Sakarya University Journal of Computer and Information Sciences
http://saucis.sakarya.edu.tr/

DOI: 10.35377/saucis...1172027 RESEARCH ARTICLE

LSTM Hyperparameters optimization with Hparam parameters for Bitcoin Price Prediction

I. Sibel Kervancı [1], M. Fatih Akay [2]

[1]Gaziantep University; IT Department Gaziantep/Türkiye
[2] Çukurova University, Faculty of Engineering, Department of Computer Engineering Adana/Türkiye

Corresponding author:
I.Sibel Kervancı , Gaziantep University; IT Department Gaziantep/Türkiye
E-mail address:
skervanci@gantep.edu.tr

Received: 7 September 2022
Revised: 27 November 2022
Accepted: 02 January 2023
Published Online: 22 March 2023

Citation: Kervancı S. and Akay MF. (2023). LSTM Hyperparameters optimization with Hparam parameters for Bitcoin Price Prediction. *Sakarya University Journal of Computer and Information Sciences*. 6 (1). https://doi.org/10.35377/saucis...1172027

ABSTRACT
Machine learning and deep learning algorithms produce very different results with different examples of their hyperparameters. Algorithm parameters require optimization because they are not specific to all problems. This paper used Long Short-Term Memory (LSTM) and eight different hyperparameters (go-backward, epoch, batch size, dropout, activation function, optimizer, learning rate, and the number of layers) to examine daily and hourly Bitcoin datasets. The effects of each parameter on the daily dataset on the results were evaluated and explained. These parameters were examined with the hparam properties of Tensorboard. As a result, it was seen that examining all combinations of parameters with hparam produced the best test Mean Square Error (MSE) values with hourly dataset 0.000043633 and daily dataset 0.00061806. Both datasets produced better results with the tanh activation function. Finally, when the results are interpreted, the daily dataset produces better results with a small learning rate and dropout values. In contrast, the hourly dataset produces better results with a large learning rate and dropout values.
Keywords: LSTM, optimization, Bitcoin, prediction, hparam

1. Introduction

For a prediction problem optimization method, dataset's properties, and learning algorithm's parameters affect the prediction accuracy and learning process. However, there is no theoretical approach to how they should be selected. Instead, parameters are determined by experimental approaches using optimization methods [1]. This paper seeks the most appropriate dataset and hyperparameters to get the best results for Bitcoin price prediction with LSTM. For this reason, this study consists of Bitcoin prices at different time intervals and examines datasets with LSTM. Also, this study detects dominant hyperparameters values and interactions by using important hyperparameters and speeding up the optimization process. An LSTM is a particular type of recurrent neural network (RNN) with long-term memory. There are different types of LSTMs that are not required to be present in the three doors mentioned. But in general, some gates hold or forget that allow data to be transferred. The number of doors can vary in different LSTM examples. For LSTM, there are layers in Keras, the number of hidden neurons, and parameters for each gate. Besides, extra dropout layer parameters can be optimized to avoid exploding gradient problems. The training of LSTM networks in solving several problems, as in neural networks, depends on several hyperparameters that determine several appearances of algorithm reactions [2]. Basically, hyperparameter optimization techniques are manual and automatic.

Manual approaches: Different combinations of values need to be tried repeatedly to obtain optimal values for each of these hyperparameters. For this, a specialist is required.

Automatic approaches: There are several hyperparameter optimization algorithms. Grid Search, Random Search, Particle swarm optimization, simulated annealing, and automated hyperparameter optimization methods. Automatic approaches are challenging to apply because of their high computational cost [2]. But now, computer clusters or graphics processor unit (GPU) processors enable more experimentation to use to cheaper than last.

Based on the reference [3], it is stated that the result of the manual search can be reliably matched with Random Search for some datasets. As proposed in the same reference Gaussian Process (GP) and Tree Structured Parzen Estimator (TPE) hyperparameter optimization algorithms are close to or surpass the performance of manual and Brute-Force Random Search algorithms. In this paper, the parameters go-backward, epoch, batch size, dropout, activation function, optimizer, learning rate and the number of layers were manually evaluated to understand the effect of hyperparameters on learning. The results of all combinations of parameters and their values are obtained and visualized by using the hparam parameter feature of the Tensorboard. The manual approach of the hyperparameters shows the success that cannot be ignored. The relatively small time series Bitcoin price was used as the manual approach and the automated approach will be used in our future study.

