Deep Learning Techniques for Household Load Forecasting

A thesis submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

Electrical Engineering
(Power and Automation)

by

Aditya Arora 2014EE30629
Deepak Singla 2014EE30506

Under the guidance of

Dr Nilanjan Senroy

Department of Electrical Engineering,
Indian Institute of Technology Delhi
November 2017

Certificate

This is to certify that the thesis titled **Deep Learning Techniques for Household Load Forecasting** being submitted by **Aditya Arora and Deepak Singla** for the award of **Bachelor of Technology** in **Electrical Engineering (Power and Automation)** is a record of bona fide work carried out by them under my guidance and supervision at the **Department of Electrical Engineering**. The work presented in this thesis has not been submitted elsewhere either in part or full, for the award of any other degree or diploma.

Dr Nilanjan Senroy
Department of Electrical Engineering
Indian Institute of Technology, Delhi

Abstract

In the years to come, the electric demand is predicted to rise steeply owing to a variety of reasons and the electric grid would need to evolve in order to meet these demands efficiently and reliably. One of the key pieces of enabling this evolution is accurately forecasting the load by a wide variety of consumers. In our work, we evaluate the possibility of using recent techniques in Artificial Intelligence so as to forecast consumer loads more accurately than by conventional techniques like Artificial Networks.

Acknowledgments

We would like to thank Professor Senroy for his ideas, support and guidance without which this work would not be possible. We are also grateful to the committee who evaluated our work in the past - Professor Bijwe, Professor Abhyankar and Professor Panigrahi - whose feedback shaped our work in these past few months.

Aditya Arora
Deepak Singla

Contents

List of Figures

List of Tables

Chapter 1

Motivation

The power demands in the near future are predicted to be tremendous due to increasing population, industrialization and the settlement of a large number of people in the urban areas among other reasons. Thus, this tremendous demand of power would exact several demands from the electric grid: that the grid should be flexible enough to meet the varying load demands, be resilient in unforeseen situations and provide an uninterrupted supply of high-quality power.

Moreover, the grid should remain efficient - both in terms of cost and losses - in the face of ever-increasing demand. Further, load forecasting plays an important part in optimizing power flow and security analysis and hence in all cases it is imperative to forecast load accurately at a wide variety of forecast horizons - i.e. how far into the future we would be predicting.

One of the most popular models in current literature is the Artificial Neural Network. This model has attained quite a lot of success for its accuracy and its speed. However, the artificial neural network suffers from a fundamental flaw: it assumes that all its inputs are independent with respect to one another i.e. they do not take into account the sequentiality present in the load data.

Newer techniques in the field of Artificial Intelligence have recently been developed out of Artificial Neural Networks that resolve this issue and take into account this sequentiality - that the inputs are in an ordered sequence. They have attained success in domains such as Natural Language Processing (ex- translating text from one language to another), time series forecasting, etc and hence could potentially be suitable for our application.

Therefore we focus on assessing the utility of using these newer techniques in artificial intelligence for forecasting the load in the future. More specifically we compare a variety of Deep Learning Architectures such as **Convolutional**

Neural Networks and Sequence to Sequence Networks with and without Attention with conventional **Artificial Neural Networks**.

In this thesis, we shall proceed with literature survey to describe how the load forecasting community has used techniques in machine learning and artificial intelligence to great effect. Following that, we shall describe the working of some common models like Artificial Neural Networks as well as more recent Sequence to Sequence models and Convolutional Neural Networks.

We will then describe the problem statement we have been working on and the results we have obtained. Finally, we shall conclude with some key observations regarding our results.

Chapter 2

Literature Survey

2.1 Introduction

In the previous chapter, we had established the need for forecasting consumer load so as to build a more reliable, resilient and efficient electric grid. Now we shall go about describing the various methods for load forecasting as in the literature we surveyed. We shall approach this from three different angles:

- Forecast Horizon: How far in the future are we forecasting?
- Aim of the Forecast: What application is the model to be used for?
- Model Complexity: How complex is the model being used?

One thing that is essential to remember is that these divisions are independent of one another and define entirely different aspects of the problem. As an example, there can exist a non-linear model that provides the load profile for the next day for day-to-day operations of a power station.

2.2 Forecast Horizon

Making any kind of decision in the electric supply sector, requires an accurate forecast of consumer demand and all decision makers face a number of complex problems on various time-scales as well as at different levels in the hierarchy of the power supply system. As an example, not only must the power be supplied for immediate demands, enough fuel must also be allocated to supply consumer needs in the coming weeks. Moreover suppose that certain areas are expected to become industrial hubs with large demand, which would require a large amount of power and hence a new station may itself need to be built. Hence there is a need for consumer demand forecasts at different levels of the supply hierarchy and at different time-scales as well.

2.2.1 Introduction

According to Hahn et. al. [3], load forecasting can be broadly categorized into four categories based on the forecast horizon. In brief, they can be summarized by the order of time-scales: very short (seconds, minutes to a few hours), short (a day to a week), medium (several weeks to several months) and long(more than a year).

Each of them has their own area of application and uses different kinds of data and we shall go about discussing each of them, describing where they are used and providing a few studies for each.

2.2.2 Very Short Term Forecasts

Very short-term forecasts are aimed at producing forecasts at time-scales of seconds to minutes to even a few hours. Providing very accurate forecasts for the next few hours in quite essential for power companies to control load flow as such forecasts have an effect on planning for an optimal amount of generation at the hourly level.

Some examples of studies that focus on this are: Charytoniuk et. al. [4] use an ensemble of Artificial Neural Networks to calculate load variations over the next hour while Yang [5] uses support vector machines and a tree-based algorithm.

2.2.3 Short Term Forecasts

Short term load forecasts aim to provide the load for up to a week ahead and hence play a crucial role in the day-to-day operation of power systems, helping in fuel adjustments and security analysis among other things.

As an example, we have Hippert et. al. [6] which is an impressive survey of short-term load forecasting papers and Kyriakides et. al. [7] which is yet another discussion of the various methods for the same.

2.2.4 Medium Term Forecasts

Medium-term load forecasts have a forecast horizon that varies from one week to one year. Such forecasts help companies negotiate with other companies on things like the cost and amount of fuel, the amount of power they can deliver and at what rate and so on.

Examples of studies include Doveh et. al.[8] which uses a wide variety of parameters as proxies for weather, financial variables and social demographics so as to forecast for nearly a year ahead while Bruhns et. al. uses a non-linear regression model to better factor in the seasonality of the data.

2.2.5 Long Term Forecasts

Long-term load forecasts have a forecast horizon that is of the order of several years. Such models have to account for weather, seasonal effects, and even socioeconomic factors as well as anticipate technological developments. Such models are typically used to plan new utilities for the future.

As examples, consider Kandil et. al. [9] that uses a knowledge-based expert system based on static (load history, weather, socio-economic parameters and so on) and dynamic (losses in the system, measurement and instrument error) parameters or Zhang et. al. [10] that uses an artificial neural network for regression to forecast up to five years in the future.

2.3 Aim of the Forecast

A forecast for consumer demand can have a variety of purposes and can be broadly placed into one of two categories: those that produce a single value as an output (like the peak load for the next day or the next week or even the total load for the next week and so on) or multiple values (like the next few hours or even an hourly forecast for the next day - the load profile).

