

Notton, G. and Voyant, C. (2018) 'Forecasting of Intermittent Solar Energy Resource.' *Advances in Renewable Energies and Power Technologies*. Elsevier, 1, January, pp. 77–114.

NXP (2022) *Preview of Essentials of Edge Computing*.

Office for National Statistics (2022) *Energy prices and their effect on households*. https://www.ons.gov.uk/economy/inflationandpriceindices/articles/energypricesandtheireffectonhouseholds/2022-02-01.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B. and Swami, A. (2016) *The Limitations of Deep Learning in Adversarial Settings*. Saarbrucken.

Pratama, I., Permanasari, A. E. and Indrayani, R. (2016) 'A review of missing values handling methods on time-series data.' *In 2016 International Conference on Information Technology Systems and Innovation (ICITISI)*. Yogyakarta: Universitas Gadjah Mada.

Ras, G., Xie, N., Gerven, M. van and Doran, D. (2021) *Explainable Deep Learning: A Field Guide for the Uninitiated*.

Rose, K., Eldridge, S. and Chapin, L. (2015) 'The Internet of Things (IoT): An Overview.' *Int. Journal of Engineering Research and Applications*, 5(12) pp. 71–82.

Sirojan, T., Lu, S., Phung, B. T. and Ambikairajah, E. (2019) 'Embedded Edge Computing for

Real-time Smart Meter Data Analytics.' *In SEST 2019 - 2nd International Conference on Smart Energy Systems and Technologies*. Porto.

Stewart, I. and Bolton, P. (2022) *Domestic Energy Prices*. House of Commons Library. [Online] [Accessed on 23rd August 2022] https://commonslibrary.parliament.uk/research-briefings/cbp-9491/.

Sukumar, P., Robert, L. and Yuvaraj, S. (2016) *Review on Modern Data Preprocessing Techniques in Web Usage Mining (WUM)*. Coimbatore.

UK Power Networks (2022) *SmartMeter Energy Consumption Data in London Households*.

Yan, K., Li, W., Ji, Z., Qi, M. and Du, Y. (2019) 'A Hybrid LSTM Neural Network for Energy Consumption Forecasting of Individual Households.' *IEEE Access*, 7, September, pp. 157633–157642.

Appendices

Appendix A – Python implementation of final ConvLSTM model

```
# train the model
def build_model(train, n_steps, n_length, n_input):
  # prepares the training data
  train_x, train_y = to_supervised(train, n_input)
  # defines parameters for the model
  verbose, epochs, batch_size = 0, 50, 16
  n_features, n_outputs = train_x.shape[2], train_y.shape[1]
  # reshapes x into subsequences [samples, timesteps, rows, cols, channels]
  train_x = train_x.reshape((train_x.shape[0], n_steps, 1, n_length, n_features))
  # reshapes the output into an appropriate format: [samples, timesteps, features]
  train_y = train_y.reshape((train_y.shape[0], train_y.shape[1], 1))
  # defines the model
  model = Sequential()
  model.add(ConvLSTM2D(128, (1,3), activation='relu', return_sequences = True, input_shape=(n_steps, 1, n_length, n_features)))
  model.add(ConvLSTM2D(64, (1,3), activation='relu'))
  model.add(Flatten())
  model.add(RepeatVector(n_outputs))
  model.add(LSTM(200, activation='relu', return_sequences=True))
  model.add(TimeDistributed(Dense(128, activation='relu')))
  model.add(TimeDistributed(Dense(1)))
  model.compile(loss='mse', optimizer= tf.keras.optimizers.Adagrad(learning_rate = 0.01))
  # fit network
  model.fit(train_x, train_y, epochs=epochs, batch_size=batch_size, verbose=verbose)
  return model
```

Appendix B – Terms of Reference

Terms of Reference

Energy Consumption Forecasting Using Hybrid Deep Learning Frameworks

Aim

The aim of this project is to build a robust hybrid deep learning framework that is able to accurately predict energy consumption across various building types, weather types, and locations. This will be done whilst looking to minimise computational cost, in an attempt to accommodate for the low computational capabilities of energy-reading devices, such as smart meters and smart boxes.

Learning Outcomes

• Choose, customise, and integrate core techniques to build sophisticated solutions for real-world Data Science problems.

• Process and analyse, effectively and efficiently, data of varying scales and from heterogeneous sources, formats, and systems, using a range of suitable languages, tools, and environments.

• Build data science products with good software practices such as in code reuse, separation of concerns, modularity, testing, and documentation.

Project Background

With the recent rise of energy prices within the UK, it is more imperative than ever to provide both consumers and providers alike with accurate energy readings. Providers must avoid undercharging so that they may still profit when accounting for current fuel prices, as wholesale gas is currently four times more expensive than it was at the start of 2021. Whilst, in contrast, consumers must avoid overpaying, as the cost of living has risen by 4.8% between December 2020 and December 2021, with 79% of respondents to the latest 'Opinions and Lifestyle Survey' saying gas and electricity are a cause of this(Statistics, 2022).