2. Literature review

This paper is divided into two parts: the effect of the dataset and hyperparameters on the learning.

2.1 The effect of the datasets

According to reference [4], bitcoin is primarily based on historical datasets, and seasonality can be weekly, daily or hourly. They used two model time-length 30, 60, 120 minutes and 180, 360, 720 minutes. As a result, there were raising the threshold and the mean holding time, the number of trades reduced while the average profit per trade rose. At the end of 50 days, it almost doubled the investment. Based on the reference [5] , predictions were made using the 1-minute and 30-minute datasets. In the 1-minute datasets, the price fluctuation between two different markets (CNY and USD) is quite high. But in the 30-minute dataset, the fluctuation has improved. Based on the reference [6], review article, different articles were evaluated according to the frequency of the time intervals, which are daily, hourly and different minute intervals. The minute and second datasets are used for instant trading, while the daily dataset is used for further dates. Based on the reference [7], used bitcoin price and S&P500 datasets with daily, 1 hourly, and 15-minute time intervals. Performance was compared with LSTM and selected hyperparameters for datasets in different time intervals. Since dataset from different periods yielded similar results, they combined them into a single ensemble model. As a result, both hourly and daily prices in this paper were used in univariate LSTM.

2.2. The effect of hyperparameters

Based on the reference [7], they used a number of layers, neurons in each layer, dropout rate, learning rate, l2 kernel regularization, optimizer, and momentum. In place of testing each hyperparameter singly, they used Hyperband, which provided an understanding of the effect of changes made in several parameters simultaneously on network performance. They found the best optimizer is 'Adam', and the best activation is 'tanh', which aligns with our results. In contrast, others are different because of parameter intervals. Based on the reference [8], most performance variations depend only on a few hyperparameters, even in very high-dimensional situations related to the relationship between hyperparameter settings and performance. Based on the reference [9], they investigated a subset of possible hyperparameters with Grid Search. Improperly selected learning rate or dropout can make it difficult for the model to learn effectively. When using hparam or Grid Search to overcome this, parameters will be scanned in all possible ranges, resulting in more efficient learning.

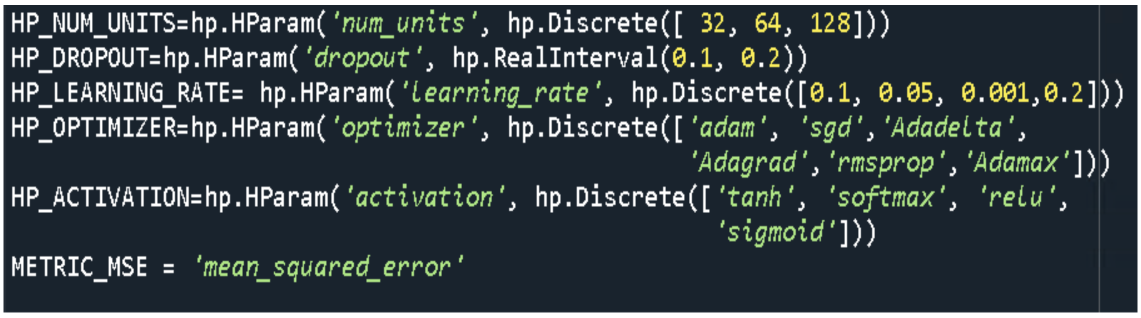
3. Methods

This section will describe the methods; Python (version 3.9.7) in Spyder (version 5.1.5), keras library (version 2.7.0), Tensorflow (version 2.7.0) backend and Tensorboard (version 2.7.0). Keras has two main models; one is sequential (), and the other is the model class used with the functional API. This paper was used as a model, sequential (). Keras has two main models; one of them is sequential (), and the other is the model class used with the functional API. This paper was used as a model, sequential (). Downloaded the daily dataset from https://www.investing.com/ and the hourly dataset from https://www.cryptodatadownload.com.Two datasets were used for the daily datasets, consisting of 2700 lines from 01 January 2015 to 22 May 2022, 1469 lines from 15 May 2018 to 22 May 2022. The hourly dataset consists of 36957 records from 06:00 15 May 2018 to 23:00 22 May 2022. While comparing the daily and hourly datasets in section 4.9, it aims to ensure fairness by using the date range of 15 May 2018 to 22 May 2022.

