www.cya.unam.mx/index.php/cya

Contaduría y Administración 68 (1), 2023, 77-96

Hours ahead automed long short-term memory (LSTM) electricity load forecasting at substation level: Newcastle substation

Previsión de carga eléctrica automatizada a unas horas adelantadas con el modelo (LSTM) al nivel subestacional: subestación de Newcastle

Wellcome Peujio-Jiotsop-Foze*, Adrián Hernández-del-Valle

Instituto Politécnico Nacional, México

Received May 17, 2021; accepted October 11, 2022
Available online October 12, 2022

Abstract

Nowadays, electrical energy is of vital importance in our lives, every country needs this resource to develop its economy, factories, businesses, and homes are the basis of the economic structure of a country. In the city of Newcastle as in other cities are in constant development growing day by day in terms of industries, homes and businesses, these elements are the ones that consume all the electricity produced in Newcastle. Although Australia has strategically located substations that serve the function of supplying all existing loads with quality power, from time to time the load will exceed the capacity of these substations and will not be able to supply the loads that will arise in the future as the city grows. To find a solution to this problem, we use a deep learning model to improve accuracy. In this paper, a Long Short-Term Memory recurrent neural network (LSTM) is tested on a publicly available 30-minute dataset containing measured real power data for individual zone substations in the Ausgrid supply area data. The performance of the model is comprehensively compared with 4 different configurations of the LSTM. The proposed LSTM approach with 2 hidden layers and 50 neurons outperforms the other configurations with a mean absolute error (MAE) of 0.0050 in the short-term load forecasting task for substations.

JEL Code: C45, C53
Keywords: deep learning; forecasting; electric load; LSTM; substation

* Corresponding author.
E-mail address: wpeujioj1700@alumno.ipn.mx (W. Peujio-Jiotsop-Foze).
Peer Review under the responsibility of Universidad Nacional Autónoma de México.

http://dx.doi.org/10.22201/fca.24488410e.2023.3356

Resumen

Hoy en día la energía eléctrica es de vital importancia en nuestras vidas, todo país necesita de este recurso para desarrollar su economía, las fábricas, los negocios y los hogares son la base de la estructura económica de un país. En la ciudad de Newcastle al igual que en otras ciudades están en constante desarrollo creciendo día a día en cuanto a industrias, hogares y negocios, estos elementos son los que consumen toda la electricidad producida en Newcastle. A pesar de que Australia cuenta con subestaciones estratégicamente ubicadas que cumplen la función de abastecer todas las cargas existentes con energía de calidad, de vez en cuando la carga superará la capacidad de estas subestaciones y no podrá abastecer las cargas que surgirán en el futuro a medida que la ciudad crezca. Para encontrar una solución a este problema, utilizamos un modelo de aprendizaje profundo para mejorar la precisión. En este trabajo, se prueba una red neuronal recurrente de memoria a corto plazo (LSTM) en un conjunto de datos de 30 minutos disponible públicamente que contiene datos de potencia real medidos para subestaciones de zonas individuales, en los datos del área de suministro de Ausgrid. El rendimiento del modelo se compara exhaustivamente con 4 configuraciones diferentes del modelo LSTM. El enfoque LSTM propuesto con 2 capas ocultas y 50 neuronas supera a las otras configuraciones con un error medio absoluto (MAE) de 0,0050 en la tarea de previsión de carga a corto plazo para subestaciones.

Código JEL: C45, C53
Palabras clave: aprendizaje profundo; previsión; carga eléctrica; LSTM; subestación

Introduction

Load forecasting has been an important process for years in the field of energy in general and of electric utility. In the industries, the needs of load forecasting, such as planning, operations, and maintenance, become more important than before. Nowadays, with the promotion of smart grid technologies, load forecasting is of even greater importance due to its applications in the planning of demand side management, electric vehicles, and distributed energy resources. Many users (households, businesses, and government) of the utilities produce their own load forecasts, which results in the inefficient and ineffective use of resources (Weicong & Yan, 2019). This paper proposes an integrated load forecasting framework with the concentration on substation level. The tool conducts a deep learning model (LSTM) analysis of Newcastle CBD substation system during the past three years of 30-minute load data for zone substation in the Ausgrid network on which normal voltage supplies to 33kV feeders are served by that substation. The purpose of the assessment is to predict the next hours loading in Newcastle CBD substation and automate the process (Ausgrid, Distribution and Transmission Annual Planning Report, 2018). The dataset is from 2014 to 2016 hourly contains 30-minute metered real power data for individual zone substations in the Ausgrid supply area from January 1st, 2014 to December 31st, 2016, in annual sets. The data is taken directly from the original Ausgrid zone substation dataset.

This analysis accounts for major factors that influence the real and reactive power consumption at any given time and day. Major factors considered include random ("stochastic") customer behaviour by time of day, day of week, and season, as well as ambient weather conditions (temperature and humidity) that have a significant impact on demand. The LSTM model is used to "predict" the hourly real power demand on randomly selected weekdays (including summer, winter, and shoulder season weekdays) when normal voltage (not reduced voltage) is applied to the feeders. The predicted real power values are then compared with the actual measurements on randomly selected weekdays to determine the accuracy of the predictions. The accuracy of the predictions is well within the target error rate, with a mean absolute error (MAE) of 0.0050.

The rest of the paper is organised as follows. Section I provides the background of the load forecasting community. Section II presents forecasting framework based on LSTM. Section III Introduces the experimental setup. The testing dataset and experimental results are given in the section IV while section V concludes the paper.