Looking at forecasts this way we can also further divide them based on the scenarios they were designed for like load flow optimization, economic dispatch, scheduling maintenance and repairs and so on.

2.3.1 Single Value Forecasts

In this category we have work done by pioneers like Park et. al. [11] who used three Artificial Neural Networks to estimate the average load for the next hour, the peak load for the next day and the total energy demand too.

We can also have Artificial Neural Networks working together like this paper by Alfuhaid et. al. [12] where a small network produces estimates of the peak and valley which is used by another network to produce a load profile.

2.3.2 Multiple Value Forecasts

Among the models that forecast multiple values we have Lee et. al. [13] who divides the day into three and uses a separate Artificial Neural Network for each to estimate the load profile for the next day.

We also have work by Bakirtzis et. al. [14] who used Neural Networks focusing on improving their performance on holidays.

2.4 Complexity of the Model

We can also divide the models used for forecasting into a variety of categories based on the complexity of the relationship between the variables it takes as input. Such models can vary from very simple linear models to very complex artificial neural networks. We shall consider each of these kinds of models in turn in the following paragraphs.

2.4.1 Linear Models

Linear models are the simplest kinds of models and were popular for quite some time because they are easy to explain and good for situations with less number of variables. They can be divided into two main categories: time-of-the-day models as well as dynamic models.

Time-of-the-Day Models

Time-of-the-day models are the simplest kind and they take only historical load as input. Due to the periodicity inherent in loads, a very common time-of-the-day model is described by:

$$L(t) = \Sigma_{i=1}^{N} \alpha_i * f_i(t) * v(t)$$

where $f_i(.)$ can be some periodic functions with periods varying from a day to a week depending on the range of forecast, α_i are coefficients and $v(t)$ represents the model error. One study that uses this approach is Fankhauser [15].

Dynamic Models

Dynamic models account for the fact that the load at a given instant is dependent not only on the historical load but on other variables such as the weather, demographics and so on. The most common variety is the Auto-regressive and Moving Average model and though other formulations do exist, they are interchangeable i.e. one can transform the model from one formulation to another due to the linearity of the models. They are primarily of the form described by:

$$L(t) = y_p(t) + y(t)$$

where $y_p(t)$ depends on the historical load and the average weather patterns while $y(t)$ accounts for the random fluctuations in weather. An example of the work that follows this modeling pattern is that of Abou-Hussien et al. [16] which accounts for daily and weekly load averages as well as a bias term that depends on the deviation in weather around the time of forecasting.

Another example of work in this area is that of Hagan et. al. [17] who propose an ARIMA model with a non-linear transformation of the temperature as an input.

2.4.2 Non Linear Models

Support Vector Machines

One of the non-linear approaches for regression is that using support vector machines. This technique was used effectively by Chen et. al. [18] to win the EUNITE competition 2001 which involved predicting the daily maximum load for the next 31 days.

Another example of the work done in this area is that of Wang et. al. [19] who proposed a hybrid model where an algorithm chooses the appropriate parameters to be fed into the support vector model at each time-step. This was shown to be comparable to the state-of-the-art at that time.

Artificial Neural Networks

Artificial Neural Networks have been quite popular in the load forecasting community for quite some time with surveys going as far back as Hippert et. al.in 2001 [6] and they are especially popular for short-term load forecasting (like Charytoniuk et. al. 2001 which we described earlier in this chapter [4]).

Other examples of work in short-term forecasting include that of Nair et. al. [20] who used a probabilistic neural network quite successfully. Yet another of the multitude of examples is the work by Hernandez et. al. [21] who used neural networks for short-term forecasting for micro-grids.

Among the many reasons why neural networks are popular is the ease with which they pick up non-linearity and how they are easy to train on large datasets and even faster to predict in real-time. This makes it easy to have multiple models in an ensemble, each tuned to a particular real-world scenario (as an example we can have different networks for each hour of the day).

Recurrent Neural Networks

The earliest formulations of recurrent neural networks were known as Elman networks and Zhu et. al. [22] used them to account for the influence of weather factors on load for the purpose of short-term load forecasting model.

More recently, Li et. al. [23] uses hybrid Elman networks for short-term load forecasting, investigating how quantizing inputs affect model accuracy and speed. More proper formulations of recurrent neural networks were developed in the 1990s and were quite useful in short-term forecasting. Vermaak et. al [24] utilized these networks for forecasting of the nation-wide load for the South African utility, ESKOM where they were shown to outperform artificial neural networks.

Kermanshahi et. al. [25] used an ensemble of recurrent neural networks and artificial neural networks for long-term load forecasting in Japan, accounting for the weather with good results which helped in long-term utility planning.

Deep Learning Techniques

More recent techniques for load forecasting are ones that have arisen out of the field of deep learning - a field that has become quite popular in the past five years. The promise of such techniques is that they would not need handcrafted features to achieve good levels of performance and they can easily handle large quantities of load data.

Examples in this field include recent work by Marino et. al. [1] and Amarsinghe et. al. [2] who use techniques known as Convolutional Neural Networks and Sequence-to-Sequence Networks for load forecasting. This is an upcoming area of research and more developments are expected in the next few years.

Chapter 3

Theory

3.1 Introduction

Our work has focused on comparing the newer deep learning techniques like Convolutional Neural Networks and Sequence-to-Sequence Models to more established techniques like artificial neural networks for very short-term load forecasting. Hence in this chapter, we shall focus on introducing the theory behind the working of these models.

3.2 Artificial Neural Networks

3.2.1 The Perceptron

Artificial Neural Networks are centered around a fundamental building block known as a perceptron. This building block was invented by Frank Rosenblatt [26] inspired by earlier work by McCulloch and Pitts [27]. A perceptron can be thought of a function that accepts N inputs x_1, x_2, ..., x_N and produces a binary output as per the rule:

$$\text{output} = \begin{cases} 0 & \text{if } \sum_{j=1}^{N} w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_{j=1}^{N} w_j x_j > \text{ threshold} \end{cases} \tag{3.1}$$

where w_1, w_2, ..., w_N and the threshold are parameters that can be tuned as per performance. We can reformulate this into a more convenient notation:

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases} \tag{3.2}$$

where w and b are vectors and "$\cdot$" represents a dot product between vectors.