MSE was used to evaluate the results and its formulation is as follows.

$$\mathrm{MSE}=\frac{\sum_{j=1}^{m}(\mathrm{yj}-\mathrm{xj})2}{m} \qquad (1)$$

The dropout technique prevents overfitting and leaves unrelated information from the network to improve performance [10]. Education of deep neural networks is complicated as the parameters of the previous layer and the distribution of the inputs of each layer change during training[11]. For this reason, the effect of the parameters (go-backward, activation function, optimizer, batch size, learning rate, and epochs) with a single layer LSTM has been observed in this paper. For all experiments, if the dropout is not specified, dropout=0.2 was taken to prevent overfitting. At the end of each experiment, the kernel reset so that no results affect each other. In experiments using the hparam feature of Tensorboard, the parameters and interactions that yield better results are more clearly seen as the measurements are logged and visualized. Tested the hparam parameters of the tensorboard by giving the values in Figure 1.



```
HP_NUM_UNITS=hp.HParam('num_units', hp.Discrete([ 32, 64, 128]))
HP_DROPOUT=hp.HParam('dropout', hp.RealInterval(0.1, 0.2))
HP_LEARNING_RATE= hp.HParam('learning_rate', hp.Discrete([0.1, 0.05, 0.001,0.2]))
HP_OPTIMIZER=hp.HParam('optimizer', hp.Discrete(['adam', 'sgd','Adadelta',
                                                'Adagrad','rmsprop','Adamax']))
HP_ACTIVATION=hp.HParam('activation', hp.Discrete(['tanh', 'softmax', 'relu',
                                                  'sigmoid']))
METRIC_MSE = 'mean_squared_error'
```

Figure 1 Hparam parameters

While discrete in Figure 1 shows that the interval is discrete, Realinterval represents the number interval. Each result of go-backward, activation function, optimizer, batch size, learning rate, and epochs are evaluated separately below.

4. Results

The results for each parameter are evaluated separately below.

4.1. “go-backward” parameter

In our tests with go-backwards parameter, it was observed that the go-backwards='False' value gave better results with the increase of the epoch. Recommended go-backwards should be 'False' for time series prediction. If the go-backwards parameter is suitable for your dataset for instance if natural language processing (NLP) is being studied, go-backward=True should be used, which allows both forward and backward contexts to be used. As shown in Table 1, when 'True' value is compared to 'False', train MSE gives better results, while test MSE gives worse results. Based on Table 1, we conclude that there may be excessive learning in the 'True' value train process.

Table 1 The effects of the go-backward on the RMSE and MSE.

Epoch	Go-backward	Activation function	Optimizer	Train MSE	Test MSE
10	False	tanh	Adam	0.000346	0.013109
10	True	tanh	Adam	0.000354	0.013603
100	False	tanh	Adam	0.000020	0.000672
100	True	tanh	Adam	0.000019	0.000694

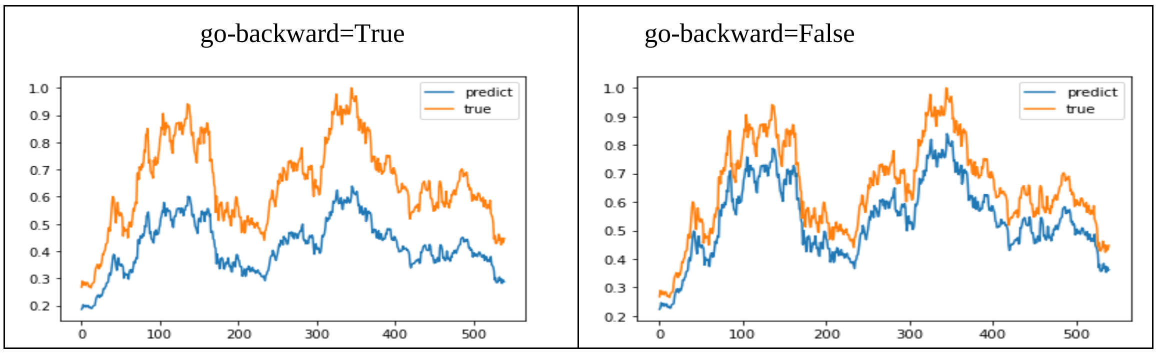


Figure 2 True and predicted value of Bitcoin price

As shown in Figure 1, the difference between the actual and estimated values from the test dataset is more reasonable with go-backward=False.