Related work

Smart grid data has been used for many electricity load forecasting tasks (Wang, Chen, Hong, & Kang, 2018). The data is treated as sequential data. Most commonly Autoregressive integrated moving average (ARIMA), Support Vector Machines (SVM), linear regression, and Artificial Neural Networks (ANN) have been tested to forecast the electricity load.

An ARIMA model is a class of statistical models for analysing and forecasting time series data. It explicitly carters to a suite of standard structures in time series data, and as such provides a simple yet powerful method for making skilful time series forecasts. ARIMA is an acronym that stands for Autoregressive Integrated Moving Average.it is a generalisation of the simpler Autoregressive Moving Average and adds the notion of integration. (Norizan, Maizah, Zuhaimy, & Suhartono, 2010) found that the MAPE for the one-step ahead out-sample forecasts from any horizon ranging from one week led time to one month one week lead time are all less than 1%. Therefore, they proposed that a double seasonal ARIMA model with one-step ahead forecast must be considered in forecasting time series data with two seasonal cycles, especially in Malaysia load data. On the other hand, (Bishnu, Motoi, Aya, & Toshiya, 2019) proposed a forecasting method for the electricity load of a university buildings using a hybrid model comprising a clustering technique and the ARIMA model and the combination has proved to increase the performance of forecasting rather than that using the ARIMA model alone.

A Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification and regression challenges. However, it is mostly used in classification problems. The SVM was proposed during the EUNITE network competition by (Bo-Juen & Ming-Wei, 2004). They found that the temperature (or other types of climate information) might not be useful in such a mid-term load forecasting problem and that the introduction of time-series concept may improve the forecasting. Support Vector machines have been successfully employed to solve nonlinear regression and time series problems. However, SVM have rarely been applied to forecasting electricity load. Moreover, Simulated Annealing (SA) algorithms were used to illustrate the proposed SVMSA (support vector machines with simulated annealing) model. SVMSA has been used by (Ping-Feng & Wei-Chiang, 2005) in load data from Taiwan, the empirical results reveal that the proposed model outperforms the other two modes, namely the autoregressive integrated moving average (ARIMA) model and the general regression neural networks (GRNN) model.

An Artificial Neural Networks (ANN) or neural networks (NNs) are biologically inspired computer programs designed to simulate the way in which the human brain processes information. ANNs gather their knowledge by detecting the patterns and relationships in data and learn (or are trained) through experience, not from programming. An ANN is formed from hundreds of single units, artificial neurons, or processing elements (PE), connected with coefficients (weights), which constitute the neural structure and are organised in layers. The power of neural computations comes from connecting neurons in a network. Each PE has weighted inputs, transfer function and one output. The behavior of a neural network is determined by the transfer functions of its neurons, by the learning rule, and by the architecture itself. During training, the inter-unit connections are optimized until the error in predictions is minimized and the network reaches the specified level of accuracy. Once the network is trained and tested it can be given new input information to predict the output. ANN represents a promising modelling technique, especially for data sets having non-linear relationships which are frequently encountered in electricity load processes. In terms of model specification, ANNs require no knowledge of the data source but, since they often contain many weights that must be estimated, they require large training sets. In addition, ANNs can combine and incorporate both literature-based and experimental data to solve problems. The various applications of ANNs can be summarised into classification or pattern recognition, prediction, and modelling.

(Hong, C., & A., 2002) proposed an artificial neural network (ANN)-based short-term load forecasting technique that considers electricity price as one of the main characteristics of the system load, demonstrating the importance of considering pricing when predicting loading in today's electricity markets. Therefore (Abdollah & Mohammad-Reza, 2013) proposed a new hybrid forecasting method based on the wavelet transform ARIMA and ANN for short-term load forecasting. In the proposed model, the autocorrelation function and the partial autocorrelation function are utilised to see the stationary or non-stationary behaviour of the load time series. Finally, the outputs of the ARIMA and ANN are summed. The empirical results show that the proposed hybrid method can improve the load forecasting accuracy suitably.

The forecasting framework based on LSTM

Multilayer neural network

Before developing the LSTM model, we will talk about a special type of network called a multilayer perceptron (MLP). This network consists of three layers: an input layer, a hidden layer, and an output layer (Peter J. & Richard A., 2016). The input layer and the output layer are fully connected to the hidden layer, respectively. Such a network with more than one hidden layer is called a deep artificial neural network. A deep layer network is shown below Figure 1. It is a well-defined technique for solving real-life problems such as speech recognition, image classification, video analysis, as well as forecasting video analysis, as well as weather forecasting, stock market forecasting and stock market stock market and energy demand forecasting.

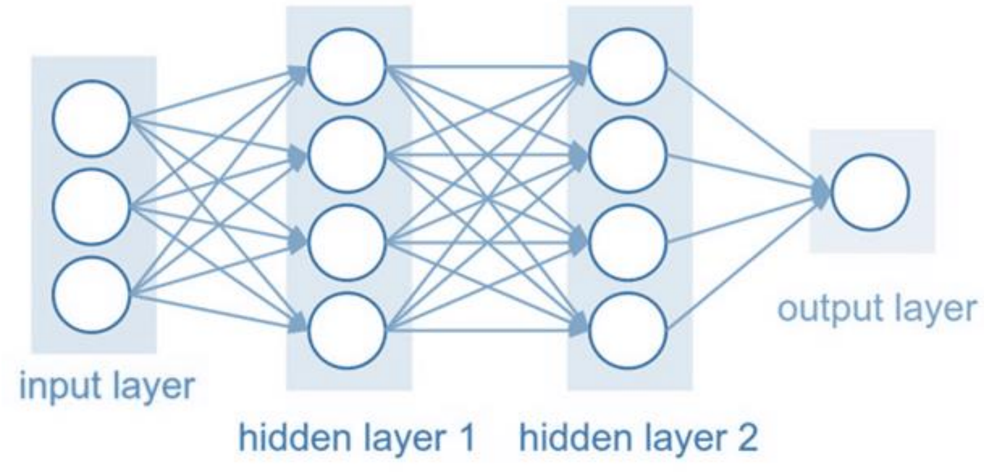

Figure 1. Deep Layer Network
Source: own elaboration with information from analyticsvidhya.com

