Development of an Encoder-Decoder Long Short-Term Memory Network To Forecast A Single Household's Weekly Energy Consumption

line 1: Brogan McShane
line 2: *School of Computing*
line 3: *Ulster University*
line 4: Belfast, Northern Ireland

***Abstract*—The following paper covers details of Deep Learning, Recurrent Neural Networks, Long Short-Term Memory models, and the weaknesses that Recurrent Neural Networks posess which the LSTM model attempts to improve upon. The paper also discusses the development, evaluation, and results of an Encoder-Decoder LSTM Model to address the many-to-many type sequence problem of forecasting a household's weekly energy consumption.**

I. Introduction

In 2004, John McCarthy, a key founder in the discipline of Artificial Intelligence, defined this sub-domain of Computer Science in the paper "What Is Artificial Intelligence" as the Science and Engineering of making intelligent machines, especially intelligent Computer Programs [1]. When asked to describe what qualifies a machine to be truly intelligent, McCarthy elaborated that an intelligent machine can achieve goals in the world [1].

Deep Learning, a sub-field of Artificial Intelligence, has seen increased popularity in recent years for its ability to out-perform Machine Learning models in Natural Language Processing, Fraud Detection, Speech Recognition and Object Detection. The Deep Learning architecture is capable of its increased accuracy (when compared to Machine Learning models) through the use of neural networks, which consist of a series of multiple layers of interconnected nodes, tasked to learn representations of data with multiple layers of abstraction, where each layer builds upon the previous layer to refine and optimize the output of the model [2].

The typical Deep Learning Neural Network consists of an input layer, several hidden layers, and an output layer, each containing their own number of nodes. Every node will have an associated connection weight and is connected to other nodes on the previous and next layer. These connection weights, as well as the strengths of each node are obtained by a process of learning from a set of training patterns [3]. If a single node on any layer has an output above the specified weight, this node will be activated – the activation will send data to the next layer of the network [4]. This results in the output of one node becoming the input of the next node - a process that continues until the output layer is reached, where the output of the computational model is presented [4].

II. Recurrent Neural Networks

A Recurrent Neural Network is a particular Deep Learning Architecture that works with sequential or time-series data to predict the next element of a sequence [5]. Where traditional Deep Learning Networks assume that input data and output data are independent of each other, the output of a recurrent neural network is based on analysis of all prior elements within the sequence [5]. Recurrent Neural Networks are distinguishable from other networks for their "memory" capability, as the neural network will take information from prior inputs to influence the current input and output [5].

Although Recurrent Neural Networks are the dominant architecture for sequential modelling, they often struggle with long-term dependencies (i.e. the RNN may not be able to accurately predict the current state when the previous state influencing this prediction is not in the recent past) and are prone to the problem of vanishing gradients [6]. To address these issues, Sepp Hochreiter and Juergen Schmidhuber introduced the Long Short-Term Memory Architecture.

A. Long Short-Term Memory Network

A Long Short-Term Memory network is a variant of the Recurrent Neural Network which is capable of learning long-term dependencies and is a solution to the vanishing gradient problem that can occur when training traditional Recurrent Neural Networks [7].

The LSTM is capable of overcoming the long-term dependency problem by adding an internal state to the RNN node, meaning that when an RNN input is received, it receives the output of the previous step, the input of the current step as well as state information from the LSTM state [8]. This LSTM state is commonly referred to as an LSTM cell, which consists of three gates: a forget gate (which determines what information in the internal state is no longer contextually relevant), an input gate (which determines what new information should be added or updated in the internal state) and an output gate (to determine which part of the internal state should be output in this instance) [8].

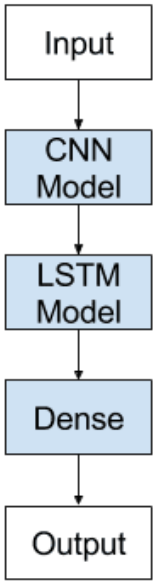
1) Different LSTM Architectures

a) CNN LSTM Model

One of the most popular deep neural networks is the Convolutional Neural Network (CNN), which takes its name from the mathematical linear operation between matrixes called convolution [9]. The CNN features Convolutional Layers which take input, perform convolution operations on the input data and then outputs the convoluted data to the next layer [10].

This Neural Network has an excellent performance in machine learning operations, such as pattern recognition, image classification and object detection, therefore making it a popular neural network for computer vision applications [11].