However, this formulation suffers from the fact that it can suddenly switch from 0 to 1 for a minor change in inputs. Since most optimization problems require smoothness in the output produced so as to tune the parameters, the weights would be hard to learn in this formulation. Hence we make a small modification to introduce smoothness:

$$output = \sigma(w \cdot x + b) \tag{3.3}$$

where σ represents the sigmoid function given by:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.4}$$

The reason that this works is that the sigmoid function is simply a smooth version of the step function as shown in the following figures

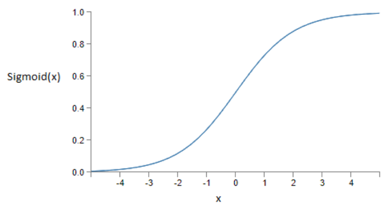

Figure 3.1: Sigmoid Function

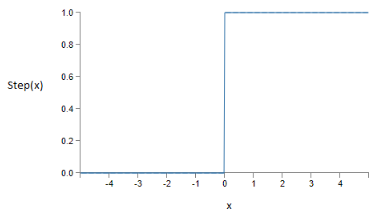

Figure 3.2: Step Function

3.2.2 The Multi-Layer Perceptron

This perceptron can be combined with many others like it to come up with the multi-layer perceptron model. Such a model looks something like:

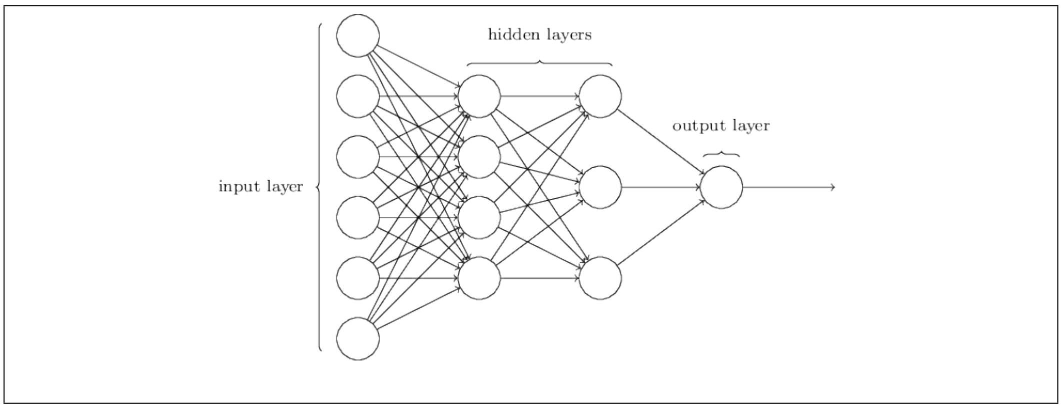

Figure 3.3: Multi-Layer Perceptron

The intuition behind having hidden layers is that each layer applies a non-linear transform to the previous layer and hence it can be proven that each layer can represent more complex functions of the input. This implies that more complex features can be learned and hence the more likely it becomes that the model learns features that can explain the real world fluctuations better.

3.2.3 The Learning Algorithm

The standard way to train artificial neural networks is a combination of two complementary algorithms: backpropagation and gradient descent. The function of backpropagation is to compute the gradient of the error metric (which assesses the performance of the model) with respect to the parameters and that of gradient descent is to use these gradients to update weights so as to reduce the error of the model.

Backpropagation

This algorithm was first applied to neural networks by Rumelhart et. al. [28] and has become the de-facto standard for training neural networks. In order to describe the algorithm [29] let us first define some notation:

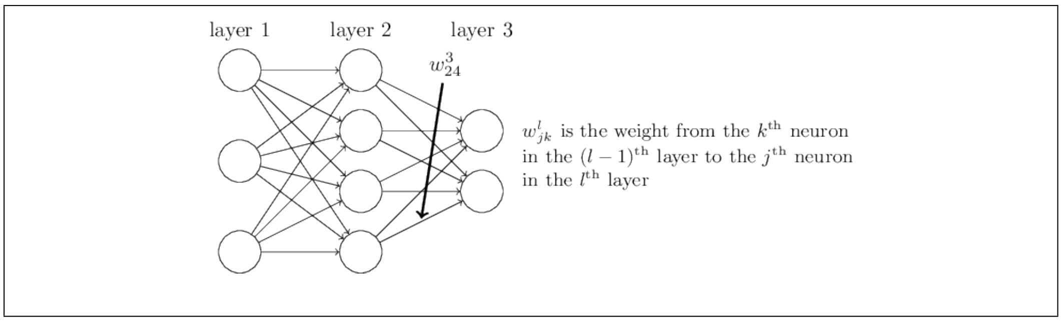
layer 1
layer 2
layer 3
w^3_{24}
w^l_{jk} is the weight from the k^{th} neuron in the $(l-1)^{\text{th}}$ layer to the j^{th} neuron in the l^{th} layer

Figure 3.4: Weight Notation

- w^l_{jk} : The weight connecting the k^{th} neuron in the $(l-1)^{\text{th}}$ layer to the j^{th} neuron in the l^{th} layer
- b^l_j: The bias of the j^{th} neuron in the l^{th} layer
- a^l_j: The activation (output) of the j^{th} neuron in the l^{th} layer

Using this notation we can say that,

$$a^l_j = \sigma\left(\sum_k w^l_{jk} a^{l-1}_k + b^l_j\right) \tag{3.5}$$

This operation can be written in vector notation as:

$$a^l = \sigma\left(w^l a^{l-1} + b^l\right) \tag{3.6}$$

Let us assume for now that there are L layers and the cost function is denoted by C and is some function of the network output and the actual outputs. The goal of backpropagation algorithm is to compute $\partial C/\partial w^l_{jk}$ and $\partial C/\partial b^l_j$. Let us define an intermediate term δ^l_j as:

$$\delta^l_j \equiv \frac{\partial C}{\partial z^l_j} \tag{3.7}$$

From this, through basic calculus, we obtain the following four equations that form the basis of the backpropagation algorithm.

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \tag{3.8}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \tag{3.9}$$

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l. \tag{3.10}$$

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l. \tag{3.11}$$

Here $\nabla_a C$ refers to the derivative of the cost function with respect to the output of the last layer, $\odot$ is similar to the element-wise product of two vectors and $\sigma'(.)$ refers to the derivative of the sigmoid function.

Gradient Descent

The function of the gradient descent algorithm is to use the gradients for cost function with respect to the weights and biases so as to update the parameters.

It is an algorithm designed for convex optimization that would converge to the global extrema, which has been adapted for the non-convex optimization problem of updating parameters.

The intuition behind this algorithm is that the gradient of a parameter represents the tangent to the error curve at the point represented by that set of parameters and the gradient points to the direction in which tweaking the weights provides the maximum reduction in the error.

Gradient descent then uses these gradients to modify the parameter according to the following equation:

$$Param^{k+1} = Param^{k} + StepSize * Gradient \tag{3.12}$$

where k and k + 1 are successive iterations of training of the model.

3.2.4 Modifying Neural Networks for Regression

The activation functions that we mentioned earlier in this chapter i.e. tanh and sigmoid lie in the range $(-1, 1)$ and $(0, 1)$ respectively. However, if we were to use a neural network for regression then we would require it to produce values in the range $(-\inf, \inf)$.

In order to solve this, we modify the activation function to one known as the rectified linear unit (relu).

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ w \cdot x + b & \text{if } w \cdot x + b > 0 \end{cases} \tag{3.13}$$

However, relu units do not output values less than zero. In our current problem statement, where we are dealing with the magnitudes of electric loads, this is not an issue as the magnitude of loads are always positive. Hence we use relu activation function in our models.

3.3 Recurrent Neural Networks

In a traditional neural network, the inputs are assumed to be independent and identically distributed. Therefore we cannot encode notions of sequentiality in an artificial neural network because it assumes all its inputs have no relation with one another. It was with the aim of rectifying this issue that recurrent neural networks were created.

This network was designed by Elman in 1990 [30] and represent one of the earliest efforts to make a practical recurrent neural network. The model is quite simple:

$$h_t = \sigma(W_h * x_t + U_h * h_{t-1} + b_h) \tag{3.14}$$

$$y_t = \sigma(W_y * h_t + b_y) \tag{3.15}$$

Here x_t represents the input at time t, h_t the hidden layer and y_t the output vector.