4.2. Activation function

Based on the reference [12], 23 different activation functions evaluated. But in this paper tanh, relu, sigmoid, and softmax were examined.

Table 2 The effects of the optimizer and the activation function on MSE (epoch=100).

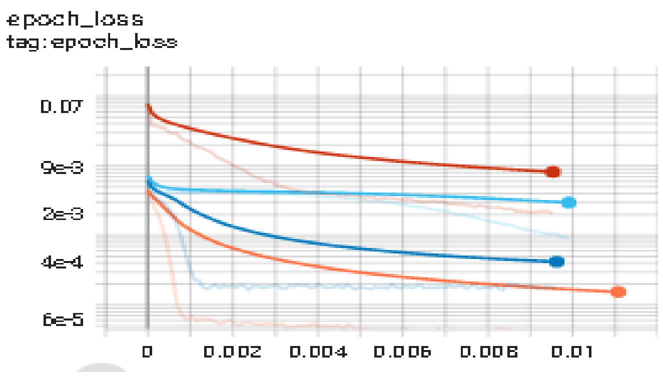
Batch size	Activation function	Optimizer	Train MSE	Test MSE
64	tanh	Adam	0.000019	**0.000667**
64	relu	Adam	0.000033	0.002130
64	sigmoid	Adam	0.000573	0.080025
64	softmax	Adam	0.000460	0.101000
64	tanh	RMSprop	0.000024	**0.000858**
64	relu	RMSprop	0.000023	0.000969
64	sigmoid	RMSprop	0.000279	0.053592
64	softmax	RMSprop	0.000481	0.095665
64	tanh	Adamax	0.000021	**0.001542**
64	relu	Adamax	0.000027	0.003094
64	sigmoid	Adamax	0.002187	0.191153
64	softmax	Adamax	0.003035	0.270137
64	tanh	SGD	0.003852	**0.328543**
64	relu	SGD	0.003741	0.342115
64	sigmoid	SGD	0.004248	0.363644
64	softmax	SGD	0.004231	0.360037
64	tanh	Adagrad	0.004515	0.384147
64	relu	Adagrad	0.003808	**0.331651**
64	sigmoid	Adagrad	0.004130	0.352627
64	softmax	Adagrad	0.004264	0.367113
64	tanh	Adadelta	0.006401	0.420758
64	relu	Adadelta	0.006896	0.417276
64	sigmoid	Adadelta	0.004808	**0.346178**
64	softmax	Adadelta	0.009292	0.446577

As shown in Table 2; root mean square propagation (RMSprop), Adaptive moment estimation (Adam), Stochastic gradient descent (SGD), adaptive moment estimation with maximum (Adamax), adaptative gradient (Adagrad) and adaptive learning rate method (Adadelta) are selected as optimizers. Optimizers (Adam, SGD, RMSprop, and Adamax) achieved the best results with tanh activation function. Adadelta with sigmoid and Adagrad with relu pairs obtained better MSE values than other activation functions. The results with Adagrad and Adadelta optimizers are quite poor. Sigmoid and softmax activation functions generally produce worse results in trials. Sigmoid and softmax activation functions were tried with 1000 epochs as they may require more epochs. Sigmoid with Adam, and epoch=1000 result Train MSE: 0.000026, Test MSE: 0.040009. Softmax with Adam, and epoch=1000 results Train MSE: 0.000023, Test MSE: 0.061527. Even with sigmoid and softmax epoch=1000, it could not approach the Test MSE of the tanh activation function.

4.3. Learning rate

Based on the reference [13], SGDs require manual adjustment of optimization parameters such as learning rates. If a person does not understand the task at hand, it is very difficult to find a good learning rate. In this case, running the learning algorithm with many optimization parameters and choosing the best performing model in a validation set should be selected.

The learning rate was applied to the same LSTM structure in Table 1, 2.



In order to see the results better, the relative property of the horizontal axis epoch value has been applied. Optimizer=Adam; activation functions tanh=orange , relu=darkblue, softmax= blue, sigmoid=brown.