A CNN model can be used in conjunction with an LSTM backend where the CNN is used to interpret sub-sequences of input that together are provided as a sequence for an LSTM model to interpret [12]. The CNN LSTM model was originally developed for visual time series prediction problems and the application of generating text descriptions for videos, images, and activity in a sequence of images [12].



Basic Illustration of the CNN LSTM Model (Source: https://machinelearningmastery.com/wp-content/uploads/2017/07/Convolutional-Neural-Network-Long-Short-Term-Memory-Network-Archiecture.png)

b) Encoder-Decoder LSTM Model

The Encoder-Decoder LSTM was designed to address problems that involve taking a sequence as an input and requires a sequence prediction as an output (commonly known as sequence-to-sequence prediction)[13].

This variant of the LSTM model consists of two recurrent neural networks that act as an encoder and decoder pair: the encoder will map a variable-length source sequence to a fixed-length vector, and the decoder maps the vector representation back to a variable-length target sequence [14].

The Encoder-Decoder LSTM was originally developed for machine translation and showed to improve overall translation performance when compared to a typical RNN [15]. Experiments outside of machine translation (such as predicting a vehicle's trajectory) have been conducted that also show that this variant of the LSTM increases accuracy when compared to typical LSTM models when facing sequence-to-sequence problems [16]

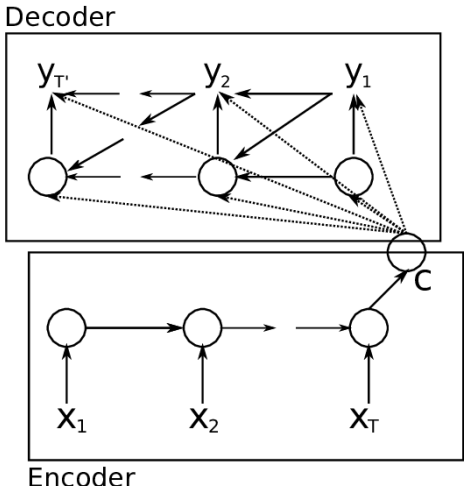


Illustration of the Encoder-Decoder LSTM (Source: https://www.researchgate.net/figure/An-illustration-of-the-proposed-RNN-Encoder-Decoder_fig5_262877889)

III. Using an LSTM Model to Predict a Household's Energy Consumption

Using a dataset that contains the measure of a household's energy consumption (recorded each minute of every day between December 2006 and November 2010), the dataset could be grouped into weeks and an LSTM model can be developed to determine how much energy this household would consume for every day over the next coming week(s). As discussed previously, the Encoder-Decoder LSTM would fit this particular problem for its effectiveness in sequence prediction.

A. Preparing the Dataset

When preparing the dataset to consist of daily data, it must be noted that upon initial investigation, there were 25979 lines which contained missing data (marked with a "?" value instead of the appropriate measure of energy consumption). To address this issue, missing values would be replaced with the previous minute's correlating value.

Once the data had been cleansed and grouped into days, this new dataset would be split into a training dataset containing the first three years of records (grouped by weeks) and a

testing dataset containing the final year of records (also grouped by weeks). One problem in doing this is that the training dataset now only has 159 records, which will affect the accuracy of the LSTM model. In order to overcome this problem, an algorithm would be written in order to increase the training dataset from the original 159 records to 1,100 records by using the concept of a "overlapping moving window", illustrated below.

Day1	Day2	Day3	Day4	Day5	Day6	Day7
Day8	Day9	Day10	Day11	Day12	Day13	Day14

Original Training Set's First Two Records

Day1	Day2	Day3	Day4	Day5	Day6	Day7
Day2	Day3	Day4	Day5	Day6	Day7	Day8
Day3	Day4	Day5	Day6	Day7	Day8	Day9

Training Dataset Reformatted to an Overlapping Moving Window to Increase Training Data (First Three Records)

B. *Designing and Developing the LSTM Network*

Discussed beforehand, the appropriate LSTM network to use consists of encoder and decoder models. Experimentation would be conducted as to determine the most effective number of neurons to make up both the encoder and decoder, the most effective action function, the appropriate loss function and which optimizer would produce the best results. In addition to this, different batch sizes and epochs would also be evaluated as to which combination would provide the best results.

1) *Discussing the LSTM Architecture to be Adopted*

To further discuss the LSTM architecture, the model begins with declaring an encoder LSTM to read the input sequence and output a vector of n elements.