LSTM framework

LSTM is very different to other deep learning techniques such as multilayer perceptron (MLP) and convolutional neural network (CNN). Data scientists use it specifically for sequence production problems. LSTM is a unique type of recurrent neural network (RNN) capable of learning long-term dependencies which is very useful for certain types of predictions that require the network to retain information over a very long time. LSTM networks are very suitable for classifying, parsing, and making predictions based on time series and sequence data.

Like a typical neural network, the LSTM is comprised of layers and neurons. Input data is propagated through the network for prediction. However, in a feedforward neural network, information flows only in forward direction from the input nodes to the hidden layers and to the output nodes, besides that, is not cycle or loop in the network, there are some issues in this specific architecture because it cannot handle sequential data very well and it only considers the current inputs and cannot memorize or take into consideration of the previous input and the LSTM overcome these problems.

Unlike a neural network, LSTM not only considers current input, but it also considers and memorizes the previous input in such a way that higher accuracy can be achieved. As mentioned like RNN, LSTM has recurrent connections so that they stay from the previous activation of the neuron from the previous time step. However, the RNN also suffers from two problems, the first is vanishing gradient problem. The vanishing gradient problem is a particular problem with RNN as the update of the network involves unrolling the network for each input time step, in effect creating a very deep network that requires weight update. The second is the exploding gradient problem, where the accumulation of large derivatives results in the model being very unstable and incapable of effective learning, the large changes in the model weights creates a very unstable network, which at extreme values, the weights become so large that is causes overflow resulting in missing (or NaN) weight values of which can no longer be expanded.

The LSTM has a unique formulation that allows it to avoid the problems and maintains a constant error which allows them to continuously learn over numerous time steps.

In short, LSTM overcomes the memory problems that RNN suffers. One reason for the success of this recurrent network lies in its ability to handle the exploding/vanishing gradient problem, which stands as a difficult issue to be circumvented when training recurrent or very deep neural networks. LSTM can achieve impressive results in sequential prediction problems and has gained huge popularity in recent years.

Sequence prediction is different to other types of supervised learning problems, the secret is within the models that must be trained and making predictions. Generally, predictions that involve sequence data are referred to as sequence prediction problems and there are four common types of sequence predictions problems, namely: sequence prediction, sequence classification, sequence generation and sequence to sequence prediction. No matter which types of problems that is dealing, the sequence imposes an explicit order on the observations and the order is very important and it must be respected in the formulations of predictions problems that use the sequence data as input and output for that model.

Limitations of LSTM

Although LSTM are good for solving sequence problems and the results are very impressive, LSTM are not failsafe. Their two most problematic aspects are those arising from overfitting and their black box character.

Overfitting occurs whenever a model fits its training set so well that it fails to generalise correctly when we use it on a test set different from the training data set, we used to build it (Jabbar & Khan, 2014). This is a common problem in many machine learning techniques. Neural networks include a multitude of adjustable parameters: the weights that model the connections between neurons (Dietterich, 1995). This large number of parameters makes them prone to overfitting problems. To solve this problem, a multitude of techniques have been proposed that, to a greater or lesser extent, make it possible to avoid this problem (Yunita, 2018), (Schittenkopf, Deco, & Brauer, 1997), (Piotrowski & Napiorkowski, 2013), (Salman & Liu, 2019), (Tetko, Livingstone, & Luik, 1995).

In certain application domains, the most pressing problem with artificial neural networks is our inability to determine how neural networks reach a conclusion. In a neural network, we can look at the input of the network and see what its output is, but its inner workings are something that cannot be described symbolically. For us, a neural network is, to a large extent, a black box. This problem is something on which much remains to be done and which may have implications for the security of systems that employ neural networks internally (Franco, 2019).

LSTM architecture

Before moving on LSTM, let us have a quick look into the RNN, to have a better understanding of the basic recurrent relations concept.

Recurrent Neural Network is a generalization of feedforward neural network that has an internal memory. RNN is recurrent in nature as it performs the same function for every input of data while the output of the current input depends on the past one computation. After producing the output, it is copied and sent back into the recurrent network. For planning, it considers the current input and the output that it has learned from the previous input.

Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition. In other neural networks, all the inputs are independent of each other. But in RNN, all the inputs are related to each other. First, it takes the $X(0)$ from the sequence of input and then its outputs $h(0)$ which together with $X(1)$ is the input for the next step. So, the $h(0)$ and $X(1)$ is the input for the next step. Similarly, $h(1)$ from the next is the input with $X(2)$ for the next step and so on. This way, it keeps remembering the context while training Figure 2.