Figure 3 Graph of loss values of activation functions for Adam optimization.

Table 3 The effect of the learning rate on test MSE.

Optimizer	Activation function	Learning rate(Lr)	Test MSE
rmsprop	tanh	0.001	0.00070297
adam	tanh	0.2	0.00070604
adam	tanh	0.001	0.00070997
rmsprop	relu	0.001	0.00078362
Adamax	tanh	0.2	0.00094503
Adagrad	tanh	0.2	0.00101260
Adamax	tanh	0.001	0.00105760
Adamax	tanh	0.1	0.00144720
rmsprop	tanh	0.001	0.00070297

When we examine the results in Table 3, the lowest MSE values are obtained with a 0.001 learning rate. When Table 2 and Table 3 are compared, it is observed that the test MSE value improves with the use of the learning rate parameter except for adam-tanh.

As seen in the Table 2; Adam method enables the adam-tanh pair to obtain the best test MSE 0.000667 without using the Lr. Due to the square root effect, Lr effect gradually diminishes and changes slowly around the globally minimum. In this way, the globally minimum is found faster and it is not requiring more Lr optimization with Adam.

4.4. Optimizer

The selection of the optimizer is important to training. SGD, Ftrl, Adam, Adadelta, RMSprop, Adagrad, Nadam, and Adamax are optimizers. SGD and its variants such as Adam [14], Adagrad [15] and RMSprop [16], are amongst the most popular training methods. Adam was found to be widely applicable despite of requiring less regulation of its hyperparameters [17].

Based on the reference [9], they concluded that Adadelta, Adagrad, and SGD required more epochs, our result supports this conclusion. SGD epoch=1000 Train MSE: 0.000020, test MSE: 0.000859 as a result, concluded that SGD want higher epochs to produce approximate results for other optimizers like Adam, RMSprop. Adam was found to be widely applicable despite requiring less regulation of its hyperparameters. The results in Table 3 were created by taking the best 11 results of that make up the tensorboard hparam parallel coordinate view in Figure 4.

Based on the reference [18], SGD and SGD variants such as momentum have proved to be an effective way of training deep networks. Our results are supported by references [18] , [9]; tanh, relu activation functions produced the best results. In general, Rmsprop and Adam showed the best performances.

4.5. Batch size

Batch is a set of dataset samples. The dataset group in each batch size independently of each other processed in parallel. If we increase the batch size, our approach will be the better, but the process takes a long time and will still result in only one backward weight update. The number of times the backward parameters are updated is determined by the number of an epoch. It is advised to pick a batch size that is as large as you can afford without going out of memory [19]. Based on the reference [20], they said that the training dataset size should be divisible by batch size to ensure safe termination because they use cyclical learning rates. However, cyclical learning did not apply in this study, and batch size fixed 64 in Tables 1, 2, and 3.

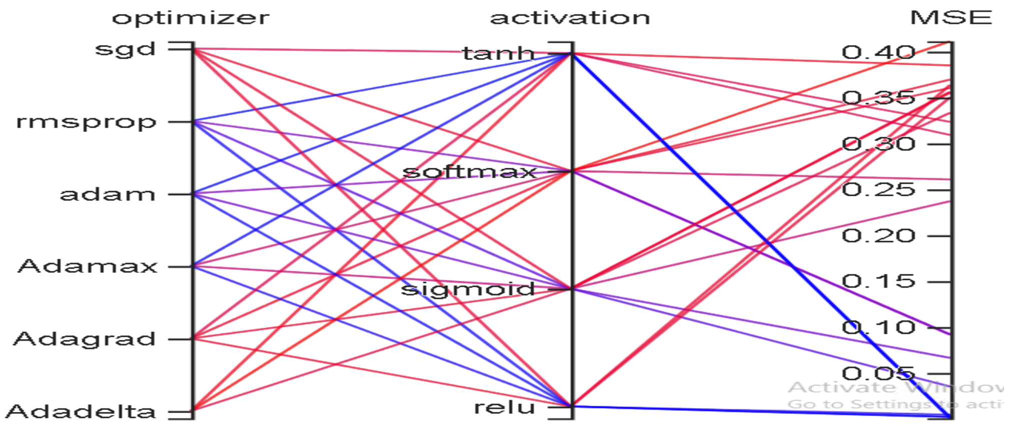


Figure 4 Optimizer and activation function parallel coordinates

Based on reference [21] the impact of batch size on performance, the higher the batch size, the better train and test MSE value like our result in Table 4 (daily dataset). However, it is not a valid generalization for all hyperparameter combinations and datasets, as seen in Tables 7 and 8.