After this, in order to connect the encoder to the decoder, a RepeatVector layer is used to repeat the encoder's 2D array of outputs in order to transform it into the appropriate format for the decoder of the LSTM (which expects a 3D input) [13].

Next, we move onto the decoder layer, which consists of the same amount of neurons in the encoder layer, to map the vector representation back to a variable-length target sequence.

In addition to this, before the output layer is reached in the encoder-decoder LSTM, the model would also include a dense TimeDistributed layer to interpret each time step in the output sequence, allowing both the dense layer and output layer to process each time step provided by the decoder.

2) *Discussing the LSTM's Activation functions*

Typical LSTMs will use the tanh activation function for the activation of the cell state and the sigmoid activation function for the node output. However, after experimenting with different activation functions and their effect on the RMSE (Root mean squared error), it was determined that the ReLU activation function provides results with the smallest RMSE. In addition to this, the linear activation function would be used as the output layer's activation function.

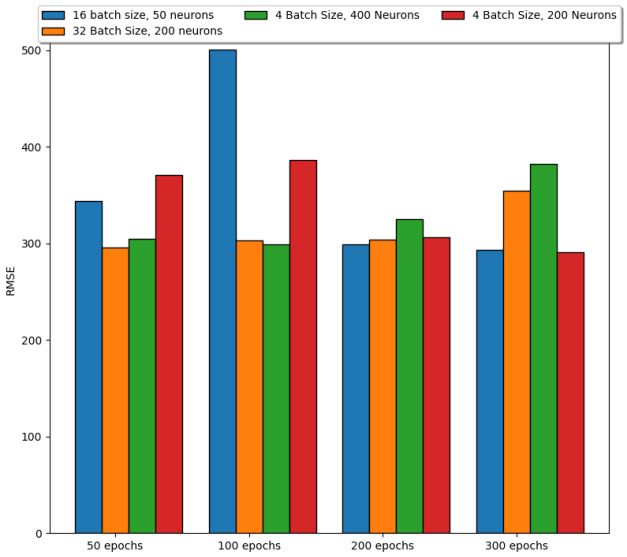
3) *Discussing the LSTM's Loss Function and Optimizer*

After determining the appropriate activation functions, the loss function chosen when compiling the model would be MSE (mean squared error), which calculates the average of the squared differences between the predicted and actual values. An advantage of using this loss function is that it will ensure that the trained model has no outlier predictions with large errors. Furthermore, this loss function is a good match for the RMSE error metric. And, in addition to this, the Adam optimizer would be used.

C. *Analysing The Performance Of Different Configurations of LSTM Models*

To evaluate different configurations of LSTM networks in order to determine which performs the best, neurons making up the encoder and decoder would be set to 50, then increased to 200 and finally increased to 400, with results of the predictions being measured through calculating the RMSE of the predicted power consumption measured in kilowatts. In addition to different sizes of neurons, various batch sizes and epoch values would be considered to further determine the best configuration for the LSTM model.

The output of the discussed experiments can be seen in the diagram below.

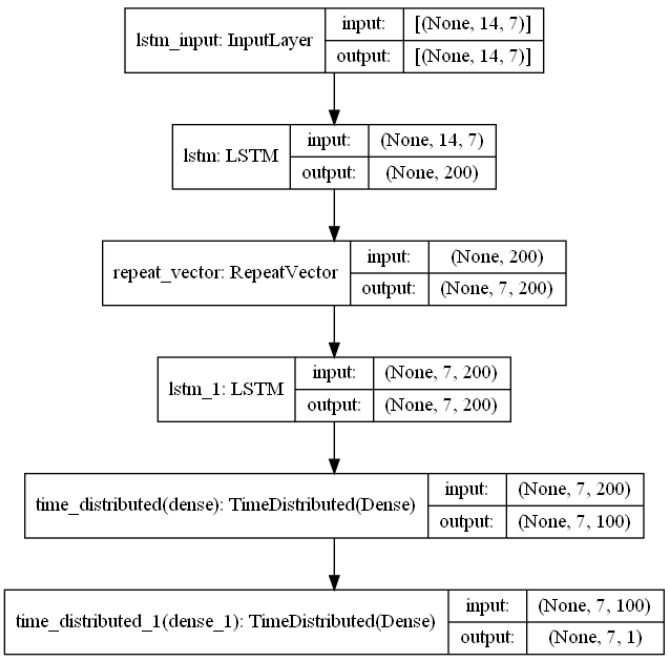


Using this information, it is determined that an LSTM model where the encoder and decoder layers consist of 200