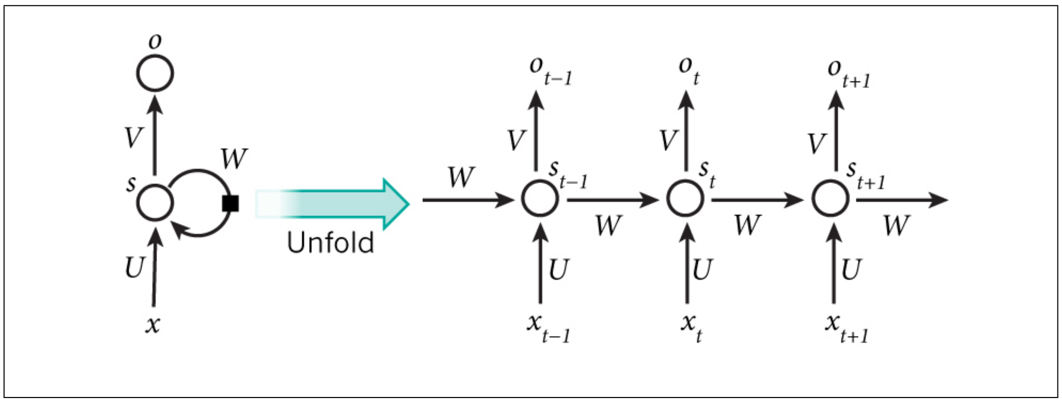

Figure 3.5: Recurrent Neural Networks

One of the most striking things about this network is how the output at any time t depends not only on the input at time t but also on the hidden state at the time (t-1) i.e. this model can encode sequentiality. In other words, this model knows that the input after time t is one before the input at the time (t+1) and one after the input at the time (t-1).

This kind of model can be extended to support many layers of recurrent connections as well and was used by Zhu et. al. [22] and Li et. al. [23], which we discussed in the previous chapter.

3.4 LSTMs: Adding Memory to Recurrent Neural Networks

One of the shortfalls of recurrent neural networks was its inability to deal with long-range dependencies that are common in long sequences of data. In order to alleviate this issue, LSTMs were created by Hochreiter and Schmidhuber in 1997 [31].

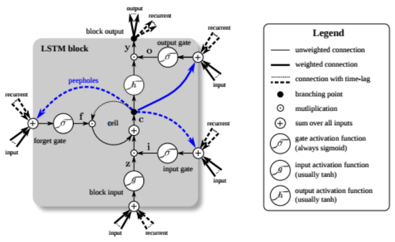

Figure 3.6: LSTM Cell

3.4.1 The LSTM Cell

The LSTM cell is like a recurrent neural network with an attached memory that enables it to encode long-term dependencies. It is most clearly presented by Gref et. al. [32] and we shall follow their path here with intermittent explanations of our own.

Forward Pass

The first part of the LSTM is the forward pass i.e. using the inputs to compute the outputs. As shown in the figure, we can use sigmoid and tanh functions to model "gates" that allow the "memory cell" at the center to store whatever is required of past values so as to minimize the model error.

There are three such gates: one that allows/blocks the contribution from the current input, one for the past cell state and one to block/reveal what is in the cell to the output.
We have several tunable parameters in the LSTM model:

- Input Weight Matrices: W_z, W_i, W_f, W_o
- Past State Weight Matrices: R_z, R_i, R_f, R_o
- Peephole Weight Vectors: p_i, p_f, p_o
- Bias Weight Vectors: b_z, b_i, b_f, b_o

Assuming x^t to be the input vector at time t, we can obtain the outputs for that time-step as follows:

Transform the input to the LSTM block:

$$z^t = tanh(W_z * x^t + R_z * y^{t-1} + b_z) \tag{3.16}$$

Acquire the value of the input gate:

$$i^t = \sigma(W_i * x^t + R_i * y^{t-1} + p_i \odot c^{t-1} b_i) \tag{3.17}$$

Compute the value of the forget gate:

$$f^t = \sigma(W_f * x^t + R_f * y^{t-1} + p_f \odot c^{t-1} + b_f) \tag{3.18}$$

Use the gates to modify the value of the cell:

$$c^t = z^t \odot i^t + c^{t-1} \odot f^t \tag{3.19}$$

Using the cell value obtain the value of the output gate

$$o^t = \sigma(W_o * x^t + R_o * y^{t-1} + +p_o \odot c^t + b_p) \tag{3.20}$$

Finally, combine the cell and the output get to calculate the LSTM output

$$y^t = tanh(c^t) \odot o^t \tag{3.21}$$

3.4.2 Backpropagation through Time

These models can be trained using the backpropagation through time algorithm, which is quite similar to that used for artificial neural networks. Due to space constraints, we shall not go over the mathematics but interested readers can find the necessary details here [32].

3.5 Seq2Seq: Optimizing Recurrent Networks for Sequences

Though LSTMs are quite powerful for exploiting sequences of data they cannot effectively map an input sequence to an output sequence and it is for such tasks that Sutskever et. al. [33] created sequence-to-sequence networks which can potentially be useful for predicting sequences of future loads like creating a load profile and so on.

These networks involve mapping the entire input sequence to a fixed length vector using an LSTM network known as the encoder which is then used by another LSTM network known as the decoder to create a sequence as the output.

Such networks are quite effective at short length sequences but falter as the length of the sequence increases. To improve their performance, we apply a technique known as attention [34]. It involves finding which of the previous inputs is most useful for predicting the current output and then focusing more on that. Such a technique can possibly be quite effective in load forecasting as well.

3.6 Convolutional Networks for Sequences

Although convolutional neural networks are most famous for their applications on images, these networks have been useful when applied to sequential data as well. [35] Here we provide a brief introduction to convolutional neural networks as well.

3.6.1 The Convolution Operation

The convolution operation involves applying a variety of transformations to sliding fixed length section of the input. These transformations are learned by the network so as to minimize the error and the intuition behind using the same transform to all parts of the input is that it allows us to detect whether or not a particular feature is present in the input, regardless of where it may be located.

Mathematically, for a two-dimensional input, the convolution operation can be represented as:

$$s(i,j) = \sum_{l} \sum_{m} I(l,m) * K(i+l, j+m) \tag{3.22}$$

where "I" represents the two-dimensional input while K represents the transformation applied toa the fixed length window.

3.6.2 The Max Pooling Operation

The purpose of the pooling operation is to detect the presence of the absence of the features in the input that can be exploited for forecasting by summarizing the outputs of the convolution layer obtained from all the different parts of the inputs. This allows us to obtain spatial invariance i.e. we can detect a feature regardless of where it is present in the input.

Chapter 4

Experimental Results

4.1 Aim

The aim of work so far has been to benchmark different techniques so as to answer the following question:

Given the load value of the past 50 time steps, what would the load be at the next time step?

as described in [1].

4.2 The Dataset

The dataset we have chosen to use has measurements of household power consumption sampled at a minute resolution for over three years leading to 2075259 measurements. [36]

There also exists active power consumed by three sub-meterings of the house but we focus on the total active power as in [1].

4.3 A Few Conjectures

4.3.1 Input Features

The promise of deep learning architectures is that they will eventually minimize the amount of pre-processing that needs to be done so as to get good performance on a variety of datasets in several domains.

