eISSN: 2637-1138
© Penerbit UMT

AN INVESTIGATION INTO THE PERFORMANCE OF THE MULTILAYER PERCEPTRON ARCHITECTURE OF DEEP LEARNING IN FORECASTING STOCK PRICES

ASSUNTA MALAR PATRICK VINCENT AND HASSILAH SALLEH *

Faculty of Ocean Engineering Technology & Informatics, Universiti Malaysia Terengganu, Kuala Nerus, Terengganu, Malaysia

**Corresponding author: hassilah@umt.edu.my*

Abstract: A wide range of studies have been conducted on deep learning to forecast time series data. However, very few researches have discussed the optimal number of hidden layers and nodes in each hidden layer of the architecture. It is crucial to study the number of hidden layers and nodes in each hidden layer as it controls the performance of the architecture. Apart from that, in the presence of the activation function, diverse computation between the hidden layers and output layer can take place. Therefore, in this study, the multilayer perceptron (MLP) architecture is developed using the Python software to forecast time series data. Then, the developed architecture is applied on the Apple Inc. stock price due to its volatile characteristic. Using historical prices, the accuracy of the forecast is measured by the different activation functions, number of hidden layers and size of data. The Keras deep learning library, which can be found in the Python software, is used to develop the MLP architecture to forecast the Apple Inc. stock price. The developed model is then applied on different cases, namely different sizes of data, different activation functions, different numbers of hidden layers of up to nine layers, and different numbers of nodes in each hidden layer. Then, the metrics mean squared error (MSE), mean absolute error (MAE) and root-mean-square error (RMSE) are employed to test the accuracy of the forecast. It is found that the architecture with rectified linear unit (ReLU) outperformed in every hidden layer and each case with the highest accuracy. To conclude, the optimal number of hidden layers differs in every case as there are other influencing factors.

Keywords: Deep learning, multilayer perceptron, forecasting stock price, ReLu, sigmoid, hyperbolic tangent

Introduction

Deep learning is a subfield of machine learning, in which features are discovered automatically. There are four major architectures of deep learning, which is also known as a deep neural network, namely Unsupervised Pre-trained Networks (UPNs), Convolutional Neural Networks (CNNs), Recurrent Neural Networks and Recursive Neural Networks, as represented in Figure 1 (Patterson & Gibson, 2017).

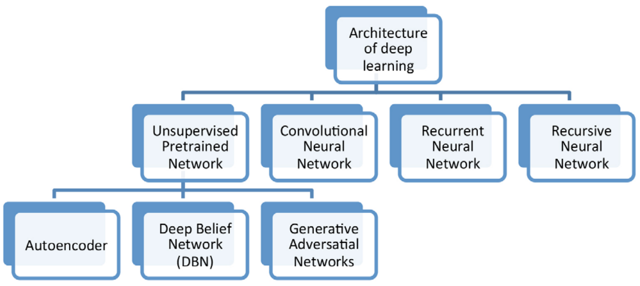
Architecture of deep learning
Unsupervised Pretrained Network
Convolutional Neural Network
Recurrent Neural Network
Recursive Neural Network
Autoencoder
Deep Belief Network (DBN)
Generative Adversatial Networks

Figure 1: The classification of deep learning architectures
(Source: Patterson & Gibson, 2017)

The multilayer perceptron (MLP) architecture is a part of the Deep Belief Network (DBN), which is classified under UPN. MLP is made up of three main layers, which are the input layer, arbitrary number of hidden layers and output layer. Figure 2 shows the architecture of MLP with the interconnecting nodes. The purpose of MLP is to forecast the target value with certain inputs (Liu *et al.*, 2017).

The performance of these architecture differs when the forecast is performed. Hiransha *et al.* (2018) has proven that MLP is better than other deep learning models, which are Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM), as they failed to identify the seasonal output of MARUTI stock. Apart from that, Naeini *et al.* (2010) concluded that feedforward multilayer perceptron performs better than the Elman recurrent network when the forecasting of a company's stock value is done based on historical data.