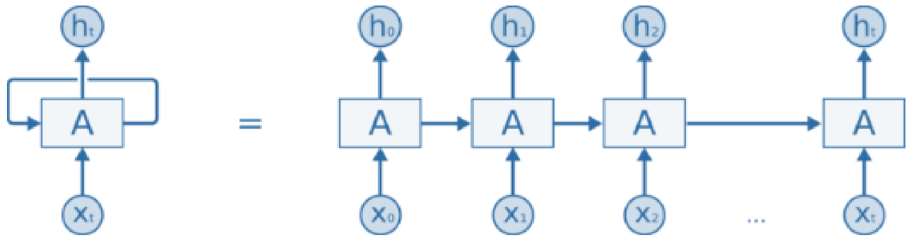

Figure 2. An unrolled recurrent neural network
Source: own elaboration with information from a machinelearningmastery.com

The formula for the current state is.

$$h_t = f(h_{t-1}, X_t) \tag{1}$$

Applying Activation Function:

$$h_t = \tanh(W_{hh} h_{t-1} + W_{Xh}, X_t) \tag{2}$$

W is weight, h is the single hidden vector, W_{hh} is the weight at previous hidden state, W_{xh} is the weight at current input state, tanh is the activation function, that implements a non-linearity that transforms the activations to the range [-1.1]

Output:

$$Y_t = W_{hY} h_t \tag{3}$$

Y_t is the output state.

From RNN to LSTM

In an LSTM network, three gates and four steps are present:

Step 1: Forget gate, decides what information to discard from the cell. It is decided by the sigmoid function. it looks at the previous state (h_{t-1}) and the content input and outputs (x_t) a number between 0 (omit this) and 1 (keep this) for each number in the cell state C_{t-1}.

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{4}$$

Step 2: Input gate, decides what values from the input to update the memory state. The tanh function decides which values to let through 0.1 And gives weightage to the values which are passed deciding their level of importance ranging from -1 to 1.

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \tag{5}$$

$$g_t = \tanh\,(W_c.[h_{t-1}, x_t] + b_g) \tag{6}$$

Step 3: Forget gate + input gate.

Update cell state

$$c_t = f_t * c_{t-1} + i_t * g_t \tag{7}$$

Step 4: Output gate, decides what to output based on input and the long-term memory of the cell. The tanh function decides which values to let through 0.1 and gives weightage to the values which are passed deciding their level of importance ranging from-1 to 1 and multiplied with output of Sigmoid Figure 3.

$$O_t = \sigma\,(W_o[h_{t-1}, x_t] + b_o) \tag{8}$$

$$h_t = o_t * \tanh(C_t) \tag{9}$$

The LSTM model is summarized by the two main function.

$$[f_t, i_t, o_t, g_t] = [\sigma, \sigma, \sigma, \tanh(c_t)] * w \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

(10)

$$c_t = f_t * c_{t-1} + i_t * g_t$$

(11)

$$h_t = o_t * \tanh(C_t)$$

(12)

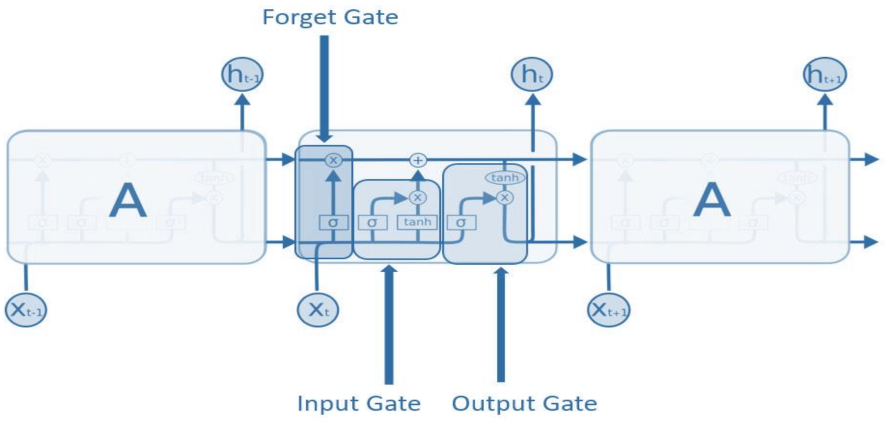

Figure 3. LSTM architecture
Source: own elaboration with information from a machinelearningmastery.com

Experimental setup

Deep learning is an algorithm that is elaborated in a programming language, the TensorFlow.Keras in Python 3 is utilized, within this API (Application Programming Interface) an artificial neural network is defined which is then converted into a set of commands that are executed on the computer. The components of the neural network that require intensive hardware resources are the processing of input data, the training of the deep learning model, the storage of the trained deep learning model and the deployment of the model.

Within all these, the training of the deep learning model is the most intensive task because two main operations are performed, one at the forward step and the other at the backward step.

In the forward step, the input is passed through the neural network and after processing the input, an output is generated. In the backward step, the neural network weights are updated based on the error obtained in the forward step. Both operations are essentially matrix multiplications.

As deep learning requires considerable hardware to efficiently execute these large matrix multiplications, this work uses the Graphics Processing Unit (GPUs)[1] from Google Collaboratory, a virtual processor from Google to enhances the models. A GPU can contain from 1 000 to 4 000 cores specialized in data processing; this high density of cores allows the GPU to have a high level of parallelism that allows it to execute many computations at the same time.

Data preparation and LSTM model building

Dataset

The selected methods were implemented on a dataset of real power load. The dataset contains 30-minute metered real power data for Newcastle substations in the Ausgrid supply area from January 1st, 2014, to December 31st, 2016, in annual sets. The data is taken directly from the original Ausgrid zone substation dataset but is reformatted here to adhere to the consistent NEAR-WESCML data format for zone substation data. The standard provides a consistent view of zone substation data across distribution businesses like TransGrid network.