We conjecture that the advantage of not using features is that the same architecture can be used in a wide variety of load forecasting settings without modification since there are no case-specific features that restrict the domain in which the model can work.

However, we also conjecture that the presence of case-specific features may aid performance and could be a source of future work as well. In our work so far we have used raw inputs as our aim was to replicate a paper [1] and benchmark it against artificial neural networks.

4.3.2 Predicting Sequences

As the experiments will show below, there is not much difference across a wide variety of architectures of newer Sequence to Sequence and Convolutional Neural Networks in comparison to standard artificial neural networks.

We conjecture that this is because the purpose of the Sequence to Sequence architectures is to map sequences to sequences and hence if we were to use them to create a load profile, they may surpass artificial neural networks. Recent work by Amarsinghe et. al. [2] suggests something similar however since the paper lacks the details about the architecture of the artificial neural network used, we cannot say with certainty.

4.4 The Architecture

So, we tested the four types of models namely ANNs, Sequence to Sequence Networks with and without Attention and the Convolutional Neural Networks on a readily available dataset of ”Individual Household Electrical Power Consumption” (UCI Machine Learning Repository [36]. This dataset contains 2 million household power measurements between December 2006 and November 2010 at a minute resolution level. After removing the missing measurements from the dataset, it was divided into three parts - 80% for training, 10% for validation and the rest 10% for testing purpose.

We tested the models both at an hour and a minute resolution and also changed the layers and depth to get the optimum results. The hour resolution data is obtained by getting the mean of the observed active powers at the minute level. The convergence is then tested on the basis of Root Mean Square Error (RMSE) and later, the evaluation on the test set is done using the best model (among the checkpoints at various epochs) picked up according to the minimum error on the validation set.

4.5 Comparison between Models

4.5.1 Artificial Neural Networks

We begin testing the models on the architecture given in Figure 4.1. The learning rate and activation were kept to be default(*ReLU*) and the optimizer used was *ADAM*. The output layer contained a linear activation to produce the output. From the observations in Table 4.1 at various depths and number of units, it is clear that the RMSE at minute resolution remains approximately equal to 0.211959 on an average while the RMSE at hour resolution is 0.180737 on an average. Though we obtain the best model at a depth of 2 with 50-30 configuration in layers 1 & 2 at minute level (RMSE = 0.207691, Fig 4.5) and 30-20 configuration in layers 1 & 2 at hour level (RMSE = 0.145676, Fig 4.6) but as such, there's no trend observed on increasing the depth or the number of units.

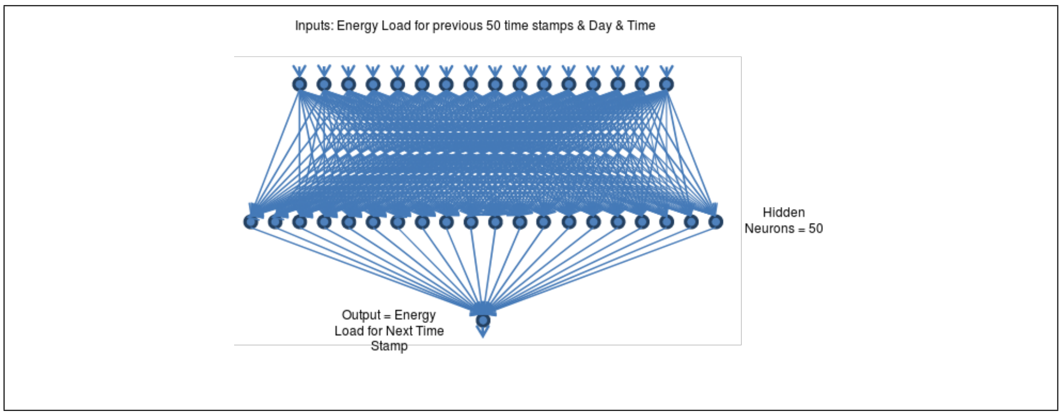
Inputs: Energy Load for previous 50 time stamps & Day & Time
Hidden Neurons = 50
Output = Energy Load for Next Time Stamp

Figure 4.1: ANN Architecture for Load Forecasting

Table 4.1: Test RMSE for Different ANN Architectures

Layers	Depth	Minute RMSE	Hour RMSE
10	1	0.210364	0.189045
20	1	0.216577	0.202007
30	1	0.208589	0.219522
50	1	0.21216	0.192782
100	1	0.209208	0.164293
30-20-1	**2**	0.213283	**0.145676**
50-10-1	2	0.210746	0.187626
50-20-1	2	0.212942	0.151476
50-30-1	**2**	**0.207691**	0.168448
40-30-20-1	3	0.21703	0.186498

4.5.2 LSTM based Sequence to Sequence (Seq2Seq) Networks

Without Attention

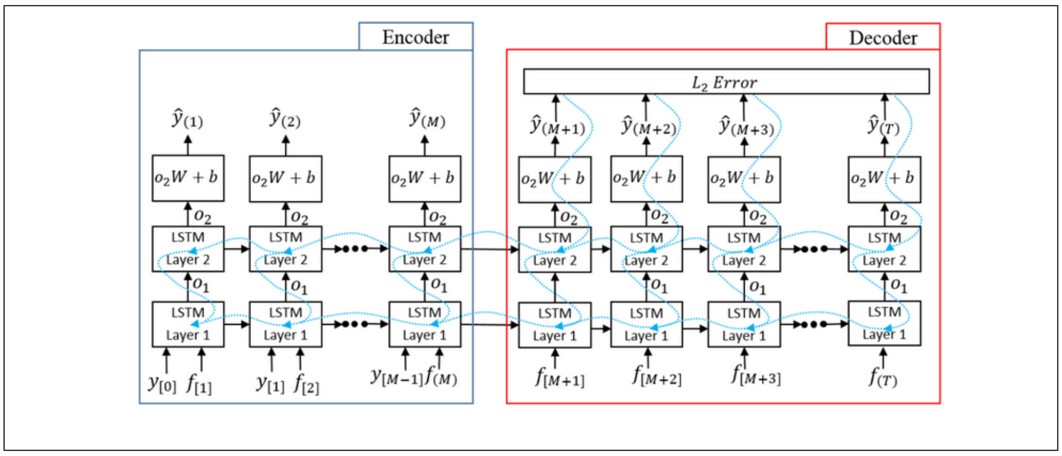

Figure 4.2: S2S Architecture for Load Forecasting[1] without Attention

The following is a replica of the architecture used in [1]. Across the models at minute resolution (Table 4.2), we get the best model when layers are 2 with 50 LSTM units each (RMSE=0.195, Fig 4.5) while at hour resolution, the model with 3 layers of 20 LSTM units come out to be the best (RMSE=0.392, Fig 4.6). Overall, the validation RMSE decreased on increasing the depth showing the goodness of fit but going wider doesn't help much.