There are certain factors that must be considered in building the MLP architecture. First, the size of data. Hiransha *et al.* (2018) found that the bigger the size of dataset, the better the performance of the architecture. Next, the number of hidden layers. According to Ved (2019), increasing the number of hidden layers improves the forecast. However, Karsoliya (2012) stated that it has been theoretically proven that there can be a problem in training the architecture after the fourth layer. Apart from that, Karsoliya (2012) also stated that as a rule of thumb, the number of nodes in hidden layers are 2/3 the size of the input layer, and can be used to determine the number of hidden nodes in each layer. Thus, the optimal number of hidden layers and hidden nodes in each layer must be identified through trial and error.

Therefore, the purpose of this study is to develop the MLP architecture using Keras in Pthon to forecast time series data and analyse the influential factors of the developed architecture by applying it on historical prices of the Apple Inc. stock (AAPL). The influential factors considered are the activation function, number of hidden layers and size of data.

Methodology

The MLP architecture consists of three main layers, which are the input layer, arbitrary number of hidden layers and output layer, as shown in Figure 2. It is a feedforward network because the output of each layer feedforward to the next layer (Widegren, 2017).

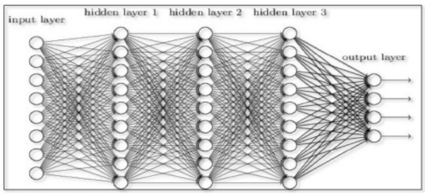

Figure 2: The MLP architecture
(Source: Upadhyay, 2019)

In this study, the number of hidden layers in the architecture varies and it goes up to nine hidden layers. Thus, equations 1 and 2 are generalised from equations 3 to 5 from a MLP network with two hidden layers.

$$\mathbf{h}^{(i)} = \mathrm{f}^{(i)}(W^{(i)T}\mathbf{h}^{(i-1)}+\mathbf{b}^{(i)}) \quad (1)$$

$$\mathbf{y} = f^{(N+1)}(W^{(N+1)T}\mathbf{h}^{(N-1)}+\mathbf{b}^{(N)}) \quad (2)$$

Where i = 2, 3, 4...N and N is the last hidden layer, $\mathbf{x} \in \mathbb{R}^{m_0}$ is the input vector, $\mathbf{y} \in \mathbb{R}^{m_3}$ is the output vector, $W^{(i)} \in \mathbb{R}^{m_{i-1} \mathrm{x} m_i}$ is the weight matrix in layer i, $\mathrm{b}^{(i)} \in \mathbb{R}^{m_i}$ is the bias term in layer i and $\mathrm{h}^{(i)} \in \mathbb{R}^{m_i}$ denotes the output of hidden layer i.

The output of the MLP network with two hidden layers and one output layer is represented as equations 3 to 5 (Widegren, 2017).

$$\mathbf{h}^{(1)} = \mathrm{f}^{(1)}(W^{(1)T}\mathbf{x}+\mathbf{b}^{(1)}) \quad (1)$$
$$\mathbf{h}^{(2)} = \mathrm{f}^{(2)}(W^{(2)T}\mathbf{h}^{(1)}+\mathbf{b}^{(2)}) \quad (2)$$
$$\mathbf{y} = \mathrm{f}^{(3)}(W^{(3)T}\mathbf{h}^{(2)}+\mathbf{b}^{(3)}) \quad (3)$$

The activation function is then applied on each hidden layer. The activation function has a "squashing" property, where it squashes the input value into a certain range depending on the type of activation function. Table 1 shows the representation of different activation functions and their squashing range.

Table 1: Activation functions

Activation function	Rectified linear unit (ReLU)	Sigmoid	hyperbolic tangent (tanh)
Equation	ReLU(a) = *max* (0, a)	sigmoid(a) = $\frac{1}{1+e^{-a}}$	tanh(a) = $\frac{2}{1+e^{-2a}}$ -1
Squashing Range	(0,∞)	(0,1)	(-1,1)

The collected data is allocated into two sets of data, which are training and testing sets. The training set consists of data that is used to fit the architecture. Its purpose is for the architecture to see and learn from the behavior of the data. Next, the testing set is used to evaluate the performance of the architecture (Shah, 2017).