TransGrid carries bulk electricity from generators through high voltage transmission lines, underground cables, and substations. The high voltage electricity is then converted to low voltage electricity suitable for household consumption at substations closer to power users. Distribution networks such as Ausgrid, Endeavour Energy and Essential Energy deliver electricity through smaller poles and wires to more than 3 million homes and businesses throughout New South Wales (NSW) and the ACT, figure 4. Newcastle substation is next to Killingworth in NSW, Australia. Ausgrid commissions the Newcastle substation in the NSW area supplying homes and businesses in Seahampton, Rhondda, Holmesville, Barnsley, Killingworth, Holmesville, Teralba, West Wallsend (Ausgrid, 2019).

Low demand very often has a trend due to organic growth highly related to gross domestic product, GDP, economic development for that reason we only keep the data for three years which is not many samples for us to keep our data set as stationary as possible for a better training and prediction.

[1] GPU is a co-processor dedicated to graphics processing or floating-point operations, to lighten the load on the central processor for applications such as gaming or interactive 3D applications. So, while much of the graphics processing is done on the GPU, the central processing unit (CPU) can focus on other computations such as artificial intelligence.

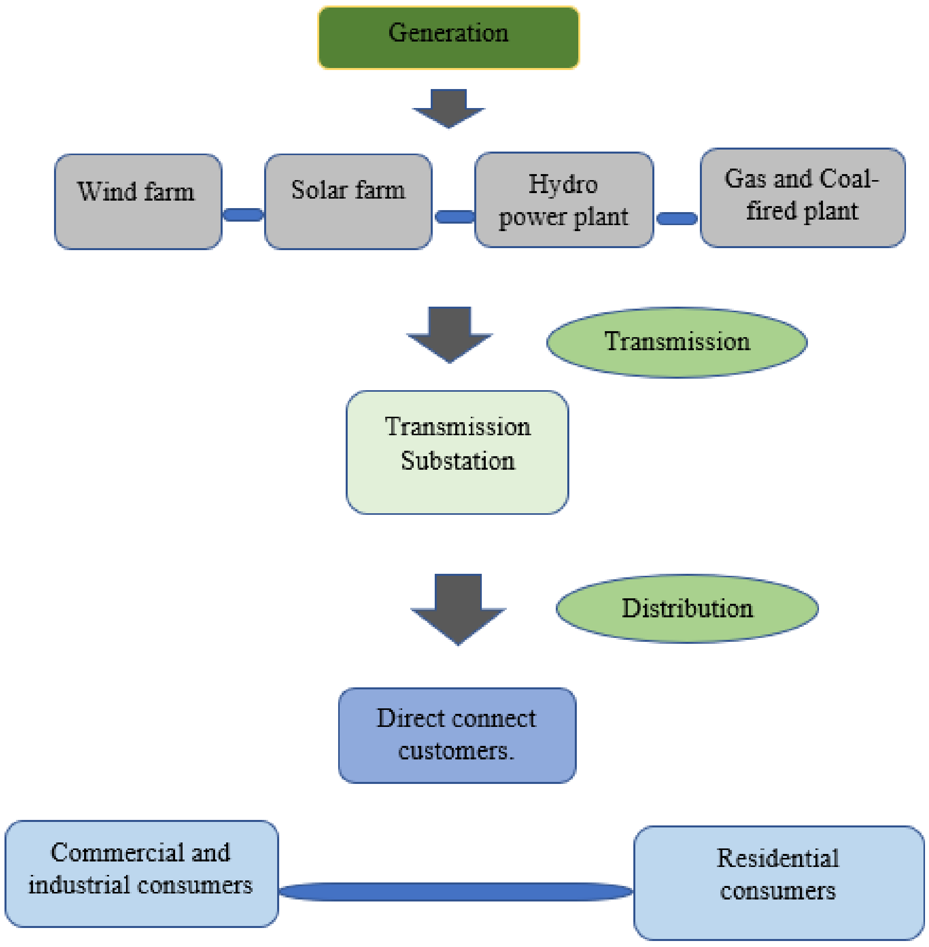

Figure 4. Transgrid network
Source: own elaboration with information from a transgrid.com.au

Data preparation

Developing a LSTM is very similar to developing RNN, there are still several key differences that we need to be aware of.

The data for our sequence prediction problem needs to be scaled when training a neural network, such as LSTM. When a network is fit on unscaled data that has a range of values [2] it is possible for large

[2] quantities in the 10s to 100s.

inputs to slow down the learning and convergence of your network and in some cases prevent the network from effectively learning your problem. There are two types of scaling of our series which include normalization and standardization. In the presented paper we use normalization. Normalization is a rescaling of the data from the original range so that all values are within the range of 0 and 1, requires that we know or can accurately estimate the minimum and maximum observable values. We may be able to estimate these values from the available data. If your time series has an upward or downward trend, estimating these expected values may be difficult and normalization may not be the best method to use for your problem.

A value is normalized as follows:

$$\mathrm{y} = (\mathrm{x} - \min) / (\max - \min) \tag{13}$$

In Python we can use Sklearn in MinMax scaler functions to perform the normalization. First, we need to fit the scaler using available training data estimate the minimum and maximum transform and normalize the data.

The second commonly used method is standardization. Standardization of a data set consists of rescaling the distribution of values so that the mean of the observed values is 0 and the standard deviation is 1. This can be thought of as subtracting the mean value or centring the data.

Like normalization, standardization can be useful, and even necessary, in some machine learning algorithms when the data have input values with different scales.

Normalization assumes that our observations fit a Gaussian distribution (bell curve) with a well-behaved mean and standard deviation. We can standardise our time series data if this expectation is not met, but we may not get reliable results.

Standardization requires that we know or can accurately estimate the mean and standard deviation of the observable values. We may be able to estimate these values from your training data.