neurons each, the batch size is set to 4 and 300 epochs are used to train the model is the configuration which produces the lowest RMSE of 290 kilowatts for predicting the household's first week of power consumption in the final year of the dataset. The implementation of this code (as well as the description for every layer) and the visualization of the keras model can be seen below.

```
def build_multi_input_encoder_decoder(train_x, train_y, verbose, epochs, batch_size):
    '''Function that will build and return a multiple input encoder-decoder LSTM
    The model consists of two sub-models
        The encoder - which reads and encodes the input sequence
        The decoder - to read the encoded input sequence
        Using this model, we will be able to forecast the power consumption
        for the next n amount of days'''

    '''train_y must be reshapedto have a three dimensional structure instead of the already
    existing structure of [samples, features]
    train_y's new three dimensional shape is to match the single week prediction shape of [1,7,1]'''
    train_y = train_y.reshape((train_y.shape[0], train_y.shape[1], 1))
    #Initialize model
    model = Sequential()
    ''' Define a hidden LSTM Encoder layer with 200 units
            -The encoder model will read the input sequence and outputs a 200 element vector
            (with one node per unit) that captures features from the input sequence
        The input shape will be defined by the numeric value passed by the train_x[1] and train_x[2] shapes'''
    model.add(LSTM(200, activation='relu', input_shape=(train_x.shape[1], train_x.shape[2])))
    ''' Next, the representation of the input sequence (train_x) will be repeated for each time step in the
        output sequence (train_y) by adding a RepeatVector layer to the model'''
    n_outputs = train_y.shape[1]
    model.add(RepeatVector(n_outputs))
    ''' Define the decoder layer with 200 units
        Decoder will output the entire sequence and not just the output at the end of the sequence as
        seen when defining the encoder layer. This means each unit in the decoder layer will output a
        value for each day to be forecasted which represents the basis for  what to predict for each day
        in the output sequence  '''
    model.add(LSTM(200, activation='relu', return_sequences=True))
    ''' model.add(TimeDistributed(Dense(100, activation="relu"))) does the following:
        The TimeDistributed layer will interpret each time step in the output sequence (train_y) '''
    model.add(TimeDistributed(Dense(200, activation='relu')))
    #Output layer
    model.add(TimeDistributed(Dense(1, activation="linear")))
    '''MSE has been chosen as the loss function as it will be used to ensure that the trained model has no
        outlier predictions with huge errors '''
    model.compile(loss='mse', optimizer='adam')
    #Train model
    model.fit(train_x, train_y, epochs=epochs, batch_size=batch_size, verbose=verbose)
    return model
```



D. Evaluating Results of The Chosen LSTM Model

As discussed previously, the configuration of the LSTM model with the lowest RMSE consisted of 200 neurons for the encoder layer with a ReLU activation function, a layer consisting of a repeat vector, a decoder layer with 200 neurons with a ReLU activation function, a dense TimeDistributed layer to interpret each time step in the output sequence and an output layer.

The below diagram shows the prediction of the first week's power consumption compared against the actual consumption. In addition to this, for every predicted week, the MSE and the RMSE are included as a method of evaluating the model's performance.

```
Week 1
Day     Actual Consumption      Predicted Consumption
Sat     1309.27             1551.26
Sun     2083.45             1717.61
Mon     1604.13             1644.35
Tues        2219.78             1640.06
Weds        1777.18             1639.81
Thurs       1769.44             1639.8
Fri     1797.21             1639.8

Week 1 MSE: 84363.69 killowats
Week 1 RMSE: 290.45 killowats
```

Although this model has the lowest RMSE in predicting the first week's values, as more predictions are made, the RMSE and MSE values only increase, indicating that the LSTM model may not be as appropriate as initially believed

```
Week 1
Day     Actual Consumption      Predicted Consumption
Sat     1309.27             1551.26
Sun     2083.45             1717.61
Mon     1604.13             1644.35
Tues        2219.78             1640.06
Weds        1777.18             1639.81
Thurs       1769.44             1639.8
Fri     1797.21             1639.8

Week 1 MSE: 84363.69 killowats
Week 1 RMSE: 290.45 killowats

Week 2
Day     Actual Consumption      Predicted Consumption
Sat     2336.9              2067.95
Sun     2508.47             1968.0
Mon     1518.68             1720.55
Tues        1995.8              1811.38
Weds        1995.8              1797.2
Thurs       2116.22             1803.96
Fri     2196.76             1803.28

Week 2 MSE: 104425.57 killowats
Week 2 RMSE: 323.15 killowats
```