Table 4 The effect of the batch size with daily dataset on test MSE values.

Batch size	Test MSE
32	0.00075906
64	0.00074598
128	0.00066059

4.6. Number of layers

If multi-layer LSTMs are used, usually the first LSTM layer is allowed to output all output (not just one output). This output is fed as input to the next LSTM layer. Since the output of the first layer will be used as input for the next layer, the “return_sequences” parameter of the LSTM layer must be "true".

Table 5 The effect of the number of layers on the train and test MSE values.

Dropout after each layer	Activation function	optimizer	Number of layers	LSTM unit in each layer	Epoch	Train MSE	Test MSE
0.2	tanh	Adam	1	64	100	0.000040	0.005751
0.2	tanh	Adam	2	64	100	0.000021	0.001992
0.2	tanh	Adam	3	64	100	0.000022	0.007281
0.2	tanh	Adam	4	64	100	0.000038	0.015507

Although better train and test MSE values are obtained with increasing layers, the values get worse after a certain number of layers as it continues to increase. As shown in Table 5, while the 2-layer structure produced the best MSE result, the values deteriorated with more than 2 layers.

4.7. Epoch

Epoch shows how many times a dataset is given to the network to train the network. In deep learning algorithms, we need to update the weights and thus pass the whole dataset multiple times to obtain a better and more accurate prediction model to optimize gradient descent. However, it is not clear how many epochs are needed to train a model with the same dataset to achieve optimum weights. For the best train to the network, different datasets proceed differently therefore epoch numbers are different [22]. Based on the reference [9], they measured the number of epochs for an optimizer, convergence observed huge variations in terms of the number of epochs until Nadam converged the fastest, and only requires few training epochs to achieve good performance. SGD, Adadelta, and Adagrad require more epoch. As seen from Table 1 and Figure 2 the increase in the number of an epoch improved the MSE values. However, after a certain increase, the number of epochs does not contribute to improving the performance, which varies according to the training you have a dataset. As seen in Figure 3, as the epoch increases, the loss decreases, and after a certain epoch value, it doesn't effect on the loss.

4.8. Dropout

Dropout is implemented on any or all hidden layers in the network except the output layer. The dropout technique removes irrelevant information from the network to prevent overfitting and improve performance [10]. Based on the reference [23], they offered dropout and defined how specific hyperparameters should be.
Based on the reference [24], the accuracy of CNN dropout was clearly better than without dropout CNN. It is increased by 58.15% accuracy on a small images' dataset, but our dataset is time series.
According to our experiments, if the dropout value is used, the difference between the train and test MSE values will decrease.

Table 6 The effect of the dropout on the train and test MSE values.

Batch size	Optimizer	Activation function	Dropout	Dataset	Train MSE	Test MSE
64	Adamax	tanh	-	hourly	0.000079	0.000042
64	Adamax	tanh	0.1	hourly	0.000080	0.000045
64	Adamax	tanh	-	daily	0.000020	0.001268
64	Adamax	tanh	0.1	daily	0.000026	0.000969

4.9. The effect of the daily and hourly datasets

The same code with the daily and hourly datasets was executed for the same date range of 15 May 2018 to May 2022. These tests were carried out to see the contribution of daily and hourly datasets to Bitcoin price prediction. Based on the reference [25], g is a noise scale and has an effect on a training and test accuracy.

$$g = Lr(\frac{train_{dataset\ size}}{batch_{size}} - 1) \quad (2)$$

Table 7 The results of the hourly dataset with hparam parameters (epoch=100).