A value is standardized as follows:

$$\mathrm{y} = (\mathrm{x} - \mathrm{mean}) / \mathrm{standard\_deviation} \tag{14}$$

$$\mathrm{mean} = \mathrm{sum(x)} / \mathrm{count(x)} \tag{15}$$

$$\mathrm{standard}_{\mathrm{deviation}} = \mathrm{sqrt}\left(\frac{\mathrm{sum}(\,(\mathrm{x} - \mathrm{mean})^2)}{\mathrm{count(x)} - 1}\right) \tag{16}$$

Deep learning libraries assume a retro eye on the representation of our data and assume that the input sequence of all features has the same length. The input to the LSTM must have a three-dimensional form consisting of:

- Samples, which is usually the number of rows in the dataset.
- Time steps, which are the past observations of a feature.
- Features, which are the columns of the dataset.

Case study

For the three years of our study, we have a sample of 52 560 data with an interval of 30 minutes. as we want to make forecasts in the hours ahead, we convert the data with an hourly interval, and we are left with a sample of 26 280 data of real power load.

From the original data set, we extract the first 98% what is about 1 075 days data for the training set and the last 2% data what is about 20 days for the test set.

To improve the performance of deep neural models hyperparameter tuning was done firstly on single machine chosen 1 configuration at a time. Then each different configuration was trained parallel on an individual node. Using Spark for computation we find the best set of hyper-parameters for LSTM training. Table 1 shows the summary of the sequential model with 50 neurons, table 2 gives us some metrics of the different configurations of our model with the training set and table 3 gives us the results of the test set with the 50 neurons configuration, which is the configuration that has the best MAE result with the training set.

The experiment was conducted using LSTM network to predict few next hours load based on last 24 hours. The results show that it is an easy task for the LSTM network architecture, and hence give low error ratios with the test dataset.

Table 1
Sequential LSTM model summary with 50 neurons

Layer (type)	Output Shape	Param #
Lstm (LSTM)	(None, 24, 50)	10 400
Lstm_1 (LSTM)	(None, 50)	20 200
Dense (Dense)	(None, 50)	2 550
Dense_1 (Dense)	(None, 1)	51
Total params: 33 201		
Trainable params: 33 201		
Non-trainable params: 0		

Source: own elaboration in Python with data from Ausgrid

Table 2
Error for various configuratins of LSTM network models on the training set

Configuration	Hidden layers	units	epochs	MAE	MSE	MAPE	Val MAE
1	2	128	135	0.0051	7.2320e-05	2.5534	0.0116
2	2	50	193	0.0050	6.2276e-05	2.4550	0.0121
3	2	30	186	0.0051	7.1659e-05	2.6321	0.0113
4	2	20	184	0.0055	7.9212e-05	2.7540	0.114

Source: own elaboration in Python with data from Ausgrid

Table 3
Errors of the LSTM network model on the test set

configuration	Hidden layers	units	epochs	MAE	MSE	MAPE	Val MAE
2	2	50	155	0.0051	7.2928e-05	2.4934	0.0112

Source: own elaboration in Python with data from Ausgrid

ARIMA

With the goal of comparing the performance of the LSTM neural network against a statistical forecasting method, in this section we apply ARIMA methodology to the load series described in previous sections.

Table *4*. shows that the levels of the time series are stationary at a level of 95% statistical confidence.

Table 4
Dickey Fuller stationarity test

Results of Dickey-Fuller Test:	
Test Statistic	-6.022480e+00
p-value	1.482397e-07
#Lags Used	3.800000e+01
Number of Observations Used	9.324000e+03
Critical Value (1%)	-3.431052e+00
Critical Value (5%)	-2.861850e+00
Critical Value (10%)	-2.566935e+00
dtype:	float64
The timeseries is stationary at 95% level of confidence	

Source: own estimation with statsmodels

Figure 5 shows the autocorrelation and partial autocorrelation functions, ACF and PACF. The results suggest that the time series is AR(p).

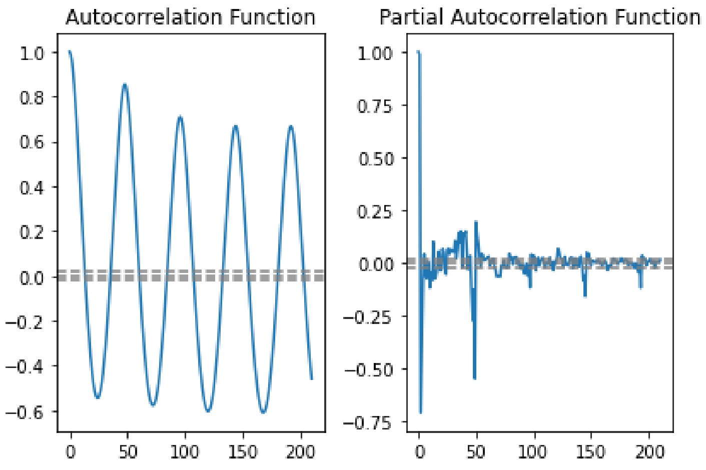

Figure 5. ACF and PACF
Source: own estimation with statsmodels

We use the following function in order to determine the order, p, that minimizes AIC:

```
%%timeit
AIC = {}
best_aic, best_order = np.inf, 0

for i in range(1,30):
   model = ARIMA(endog= temp, order=(i,0,0))
   results_AR = model.fit(disp=-1)
   AIC[i] = results_AR.aic

   if AIC[i] < best_aic:
      best_aic = AIC[i]
      best_order = i

print('BEST ORDER', best_order, 'BEST AIC:', best_aic)
```

The function indicates that the “Best order” is 29: BEST ORDER 29 BEST AIC: 16890.5838. We used 90% of the data to train the model. The complete model is included in the Annex.[3]

Figure 6 shows the actuals and fitted values, and the 95% confidence interval.