To train the architecture, backpropagation and a gradient descent algorithm is used. First, all the weight and bias terms in equations 1, 4 and 5 are initialised to small random values nearer to zero. Then, it is applied on the activation function so that it can propagate forward. A backpropagation algorithm and MSE is used to train the architecture.

A gradient descent algorithm is used to optimise the network. The Adaptive Moment Estimation (Adam) is used to compute the adaptive learning rates for each weight. By considering and demonstrating the exponentially decaying average of the past gradient and past squared gradients, it can be shown that Adam works well for the adaptive learning method. To avoid bias in the optimisation algorithm, the training set is shuffled after each epoch (Xu *et al.*, 2017). Equation 8 is used to calculate MSE, with T being the number of datasets in the training set.

The error of each unit in the output layer is computed and then the errors of each hidden unit from the last to first hidden layer are computed. Before the update of the weight and bias terms, their increments are calculated. The network trains all the patterns repeatedly until the total error falls to some pre-determined low target value (MSE, loss function) and then it stops. The test data is used after the values of weight and biases have been determined, and they are applied to the test data (Amardeep & Swamy, 2017).

Finally, evaluation metrics are employed to evaluate the performance of the built architecture. The metrics are the mean absolute error (MAE), mean squared error (MSE) and root-mean-square error (RMSE). These criteria are preferred to be smaller as it reflects the error. The MAE, MSE and RMSE is shown in equations 6, 7 and 8, respectively.

$$MAE = \frac{\sum_{i=1}^{T} |\hat{y}_i - y_i|}{T} \tag{6}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{T} (\hat{y}_i - y_i)^2}{T}} \tag{7}$$

$$MSE = \frac{\sum_{i=1}^{T} (\hat{y}_i - y_i)^2}{T} \tag{8}$$

Where T is the total data in testing set.

Results and Discussion

The daily historical prices of the AAPL stock were obtained from Yahoo Finance. The collected stock prices are from 31 December 2012 to 28 December 2016. The Total data collected were 1261 data points. These data comprised daily open, high, low, and close prices of the stock (Apple Inc. (AAPL), 2019). The collected data is initially transformed into a matrix form. Then, it is divided into two sets (i.e. training and testing sets) and it is applied into four types of cases as shown in Table 2.

Table 2: Allocation of data for various cases

Types of case	Total data	Date of different set		Percentage of allocation	
		Training	Testing	Training	Testing
Case 1 (>1000 data)	1261	31/12/2012 to 31/12/2015	31/1/2015 to 29/12/2019	60%	40%
Case 2 (>1000 data)	1009		31/1/2015 to 28/12/2016	75%	25%
Case 3 (<1000 data)	631	3/7/2014 to 31/12/2015		60%	40%
Case4 (<1000 data)				60%	40%

Next, the number of hidden layers and nodes are determined using the rule of thumb method,where it follows the ratio 3:2 (i.e., previous number of hidden nodes: current number of hidden nodes).The number of input data is used as the initial value for the ratio 3:2, except for Case 3. Table 3 and Table 4 show the number of nodes in each hidden layer for Cases 1, 2 and 3, and Case 4, respectively.

Once the number of hidden nodes in each layer is determined, using any of the activation function stated in methodology, the processing layers are built. Then, to train the network, Adam is used to optimise the model by using MSE as a loss function.

Table 3: The number of nodes in each hidden layer for Cases 1, 2 and 3

Number of hidden layers	Number of nodes in the previous hidden layer	Number of nodes in the current hidden layer
1	757	505
2	505	337
3	337	225
4	225	150
5	150	100
6	100	67
7	67	45
8	45	30
9	30	20

Table 4: The number of nodes in each hidden layer for Case 4

Number of hidden layers	Number of nodes in the previous hidden layer	Number of nodes in the current hidden layer
1	378	252
2	252	168
3	168	112
4	112	75
5	75	50

6	50	33
7	33	22
8	22	15
9	15	10

To fit the architecture in this study, the number of epochs has been set 1000. Therefore, all the dataset passed forwards and backwards through the neural networks 1000 times. The number of epochs is not one, because one epoch is very big to load into the computer at once, thus it must be divided into a few small batches, where in this coding, it is randomly generated. Since, one epoch results in an underfitting curve, the number of epochs must be increased (Sharma, 2017).