Batch size	Optimizer	Activation function	Dropout	Learning rate(Lr)	Test MSE
64	Adamax	tanh	0.2	0.001	**0.000043633**
64	Adadelta	tanh	0.1	0.05	0.000045415
64	Adadelta	tanh	0.1	0.1	0.000047583
64	Adadelta	tanh	0.2	0.05	0.000047906
32	Adagrad	tanh	0.1	0.1	0.000048397
64	Adamax	tanh	0.1	0.001	0.000048572
32	Adamax	tanh	0.1	0.001	0.000050041
64	Adagrad	tanh	0.2	0.1	0.000051697
64	Adadelta	tanh	0.1	0.2	0.000052008
32	Adagrad	tanh	0.2	0.2	0.000052436

Table 7 shows the best 10 values and parameters out of 576 combinations (Figure 1) for the hourly dataset.

Table 8 The results of the daily dataset with hparam parameters (epoch=100).

Batch size	Optimizer	Activation function	Dropout	Learning rate(Lr)	Test MSE
64.000	tanh	0.1	Adamax	0.001	**0.00061806**
128.00	relu	0.1	adam	0.001	0.00061871
128.00	tanh	0.1	Adagrad	0.100	0.00062113
128.00	tanh	0.2	adam	0.001	0.00062767
128.00	tanh	0.2	Adagrad	0.100	0.00062931
64.000	tanh	0.1	Adadelta	0.200	0.00063148
64.000	tanh	0.2	Adadelta	0.200	0.00063465
128.00	relu	0.2	Adagrad	0.100	0.00063694
32.000	relu	0.1	Adamax	0.001	0.00063716

32.000	softmax	0.1	adam	0.001	0.00063835

Table 8 shows the best 10 values and parameters out of 576 combinations (Figure 1) for the daily dataset.

The daily dataset produces better results with a small learning rate and small dropout values, whereas the hourly dataset produces better results with a large learning rate and large dropout values.

5. Conclusions and Recommendations

The contribution of this paper, the Bitcoin price prediction problem is to infer which of the hyperparameters work better with each other.

We observed that the test MSE values of the hourly dataset gave better results than the daily dataset.

For numerical time series, it is recommended to set the go-backward parameter to False.

Tanh and relu activation functions are suitable for these datasets.

It is quite clear that the SGD optimizer gives better results with the learning rate parameter, and we recommend they to be used together.

In general, the impact of batch size on performance, the higher the batch size, the better the test MSE value.

When all other parameters were fixed, it was seen that tanh and relu activation functions with SGD, Adam, RMSprop and Adamax optimizers, and sigmoid activation function with Adadelta optimizer gave better results.

Achieved the two-layer LSTM with the parameters we optimized, better than the 3 and 4-layer LSTM, which shows us the importance of hyperparameter optimization.

As seen in Tables 7 and 8 the result of “g”, better MSE values were obtained when Lr decreased, and batch size was constant, or Lr was constant and batch size increased, or dropout increases, and the other parameters fixed.

When the test MSE values of the daily and hourly datasets are examined, the hourly dataset is recommended since the hourly dataset produces better test MSE values.

References

[1] D. Choi, J. C. Shallue, Z. Nado, J. Lee, J. C. Maddison and E. G. Dahl, ”On Empirical Comparisons of Optimizers for Deep Learning,*” Computing Research Repository (CoRR),* vol. abs/1910.05446, Oct 2019.

[2] B. Nakisa, M. N. Rastgoo, A. Rakotonirainy, F. Maire and V. Chandra, ”Long Short Term Memory Hyperparameter Optimization for a Neural Network Based Emotion Recognition Framework,” *IEEE Access,* vol. 6, pp. 49325 - 49338, Sept 2018.

[3] J. Bergstra, R. Bardenet, Y. Bengio and B. Kegl, ”Algorithms for Hyper-Parameter Optimization,” *Advances in Neural Information Processing Systems 24 (NIPS 2011),* 2011.

[4] R. K. Rathore, D. Mishra, P. S. Mehra, O. pal, A. S. Hashim, A. Shapi'i, T. Ciano and M. Shutaywi, ”Real-world model for bitcoin price prediction,”*Information Processing and Management,* vol. 59, no. 4, 2022.

[5] T. Shintate and L. Pichl, ”Trend Prediction Classification for High Frequency Bitcoin Time Series with Deep Learning, ” *Journal of Risk Financial Management,* vol 12, no. 1, 2019.

[6] I. S. Kervancı and M. F. Akay, ”Review on Bitcoin Price Prediction Using Machine Learning and Statistical Methods,” *Sakarya University Journal of Computer and Information Sciences,* vol. 3, no. 3, pp. 272-282, 2020.