[3] It is worth noting that the function is very time consuming.

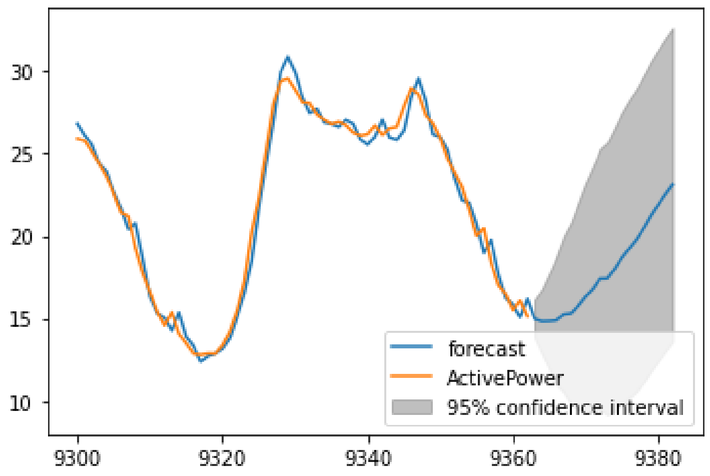

Figure 6. Actuals, fitted and 95% confidence interval
Source: own estimation with statsmodels

The out of sample MSE is 0.3547, which exceeds the MSE's attained by the LSTM neural networks.

Conclusions

The city of Newcastle, like many cities around the world, is constantly growing in terms of industry, homes, and businesses, and it is these elements that consume all the electrical energy produced in this locality. With this upward shift, the electricity consumption needs of this society must be met in any case. To solve this electricity demand problem, in this work a LSTM model was applied to the electricity load demand data at the Newcastle substation to forecast the demands in the next hours to come. The results obtained with the model with a configuration of 50 neurons, 2 hidden layers, present a MAE of 0.0050 with a training set and 0.0051 with a test set. These results show us that the deep learning model, which has been shown to be helpful in several areas, can help to improve the quality of predictions in the electricity demand area, since the very nature of the demand problem can be a limiting factor, and our LSTM model has the advantage of accepting many factors related to the input energy demand to improve the predictions.

References

Abdollah, K., & Mohammad-Reza, A. (2013). A hybrid method based on wavelet, ANN and ARIMA model for short-term load forecasting. Journal of Experimental & Theoretical Artificial Intelligence, 167-182.

Ausgrid. (2018). Distribution and Transmission Annual Planning Report. Sydney: Ausgrid.

Ausgrid. (2019). Distibution and Transmission Annual Planning Report. Sydney.

Bishnu, N., Motoi, Y., Aya, Y., & Toshiya, Y. (2019). Electricity load forecasting using clustering and ARIMA model for energy management in buildings. Japan Architectural Review .

Bo-Juen, c., & Ming-Wei, C. (2004). Load forecasting using support vector machines:A study on EUNITE competition 2001. IEEE transactions on power systems, 1821-1830.

Dietterich, T. (1995). Overfitting and undercomputing in machine learning. ACM Computing Surveys, 27(3), 326-327. doi:https://doi.org/10.1145/212094.212114

Franco, A. C. (2019). Deep Learning. Sevilla: universidad de Sevilla.

Hong, C., C., C., & A., S. (2002). ANN-based short-term load forecasting in electricity markets. IEEE Winter Meeting Power Engineering Society, 2, 411-415.

Jabbar, H. K., & Khan, R. Z. (2014). Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study). Jabbar, H. K., & Khan, R. Z. (2014). Computer Science, Communication and Instrumentation Devices.

Norizan, M., Maizah, A., Zuhaimy, I., & Suhartono, S. (2010). Short-Term Load Forecasting using double seasonal ARIMA model. Statistics Faculty of Computer and Mathematical Sciences.

Peter J., B., & Richard A., D. (2016). Introduction to Time Series and Forecasting. Switzerland: Springer Texts in Statistics.

Ping-Feng, P., & Wei-Chiang, H. (2005). Support vector machines with simulated annealing algorithms in electricity load forecasting. Energy conversion and management, 2669-2688.

Piotrowski, A. P., & Napiorkowski, J. J. (2013). A comparison of methods to avoid overfitting in neural networks training in the case of catchment runoff modelling. Journal of Hydrology, 476, 97–111. doi:https://doi.org/10.1016/j.jhydrol.2012.10.019

Salman, S., & Liu, X. (2019). Overfitting mechanism and avoidance in deep neural networks. In arXiv [cs.LG]. doi:http://arxiv.org/abs/1901.06566

Schittenkopf, C., Deco, G., & Brauer, W. (1997). Two strategies to avoid overfitting in feedforward networks. Neural Networks: The Official Journal of the International Neural Network Society, 10(3), 505–516. doi:https://doi.org/10.1016/s0893-6080(96)00086-x

Tetko, I. V., Livingstone, D. J., & Luik, A. I. (1995). Neural network studies. 1. Comparison of overfitting and overtraining. Journal of Chemical Information and Computer Sciences, 35(5), 826–833. doi:https://doi.org/10.1021/ci00027a006

Wang, Y., Chen, Q., Hong, T., & Kang, C. (2018). Review of smart meter data analytics: Applications, methodologies, and challenges. IEEE Transactions on smart Grid.

Weicong, K., & Yan, X. (2019). Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network. IEEE Transactions on smart grid.