Table 4.2: Test RMSE for Different Seq2Seq Architectures without Attention

Layers	Units	Minute RMSE	Hour RMSE
1	5	0.222	0.437
2	5	0.203	0.431
3	5	0.199	0.433
4	5	0.204	0.424
1	20	0.203	0.408
2	20	0.202	0.397
3	**20**	0.202	**0.392**
1	50	0.201	0.398
2	**50**	**0.195**	0.393
1	100	0.201	0.397

Table 4.3: Test RMSE for Different Seq2Seq Architectures with Attention

Layers	Units	Minute RMSE	Hour RMSE
2	5	0.180665	0.428480
3	5	0.176760	0.414452
4	5	0.191332	0.409852
2	**20**	**0.175622**	0.402106
3	20	0.176663	0.398950
2	**50**	0.176326	**0.393810**

With Attention

The attention is always added to the next to the last layer of the decoder (see Fig 4.3). On adding attention, we definitely observed improvement in the fitness at minute resolution(see RMSE in Table 4.3) since now it dropped in the range 0.17-0.18 but there's not much improvement for hour resolution. Moreover, we can say that the previous hour inputs don't help much while adding Attention Mechanism. It might be possible that if the time window(number of time steps) is large, such an input doesn't helps in reducing the error.

Here, we obtain the best model with 2 layers of 20 LSTM units at minute resolution (RMSE=0.175622, Fig 4.5) and with 2 layers of 50 LSTM units at hour resolution (RMSE=0.393810, Fig 4.6).

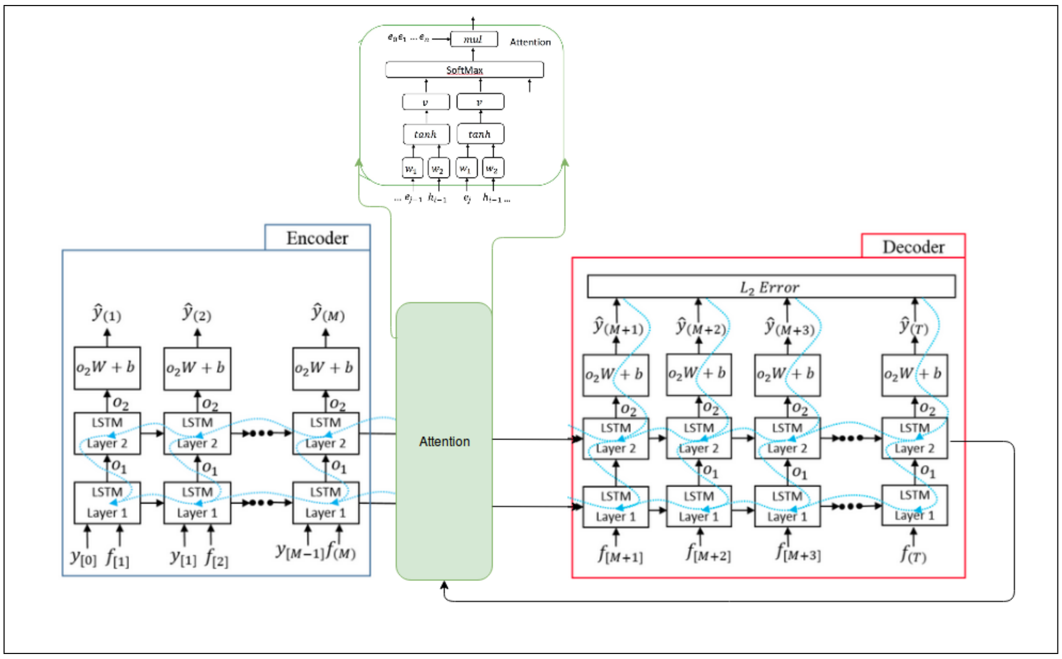

Figure 4.3: S2S Architecture with Attention for Load Forecasting

4.5.3 Convolutional Neural Networks

The overall network for using convolutional neural networks for load forecasting looks something like Figure 4.4[2]. Here, two hidden layers with 20 neurons and ReLU activation were used. The last layer had a linear activation in order to give the output. We tested different CNN architectures by changing the convolution filters, kernel sizes or the pooling filters.

Here, we can see that on increasing the size of pooling filter, the RMSE decrease resulting in poor fit. Though CNNs doesn't perform well at the minute resolution with the best model having RMSE of 0.262238 (Table 4.4), it outperforms each of the models tested before at hour resolution (RMSE = 0.043336 for the best model).

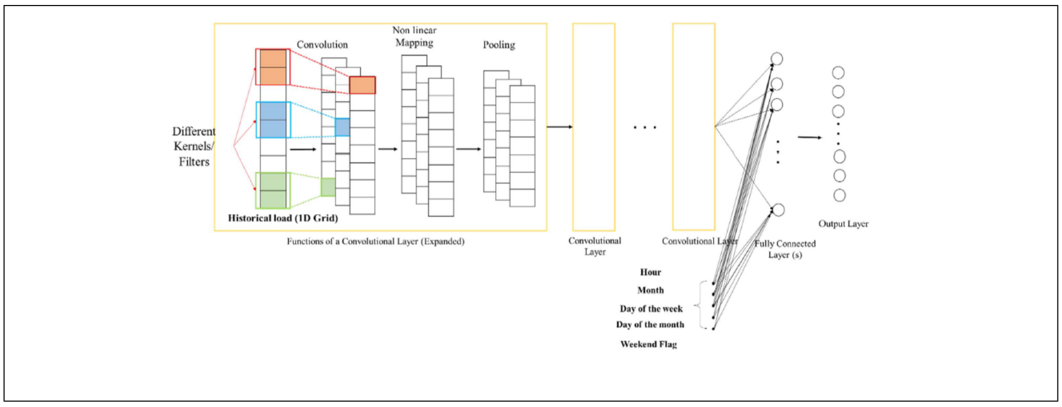

Figure 4.4: CNN Architecture for Load Forecasting[2]

Table 4.4: Test RMSE for Different CNN Architectures

Conv Filters	Kernel Size	Pool Filters	Hidden Size	Minute RMSE	Hour RMSE
[4]	**[[10]]**	**[[2]]**	**[20,20]**	0.263560	**0.043336**
[4]	[[10]]	[[5]]	[20,20]	0.316765	0.180047
[4]	[[20]]	[[2]]	[10,10]	0.262690	0.170212
[10]	[[20]]	[[2]]	[10,10]	0.264193	0.061587
[5,5]	[[10],[5]]	[[3],[3]	[20,20]	0.269748	0.233356
[4,8,8]	[[10],[10],[6]]	[[2],[2],[2]]	[20,20]	0.263721	0.145410
[4,4,8]	**[[10],[5],[2]]**	**[[1],[1],[2]]**	**[10]**	**0.262238**	0.054470
[4,8,16,16,32]	[[3],[3],[3],[3],[3]]	[[2],[2],[2],[2],[2]]	[20,20]	0.289674	0.148782

4.6 Summary Table and Plots

In the end, we present the best models for the different architectures and we can compare them easily seeing that Sequence to Sequence models with Attention perform the best at minute level while CNNs win at the hour level (Table 4.5). We have also plotted the best models comparing the absolute percentage errors.