Next, the data in the testing set is used to forecast the AAPL stock price and the performance of the architecture is evaluated by calculating the MAE, RMSE and MSE using Equations 6, 7 and 8, respectively. Once the architecture is built, different activation functions, as shown in Table 1, are used. Then the forecast is performed using the architecture with one to nine hidden layers containing different sizes of data as shown in Table 2. Tables 5 to 8 show the performance of the forecast for each case. The bolded values in Tables 5 to 8 represent the lowest value of MSE, MAE and RSME for each activation function. It can be observed that the ReLU activation function has the lowest value for all MSE, MAE and RMSE in all four cases.

Table 5: The performance evaluation for Case 1

A lot of data

No. of hidden layer	**ReLU**			**Sigmoid**			**Tanh**		
	MSE	**MAE**	**RMSE**	**MSE**	**MAE**	**RMSE**	**MSE**	**MAE**	**RMSE**
1	0.429	0.531	0.655	15.624	**2.614**	**3.953**	**26.320**	**3.302**	**5.130**
2	0.441	0.529	0.664	101.490	6.090	10.074	170.270	7.935	13.049
3	2.765	1.559	1.663	198.456	8.428	14.087	255.273	10.613	15.977
4	**0.261**	**0.393**	**0.510**	232.805	9.842	15.258	1824.354	34.120	42.712
5	0.771	0.744	0.878	257.469	10.240	16.046	1832.541	34.236	42.808
6	0.268	0.398	0.518	**9.981**	270.973	16.461	1834.946	34.270	42.836
7	0.344	0.469	0.587	1839.374	34.333	42.888	1834.045	34.257	42.826
8	0.277	0.404	0.526	1833.304	34.247	42.817	1833.149	34.245	42.815
9	0.552	0.618	0.743	1834.857	34.269	42.835	1832.302	34.233	42.805

Table 6: The performance evaluation for Case 2

Less data training set

No. of hidden layer	**ReLU**			**Sigmoid**			**Tanh**		
	MSE	**MAE**	**RMSE**	**MSE**	**MAE**	**RMSE**	**MSE**	**MAE**	**RMSE**
1	0.409	0.513	0.639	0.741	0.733	0.861	2.351	1.447	1.533
2	0.826	0.789	0.909	**0.315**	**0.415**	**0.561**	1.522	0.999	1.234
3	1.276	1.026	1.130	2.224	1.386	1.491	**0.846**	**0.714**	**0.920**
4	1.258	1.017	1.121	1.796	1.230	1.340	2.043	1.273	1.429

5	**0.262**	0.401	**0.512**	0.330	0.450	0.575	189.700	11.551	13.773
6	0.262	**0.400**	0.512	0.424	0.510	0.651	187.792	11.472	13.704
7	0.664	0.683	0.815	186.494	11.417	13.656	185.801	11.388	13.631
8	0.451	0.547	0.671	185.143	11.361	13.607	185.005	11.355	13.602
9	0.977	0.875	0.989	185.584	11.379	13.623	184.715	11.343	13.591

Table 7: The performance evaluation for Case 3

Less data with old hidden nodes									
No. of hidden layer	**ReLU**			**Sigmoid**			**Tanh**		
	MSE	**MAE**	**RMSE**	**MSE**	**MAE**	**RMSE**	**MSE**	**MAE**	**RMSE**
1	0.420	0.530	0.648	0.812	0.758	0.901	**0.637**	**0.675**	**0.798**
2	0.380	0.485	0.616	**0.349**	**0.459**	**0.590**	161.115	10.574	12.693
3	**0.267**	**0.403**	**0.517**	3.671	1.717	1.916	156.389	10.390	12.506
4	0.737	0.726	0.859	156.542	10.396	12.512	158.165	10.459	12.576
5	1.287	1.018	1.135	154.954	10.334	12.448	156.452	10.393	12.508
6	0.718	0.714	0.848	156.751	10.404	12.520	157.354	10.428	12.544
7	0.612	0.662	0.783	156.388	10.390	12.506	156.810	10.407	12.522
8	0.272	0.405	0.522	156.627	10.399	12.515	157.490	10.433	12.550
9	0.357	0.468	0.598	156.715	10.403	12.519	156.456	10.393	12.508