[1] J. McCarthy, "What is Artificial Intelligence?", *Homes.di.unimi.it*, 1998. [Online]. Available: https://homes.di.unimi.it/borghese/Teaching/AdvancedIntelligentSystems/Old/IntelligentSystems_2008_2009/Old/IntelligentSystems_2005_2006/Documents/Symbolic/04_McCarthy_whatisai.pdf

[2]IBM Education, "What is Deep Learning?", *Ibm.com*, 2020. [Online]. Available: https://www.ibm.com/cloud/learn/deep-learning

[3]K. Gurney and N. York, "An introduction o neural networks", 1997. [Online]. Available: http://www.macs.hw.ac.uk/~yjc32/project/ref-NN/Gurney_et_al.pdf.

[4]IBM Education, "What are Neural Networks?", *Ibm.com*, 2022. [Online]. Available: https://www.ibm.com/cloud/learn/neural-networks#toc-how-do-neu-vMq6OP-P

[5]IBM Education, "What are Recurrent Neural Networks?", *Ibm.com*, 2022. [Online]. Available: https://www.ibm.com/cloud/learn/recurrent-neural-networks

[6]Deeplearning.ai, "Recurrent Neural Networks (RNNs) and Vanishing Gradients", *Youtube.com*, 2022. [Online]. Available: https://www.youtube.com/watch?v=NgxMUHTJYmU

[7]J. Brownlee, "A Gentle Introduction to Long Short-Term Memory Networks by the Experts", *Machine Learning Mastery*, 2022. [Online]. Available: https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/

[8]IBM Technology, "What is LSTM (Long Short Term Memory)?", *Youtube.com*, 2022. [Online]. Available: https://www.youtube.com/watch?v=b61DPVFX03I

[9] Goodfellow, I., Bengio, Y. and Courville, A. (2016). Deep Learning. [online] Google Books. MIT Press. Available at: https://books.google.co.uk/books?hl=en&lr=&id=omivDQAAQBAJ&oi=fnd&pg=PR5&dq=Ian+Goodfellow+and+Yoshua+Bengio+and+Aaron+Courville&ots=MNP4eslGTV&sig=jqyp_EdQTkxY-CylaMnU4XFjr4&redir_esc=y#v=onepage&q=Ian%20Goodfellow%20and%20Yoshua%20Bengio%20and%20Aaron%20Courville&f=false [Accessed 9 Nov. 2021].

[10] Wimpee, Z. (2017). Convolutional Neural Networks (CNNs) explained. [online] www.youtube.com. Available at: https://www.youtube.com/watch?v=YRhxdVk_sI

[11] Albawi, S., Mohammed, T.A. and Al-Zawi, S. (2017). Understanding of a convolutional neural network. 2017 International Conference on Engineering and Technology (ICET). [online] Available at: https://ieeexplore.ieee.org/document/8308186/.

[12]J. Brownlee, "How to Develop LSTM Models for Time Series Forecasting", *Machine Learning Mastery*, 2022. [Online]. Available: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/

[13]J. Brownlee, "Encoder-Decoder Long Short-Term Memory Networks", *Machine Learning Mastery*, 2022. [Online]. Available: https://machinelearningmastery.com/encoder-decoder-long-short-term-memory-networks/

[14]M. Schuster and K. Paliwal, "Bidirectional Recurrent Neural Networks", *Maxwell.ict.griffith.edu.au*, 2022. [Online]. Available: https://maxwell.ict.griffith.edu.au/spl/publications/papers/ieeesp97_schuster.pdf

[15]K. Cho, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation", *Arxiv.org*, 2022. [Online]. Available: https://arxiv.org/pdf/1406.1078.pdf

[16] S. H. Park, B. Kim, C. M. Kang, C. C. Chung and J. W. Choi, "Sequence-to-Sequence Prediction of Vehicle Trajectory via LSTM Encoder-Decoder Architecture," 2018 IEEE Intelligent Vehicles Symposium (IV), 2018, pp. 1672-1678, doi: 10.1109/IVS.2018.8500658.