Table 4.5: Summary Table of performance of four architectures according to best models

Architecture	Min Minute RMSE	Min Hour RMSE
Artificial Neural Networks	0.208	0.146
Seq2Seq without Attention	0.195	0.392
Seq2Seq with Attention	**0.176**	0.394
Convolutional Neural Networks	0.262	**0.043**

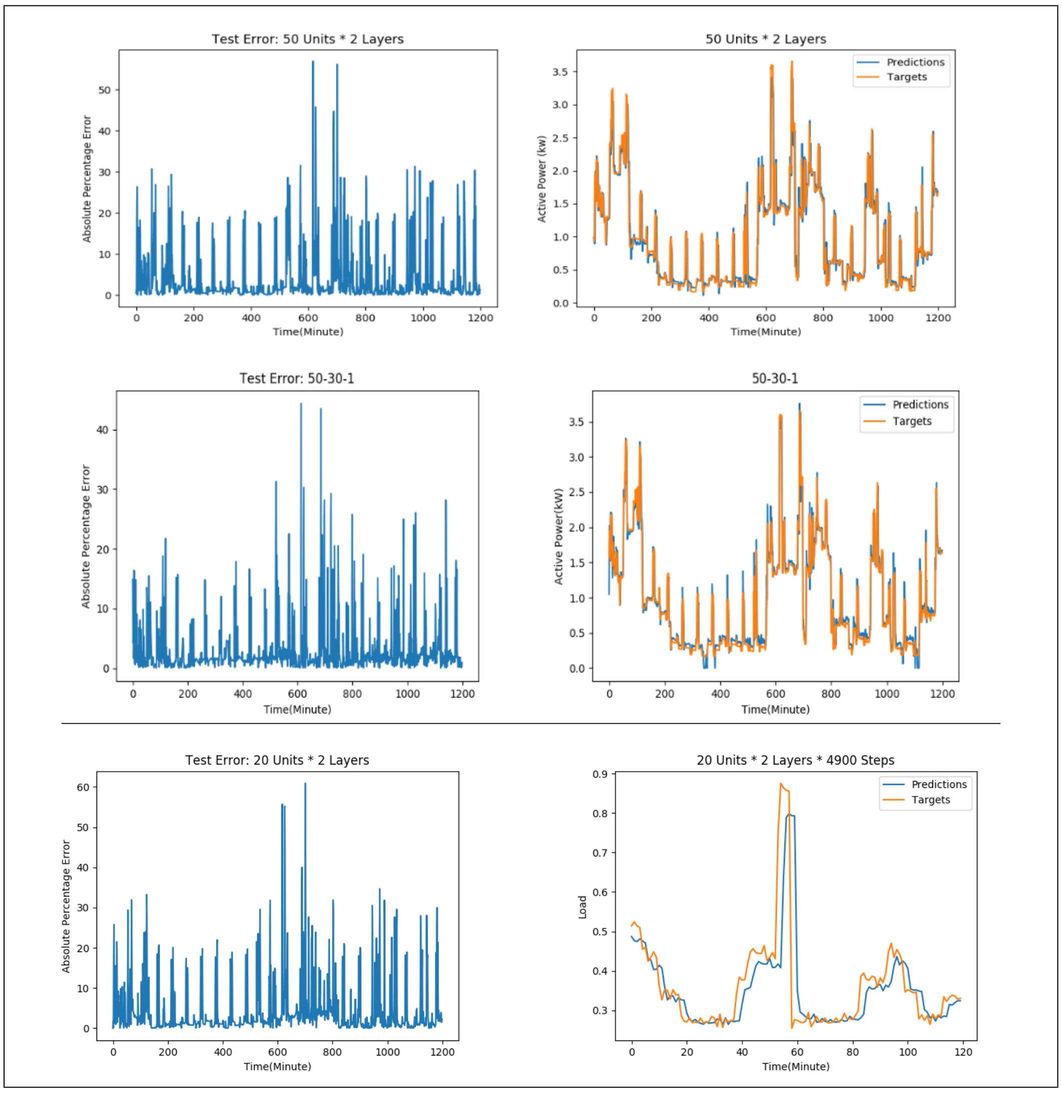

Figure 4.5: Sample plots for Best Models for Minute Resolution: Top - Seq2Seq without Attention, Middle - ANN, Bottom - Seq2Seq with Attention

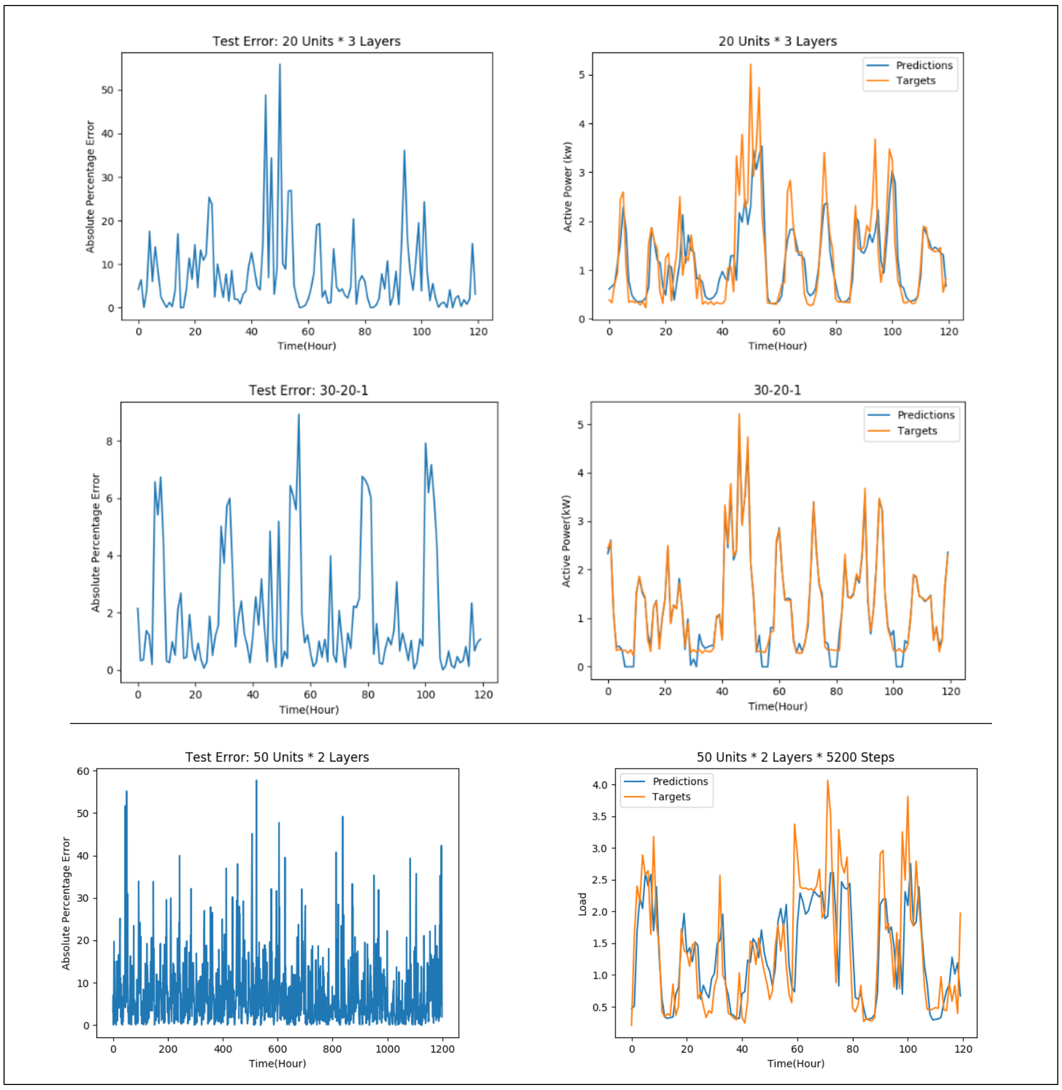

Figure 4.6: Sample plots for Best Models for Hour Resolution: Top - Seq2Seq without Attention, Middle - ANN, Bottom - Seq2Seq with Attention

Chapter 5

Conclusion

We, therefore, came with four different architectures for energy load forecasting and compared them at minute and hour level resolution. From the different architectures, we affirm that the Sequence-to-Sequence networks with Attention work the best at minute level while the Convolutional Neural networks overtake them an hour level. Although there was a very little change in the eror between the two still the attention adds up the confidence in the model and gives a better prediction for the next time step from the previous fifty time-steps.