Table 8: The performance evaluation for Case 4

Less data with 3:2 ratio hidden node									
No. of hidden layer	**ReLU**			**Sigmoid**			**Tanh**		
	MSE	**MAE**	**RMSE**	**MSE**	**MAE**	**RMSE**	**MSE**	**MAE**	**RMSE**
1	0.446	0.540	0.668	3.744	1.796	1.935	**1.130**	**0.942**	**1.063**
2	0.375	0.484	0.612	**0.570**	**0.604**	**0.755**	157.818	10.446	12.563
3	0.283	0.416	0.532	156.518	10.395	12.511	159.008	10.492	12.610
4	0.708	0.719	0.841	157.683	10.441	12.557	157.176	10.421	12.537
5	**0.261**	**0.397**	**0.511**	157.298	10.426	12.542	157.360	10.428	12.544
6	0.264	0.398	0.514	156.690	10.402	12.518	156.273	10.386	12.501
7	1.303	1.035	1.141	156.575	10.397	12.513	157.195	10.422	12.538
8	0.299	0.421	0.547	72.276	7.249	8.502	156.686	10.402	12.517
9	1.378	1.068	1.174	452.250	19.865	21.266	82.221	7.419	9.068

Size of Data In Testing Set

By comparing Tables 5 and 6, it can be observed that among the three activation functions, ReLU performs better overall, with the lowest error measures. On the other hand, the error measures are bigger for the architecture with sigmoid and tanh, even if it is compared within the cases. Besides that, it can be observed that the values of error measures dropped drastically in Table 8. This is because when the size of data in the testing set is reduced to half in Case 2, there was no deviation between the forecasted price and real price.

Therefore, reducing the data in the testing set can improve the performance of the architecture with sigmoid and tanh activation functions as well. However, ReLU still outperforms in both cases with the lowest error measures. In Case 1, the MSE is 0.2606, MAE is 0.3931 and RMSE is 0.5105, and in Case 2, the MSE is 0.2619, MAE is 0.3995 and RMSE is 0.5117. It turns out that reduction of data in the testing set significantly improves the architecture with sigmoid and tanh activation functions, but the performance of the architecture with ReLU drops slightly as it can be observed by comparing the values of error measures in Tables 8 and 9.

Size of Data

In Case 4, ReLU still outperforms the other two activation functions. From Table 8, it can be observed that ReLU performs best in this case with five hidden layers, where the error measures result in the lowest value (MSE:0.2614; RMSE:0.5113). From this, it can be concluded that the more the data, the better the performance of the developed architecture.

Number of Hidden Nodes in Each Layer

In Case 3, the number of first hidden node is not determined based on the number of input data. In this case, the number on hidden nodes used is the same as in Cases 1 and 2. Whereas in Case 4, the determination of the number of nodes is done based on the number of input data. Refer to Tables 3 and 4 for the number of hidden nodes in each hidden layer in Cases 3 and 4, respectively.

It can be observed from Table 7 that the architecture with ReLU still performs better, even when the number of hidden nodes is comparatively smaller in each hidden layer. The architecture with ReLU performs better in Case 3, with three hidden layers, where the values of error measure are 0.2671, 0.4027 and 0.5168 for MSE, MAE and RMSE, respectively. It can be clarified that the architecture is suitable when the number of hidden nodes has a ratio of 3:2, where the first number of hidden nodes is determined based on the number of inputs. As it observed in Table 5, the value of error measures of the five hidden layers in Case 4 is smaller compared with the lowest value of error measures of Case 3.