According to Jogunola et al. (Olamide Jogunola, 2022), multiple deep learning models have been proposed for load forecasting, and have demonstrated a degree of improvement in accuracy over traditional methods. However, she also states that the computation time of these models is "mostly ignored", meaning that successful deployment would be difficult on the memory-constrained devices that would be looking to implement them (smart meters, smart boxes, etc.).

In an attempt to balance the aforementioned relationship between consumers and providers, this project shall seek to further the work of Jogunola et al., in looking to create a hybrid deep learning framework that is able to accurately predict energy consumption across a wide variety of contexts whilst maintaining an appropriate level of computational expensiveness for devices such as smart meters and smart boxes.

Objectives

1. Conduct a literature review

2. Gather and prepare data necessary for training and testing the deep learning framework.

3. Build an initial deep learning framework, training and testing the prepared dataset on it for comparison.

4. Carry out a comparative analysis of my framework versus other similar frameworks, making changes to the initial framework if deemed significant for improving performance.

5. Conduct some form of ablation study/iterative approach in order to attempt to optimise the deep learning framework further.

6. Write a complete dissertation report documenting the project.

7. Produce a video to present the project for the viva voce.

Potential Problems

Obtaining a dataset that is available to the public domain for use with this project. If unable to find a public dataset, then a dataset will need to be sourced, and permission obtained to be able to use and publish the dataset alongside the project.

In order to align with modern GDPR practises, any and all identifiable, personal information must (and therefore shall) be removed from the dataset (Union, 2016).

There may be difficulty when attempting to implement the framework into a real-world setting. A potential fix is hard to define ahead of the project, as there are many possible issues as to why this would be. Any attempts to fix said issues shall be mentioned, and the possible implications of real-world application of the final framework shall be stated if unable to implement it.

It may be difficult to measure what is considered an 'appropriate' amount of computational cost. In this case estimations shall be made using knowledge of the hardware inside of devices such as smart meters and smart boxes.

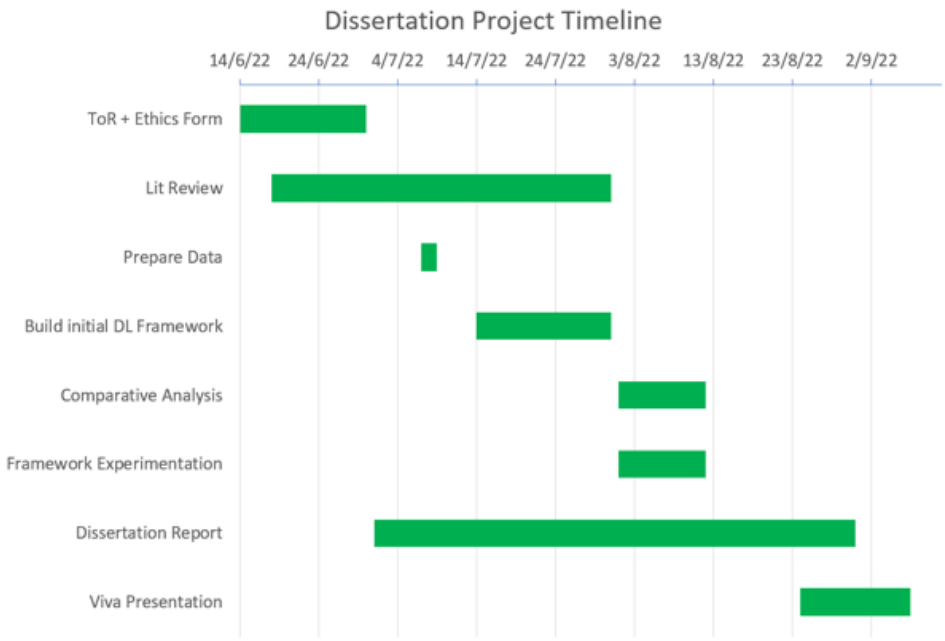
Required Resources

A Python IDE for constructing the DL framework (e.g. Google Colab).

Python Libraries (Tensorflow, Keras, Numpy, Pandas).

Generic Desktop/Laptop capable of running the IDE. Possibly a more powerful machine depending on computational constraints and/or local runtimes.

Schedule


Dissertation Project Timeline
14/6/22
24/6/22
4/7/22
14/7/22
24/7/22
3/8/22
13/8/22
23/8/22
2/9/22
ToR + Ethics Form
Lit Review
Prepare Data
Build initial DL Framework
Comparative Analysis
Framework Experimentation
Dissertation Report
Viva Presentation

References

Olamide Jogunola, B. A. (2022). CBLSTM-AE: A Hybrid Deep Learning Framework for Predicting Energy Consumption. *Energies*, 810.

Statistics, O. f. (2022, 07 03). *Energy Prices and their effect on households*. Retrieved from Office for National Statistics: https://www.ons.gov.uk/economy/inflationandpriceindices/articles/energypricesandtheireffectonhouseholds/2022-02-01

Union, T. E. (2016). Regulations. *Official Journal of the European Union*.