From our work so far, we infer that the recent deep neural networks perform comparably to Artificial Neural Networks, which suggests that the task of predicting only the next time step from the past few is a relatively easy task. We conjecture that when the forecast horizon is increased, the newer models will overtake the traditional techniques of load prediction since they are specialized for dealing with sequences. However future work will be needed to provide definite answers.

Though our results are valid only for very short-term load forecast, future work can also be done to investigate the effectiveness of our techniques to short, medium and long-term forecasts. Another line of work can be finding out the best features to maximize the performance of newer techniques - we choose to use raw inputs so as to establish baseline performance but handcrafted features can provide better performance.

The field is progressing fast and we require the best possible model for load prediction. Such studies can thus help us in improving the algorithms and models oriented to this area for future.

Bibliography

[1] Daniel L Marino, Kasun Amarasinghe, and Milos Manic. Building energy load forecasting using deep neural networks. In *Industrial Electronics Society, IECON 2016-42nd Annual Conference of the IEEE*, pages 7046–7051. IEEE, 2016.

[2] Kasun Amarasinghe, Daniel L Marino, and Milos Manic. Deep neural networks for energy load forecasting. In *Industrial Electronics (ISIE), 2017 IEEE 26th International Symposium on*, pages 1483–1488. IEEE, 2017.

[3] Heiko Hahn, Silja Meyer-Nieberg, and Stefan Pickl. Electric load forecasting methods: Tools for decision making. *European journal of operational research*, 199(3):902–907, 2009.

[4] Wiktor Charytoniuk and M-S Chen. Very short-term load forecasting using artificial neural networks. *IEEE transactions on Power Systems*, 15(1):263–268, 2000.

[5] Jingfei Yang. *Power system short-term load.* PhD thesis, Technische Universität, 2006.

[6] Henrique Steinherz Hippert, Carlos Eduardo Pedreira, and Reinaldo Castro Souza. Neural networks for short-term load forecasting: A review and evaluation. *IEEE Transactions on power systems*, 16(1):44–55, 2001.

[7] Elias Kyriakides and Marios Polycarpou. Short term electric load forecasting: A tutorial. *Trends in Neural Computation*, pages 391–418, 2007.

[8] E Doveh, P Feigin, D Greig, and L Hyams. Experience with fnn models for medium term power demand predictions. *IEEE Transactions on Power Systems*, 14(2):538–546, 1999.

[9] MS Kandil, Sm M El-Debeiky, and NE Hasanien. Long-term load forecasting for fast developing utility using a knowledge-based expert system. *IEEE transactions on Power Systems*, 17(2):491–496, 2002.

[10] Zhiheng Zhang and Shijie Ye. Long term load forecasting and recommendations for china based on support vector regression. In *Information Management, Innovation Management and Industrial Engineering (ICIII), 2011 International Conference on*, volume 3, pages 597–602. IEEE, 2011.

[11] Dong C Park, MA El-Sharkawi, RJ Marks, LE Atlas, and MJ Damborg. Electric load forecasting using an artificial neural network. *IEEE transactions on Power Systems*, 6(2):442–449, 1991.

[12] AS AlFuhaid, MA El-Sayed, and MS Mahmoud. Cascaded artificial neural networks for short-term load forecasting. *IEEE Transactions on Power Systems*, 12(4):1524–1529, 1997.

[13] KY Lee, YT Cha, and JH Park. Short-term load forecasting using an artificial neural network. *IEEE Transactions on Power Systems*, 7(1):124–132, 1992.

[14] AG Bakirtzis, V Petridis, SJ Kiartzis, MC Alexiadis, and AH Maissis. A neural network short term load forecasting model for the greek power system. *IEEE Transactions on power systems*, 11(2):858–863, 1996.

[15] HR Fankhauser. A novel approach to on-line short and intermediate-term load forecasting. In *proc., 8th Power Systems Computation Conference*, pages 376–380, 1984.

[16] MS Abou-Hussien, MS Kandlil, MA Tantawy, and SA Farghal. An accurate model for short-term load forecasting. *IEEE Transactions on Power Apparatus and Systems*, (9):4158–4165, 1981.

[17] Martin T Hagan and Suzanne M Behr. The time series approach to short term load forecasting. *IEEE Transactions on Power Systems*, 2(3):785–791, 1987.

[18] Bo-Juen Chen, Ming-Wei Chang, et al. Load forecasting using support vector machines: A study on eunite competition 2001. *IEEE transactions on power systems*, 19(4):1821–1830, 2004.

[19] Jianjun Wang, Li Li, Dongxiao Niu, and Zhongfu Tan. An annual load forecasting model based on support vector regression with differential evolution algorithm, 2012.

[20] Aneesh Nair and SK Joshi. Short term load forecasting using probabilistic neural network based algorithm. In *Computational Intelligence and Communication Networks (CICN), 2010 International Conference on*, pages 128–132. IEEE, 2010.

[21] Luis Hernandez, Carlos Baladrón, Javier M Aguiar, Belén Carro, Antonio J Sanchez-Esguevillas, and Jaime Lloret. Short-term load forecasting for microgrids based on artificial neural networks. *Energies*, 6(3):1385–1408, 2013.

[22] Sheng ZHU, Chuan-wen JIANG, and Zhi-jian HOU. Application of a weather component based elman neural network to short-term load forecasting [j]. *Proceedings of Electric Power System and Automation*, 1:005, 2005.

[23] Penghua Li, Yinguo Li, Qingyu Xiong, Yi Chai, and Yi Zhang. Application of a hybrid quantized elman neural network in short-term load forecasting. *International Journal of Electrical Power & Energy Systems*, 55:749–759, 2014.

[24] J Vermaak and EC Botha. Recurrent neural networks for short-term load forecasting. *IEEE Transactions on Power Systems*, 13(1):126–132, 1998.

[25] Bahman Kermanshahi. Recurrent neural network for forecasting next 10 years loads of nine japanese utilities. *Neurocomputing*, 23(1):125–133, 1998.

[26] F. Rosenblatt. *Principles of neurodynamics: perceptrons and the theory of brain mechanisms.* Report (Cornell Aeronautical Laboratory). Spartan Books, 1962.

[27] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[28] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[29] Michael Nielsen. *Neural Networks and Deep Learning.* Determination Press, 2015.

[30] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[31] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[32] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 2017.

[33] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[34] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[35] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

[36] Kevin Bache and Moshe Lichman. Uci machine learning repository. 2013.