From the comparison between the cases, it can be concluded that the MLP architecture performs better with the ReLU activation function. Besides that, we can say that the forecast using tanh and sigmoid activation functions can be improved by increasing the number of epochs during training.

Conclusion

Deep learning has been applied in a diverse range of fields, and researches are being done to improve the architecture, which has been applied in almost all fields. One of the ways improve it is by conducting a study on specific factors that alter the performance of the architecture. In this study, the first objective is achieved by building a MLP architecture for time series forecasting using Keras, the deep learning library in the Python software. The developed architecture is applied on the forecasting of the AAPL stock price. The architecture is then analysed and evaluated on different factors to achieve the second objective. The factors that are used for the evaluation are the size of data in testing set, size of data and the number of hidden nodes in each layer.

It is proven that when the data in the testing set is smaller, the performance of the architecture with sigmoid and tanh activation functions is better. This is not the case for the ReLU activation function. The architecture with ReLU performs better when the data in the testing set is bigger. When it is compared with the size of data in the testing set, the ideal architecture for forecasting would be an architecture with ReLU activation function with more data in the testing set. Apart from that, when the size of data loaded into the architecture is smaller, the performance of the architecture with sigmoid and tanh is better. In contrast, the architecture with ReLU performs better when the size of data used is bigger.

Lastly, the performance of the architecture with sigmoid and tanh is better when the first number of the hidden nodes are not determined by rule of thumb. But, still, the performance of the architecture with the ReLU activation function is the best when rule of thumb is used.

By all counts and with proven results from each case, it can be said that ReLU is the best activation function for forecasting time series data. Hence, the ideal architecture for time series data with similar characteristic to the AAPL stock price would be an architecture with more data, four hidden layers and the ReLU activation function. Moreover, it is observed that the performance of the architecture differs in each case for different numbers of hidden layers. Hence, it can be said that the optimal number of hidden layers differs in every circumstances.

Acknowledgements

Authors wish to thank the university for the research opportunity and the reviewers for the constructive comments.

References

Amardeep, R., & Swamy, K.Y. (2017). Training feed forward neural network with backpropogation. *International Journal of Engineering and Computer Science, 6*(1), 19860-19866.

Apple Inc. (AAPL). 2019. *Yahoo Finance.* Retrieved from https://finance.yahoo.com/quote/A APL/history?period1=1356969600&period2=1514736000&interval=1d&filter=history&frequency=1d.

Hiransha, M., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. (2018). NSE Stock Market prediction using deep-learning models. *Procedia Computer Science, 132,* 1351-1362.

Karsoliya, S. (2012). Approximating number of hidden layer neurons in multiple hidden layer BPNN architecture. *International Journal of Engineering Trends and Technology 3*(6).

Liu, W., Wand, Z. Liu, X. Zeng, N. Liu, Y. & Alsaadi, F. E. (2017). A survey of deep neural network architectures. *Neurocomputing,* (234),11–26.

Naeini, M. P, Taremian, H., & Hashemi, H.B. (2010). Stock market value prediction using neural networks. *International Conference on Computer Informatics System and Industrial Management Application (CISIM*), 132-136.

Patterson, J., & Gibson, A. (2017). Deep learning. Retrieved from https://www.oreilly.com/library/view/deep-learning/9781491924570/ch04.html.

Shah, T. (2017). About train, validation and test sets in machine learning. Retrieved from https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7.

Sharma, S. (2017). Epoch vs batch size vs iterations. Retrieved from https://towardsdatascienc e.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9.

Upadhyay, Y. (2019). Introduction to FeedForward Neural Networks. Retrieved from https:// towardsdatascience.com/feed-forward-neural-networks-c503faa46620.

Ved. (2019). How to improve performance of neural network. Retrieved from https://d4datasci ence.wordpress.com/2016/09/29/fbf/.

Widegren, P. (2017). Deep learning-based forecasting of financial assets. dissertation degree. KTH Royal Institution Technology.

Xu, Y. P., Lai, J., & L. (2017). Deep learning based regression and multiclass models for acute oral toxicity prediction with automatic chemical feature extraction. Retrieved from https://arxiv.org/pdf/1704.04718.pdf.