[7] J. Michańków, P. Sakowski and R. Ślepaczuk, ”LSTM in Algorithmic Investment Strategies on BTC and S&P500 Index,” *Sensors* , vol. 22, no. 3, 2022.

[8] F. Hutter, H. Hoos and K. Leyton-Brown, ”An Efficient Approach for Assessing Hyperparameter Importance,” *Proceedings of the 31st International Conference on Machine Learning*, 2014.

[9] N. Reimers and I. Gurevych, ”Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks,” *EMNLP 2017*, 2017.

[10] A. U. Rehman, A. K. Malik, B. Raza and W. Ali , ”A Hybrid CNN-LSTM Model for Improving Accuracy of Movie Reviews Sentiment Analysis,” *Multimedia Tools and Applications,* no. 78, pp. 26597–26613, June 2019.

[11] S. Ioffe and C. Szegedy, ”Batch Normalization: Accelerating Deep Network Training Reducing Internal Covariate Shift,” *Proceedings of the 32 nd International Conference on Machine Learning*, Lille, France, 2015.

[12] A. Farzad, H. Mashayekhi and H. Hassanpour , ”A comparative performance analysis of different activation functions in LSTM networks for classification,” *Neural Computing and Applications volume*, vol. 31, pp. 2507–2521, 2019.

[13] Q. V. Le , J. Ngiam, A. Coates, A. Lahiri, B. Prochnow and A. Y. Ng, ”On Optimization Methods for Deep Learning,” *ICML,* 2011.

[14] D. P. Kingma and J. Ba, ”Adam: A Method for Stochastic Optimization,” *Published as a conference paper at the 3rd International Conference for Learning Representations,* San Diego, 2015.

[15] J. Duchi , E. Hazan and Y. Singer, ”Adaptive Subgradient Methods for Online Learning and Stochastic Optimization,” *Journal of Machine Learning Research,* vol. 12, pp. 2121-2159, 2011.

[16] G. Hinton, N. Srivastava and K. Swersky, ”Lecture 6.5 Rmsprop- Divide the Gradient by a Running Average of its Recent Magnitude,” *Toronto Uni,* 2012.

[17] S. Merity, N. S. Keskar and R. Socher, ”Regularizing and Optimizing LSTM Language Models,” *ICLR 2018,* Vancouver, BC, Canada, 2018.

[18] I. Sutskever, J. Martens and G. D. Geoffr, ”On the Importance of Initialization and Momentum in Deep Learning,” *Proceedings of the 30th International Conference on Machine Learning, PMLR,* 2013.

[19] keras, ”keras,” Feb. 2020. [Online]. Available: https://keras.io/layers/recurrent/.

[20] Smith, Leslie N.;U.S. Naval Research Laboratory, ”Cyclical Learning Rates for Training Neural Networks,” *2017 IEEE Winter Conference on Applications of Computer Vision (WACV),* Santa Rosa, CA, USA, 2017.

[21] N. Golmant, N. Vemuri, Z. Yao, V. Feinberg, A. Gholami, K. Rothauge, M. W. Mahoney and J. Gonzalez, ”On the Computational Inefficiency of Large Batch Sizes for Stochastic Gradient Descent,” *arXiv preprint arXiv:1811.12941,* Nov 2018.

[22] S. SIAMI NAMIN and A. SIAMI NAMIN, ”Forecasting Economic and Financial Time Series: ARIMA vs. LSTM,” *arXiv preprint arXiv: 1803.06386*, 2018.

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, ”Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *Journal of Machine Learning Research,* vol.15, pp. 1929-1958, 2014.

[24] K. Pasupa ve W. Sunhem, "A Comparison Between Shallow and Deep Architecture Classifiers on Small Dataset,” *8th ICITEE,* Yogyakarta, Indonesia, 2016.

[25] S. L. Smith, P. J. Kindermans, C. Ying and Q. V. Le, ”Don't decay the learning rate, increase the batch size,” *ICLR (International Conference on Learning Representations),* Vancouver, 2018.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Availability of Data and Material

Not applicable.

Ethical Approval and Informed Consent

It is declared that during the preparation process of this study, scientific and ethical principles were followed, and all the studies benefited from are stated in the bibliography.

Plagiarism Statement

This article has been scanned by iThenticate ™.