Yunita, S. (2018). Neural and Non-neural Approaches to AuthorshipAttribution. Department of Computer ScienceThe University of Sheffield.

Annex

ARMA Model Results

Dep. Variable:	ActivePower	No. Observations:	9363
Model:	ARMA(29, 0)	Log Likelihood	-8414.292
Method:	css-mle	S.D. of innovations	0.594
Date:	Sat, 08 Oct 2022	AIC	16890.584
Time:	16:12:42	BIC	17112.064
Sample:	0	HQIC	16965.799

	coef	std err	z	P>|z|	[0.025	0.975]
const	20.9291	0.179	116.876	0.000	20.578	21.280
ar.L1.ActivePower	1.3666	0.010	132.367	0.000	1.346	1.387
ar.L2.ActivePower	-0.0172	0.018	-0.982	0.326	-0.052	0.017
ar.L3.ActivePower	-0.3138	0.018	-17.922	0.000	-0.348	-0.279
ar.L4.ActivePower	-0.1981	0.018	-11.127	0.000	-0.233	-0.163
ar.L5.ActivePower	0.1467	0.018	8.187	0.000	0.112	0.182
ar.L6.ActivePower	0.0875	0.018	4.878	0.000	0.052	0.123
ar.L7.ActivePower	-0.1492	0.018	-8.310	0.000	-0.184	-0.114
ar.L8.ActivePower	-0.0626	0.018	-3.479	0.001	-0.098	-0.027
ar.L9.ActivePower	0.1669	0.018	9.268	0.000	0.132	0.202
ar.L10.ActivePower	-0.0040	0.018	-0.223	0.824	-0.039	0.031
ar.L11.ActivePower	0.0504	0.018	2.794	0.005	0.015	0.086
ar.L12.ActivePower	-0.2877	0.018	-16.006	0.000	-0.323	-0.252
ar.L13.ActivePower	0.1834	0.018	10.077	0.000	0.148	0.219
ar.L14.ActivePower	0.0644	0.018	3.525	0.000	0.029	0.100
ar.L15.ActivePower	0.0018	0.018	0.096	0.923	-0.034	0.038
ar.L16.ActivePower	-0.0987	0.018	-5.402	0.000	-0.135	-0.063
ar.L17.ActivePower	-0.0666	0.018	-3.660	0.000	-0.102	-0.031
ar.L18.ActivePower	0.1452	0.018	8.076	0.000	0.110	0.180
ar.L19.ActivePower	0.0261	0.018	1.447	0.148	-0.009	0.061
ar.L20.ActivePower	-0.1509	0.018	-8.368	0.000	-0.186	-0.116
ar.L21.ActivePower	-0.0107	0.018	-0.594	0.552	-0.046	0.025

ar.L22.ActivePower	0.1038	0.018	5.763	0.000	0.069	0.139
ar.L23.ActivePower	0.0733	0.018	4.077	0.000	0.038	0.109
ar.L24.ActivePower	-0.1174	0.018	-6.531	0.000	-0.153	-0.082
ar.L25.ActivePower	-0.0022	0.018	-0.125	0.901	-0.037	0.033
ar.L26.ActivePower	0.0307	0.018	1.723	0.085	-0.004	0.066
ar.L27.ActivePower	-0.0451	0.018	-2.570	0.010	-0.079	-0.011
ar.L28.ActivePower	-0.0124	0.018	-0.708	0.479	-0.047	0.022
ar.L29.ActivePower	0.0556	0.010	5.382	0.000	0.035	0.076

Roots				
	Real	Imaginary	Modulus	Frequency
AR.1	1.0040	-0.1346j	1.0130	-0.0212
AR.2	1.0040	+0.1346j	1.0130	0.0212
AR.3	1.0552	-0.0000j	1.0552	-0.0000
AR.4	0.9999	-0.3356j	1.0547	-0.0515
AR.5	0.9999	+0.3356j	1.0547	0.0515
AR.6	0.9272	-0.5355j	1.0708	-0.0834
AR.7	0.9272	+0.5355j	1.0708	0.0834
AR.8	0.7713	-0.7396j	1.0686	-0.1217
AR.9	0.7713	+0.7396j	1.0686	0.1217
AR.10	0.5642	-0.9369j	1.0937	-0.1637
AR.11	0.5642	+0.9369j	1.0937	0.1637
AR.12	0.2988	-1.0313j	1.0737	-0.2051
AR.13	0.2988	+1.0313j	1.0737	0.2051
AR.14	0.2818	-1.2442j	1.2757	-0.2145
AR.15	0.2818	+1.2442j	1.2757	0.2145
AR.16	-0.1335	-1.0576j	1.0660	-0.2700
AR.17	-0.1335	+1.0576j	1.0660	0.2700
AR.18	-0.4024	-1.0035j	1.0812	-0.3107
AR.19	-0.4024	+1.0035j	1.0812	0.3107
AR.20	-0.6490	-0.9082j	1.1162	-0.3488
AR.21	-0.6490	+0.9082j	1.1162	0.3488
AR.22	-0.7977	-0.7531j	1.0971	-0.3796
AR.23	-0.7977	+0.7531j	1.0971	0.3796
AR.24	-1.1257	-0.1809j	1.1401	-0.4746
AR.25	-1.1257	+0.1809j	1.1401	0.4746
AR.26	-1.0589	-0.3705j	1.1218	-0.4464
AR.27	-1.0589	+0.3705j	1.1218	0.4464
AR.28	-1.0961	-0.5976j	1.2484	-0.4206
AR.29	-1.0961	+0.5976j	1.2484	0.4206