Indonesian Journal of Artificial Intelligence and Data Mining (IJAIDM)
Vol. 4, No. 1, March 2021, pp. 1 – 10
p-ISSN: 2614-3372 | e-ISSN: 2614-6150

Improving Stock Price Prediction with GAN-Based Data Augmentation

Julisa Bana Abraham
Universitas Gadjah Mada
Tehn Department of Electrical and Information Engineering, Universitas Gadjah Mada
Email: julisa.bana.abraham@mail.ugm.ac.id

Article Info

Article history:
Received Sept 29th, 2020
Revised Oct 18th, 2020
Accepted Nov 02nd, 2020

Keyword:
GAN
Time-series Forecasting
Data augmentation
Deep learning

ABSTRACT

The stock price is one of the most studied time series data because it is deemed to be profitable doing so, however stock price data is still difficult to predict because it is non-linear, non-parametric, non-stationary, and chaotic. One of the methods that most recently used to predict stock price data is deep learning. Although deep learning has a good performance to solve various problems, deep learning must be trained using a lot of data or this method will experience overfitting. This paper proposes a scheme to train a classifier model for predicting stock price time series data using augmented time-series data generated using GAN. The evaluation shows that the classifier model trained using augmented data has better performance on the AMZN and FB stock price datasets.

Copyright © 2021 Puzzle Research Data Technology

Corresponding Author:
Julisa Bana Abraham
Departement of Electrical and Information Engineering,
Universitas Gadjah Mada
Jl. Grafika No. 2 , Kampus UGM, Sinduadi, Mlati, Senolowo, Sinduadi, Kec. Mlati, Kabupaten Sleman, Daerah Istimewa Yogyakarta 55281
Email: julisa.bana.abraham@mai.ugm.ac.id

DOI: http://dx.doi.org/10.24014/ijaidm.v4i1.10740

1. INTRODUCTION

The stock price is one of the most studied time series data because it is deemed to be profitable doing so, however stock price data is still difficult to predict because it is non-linear, non-parametric, non-stationary, and chaotic [1]. One of the methods that most recently used to predict stock price data is deep learning, this method is based on a multi-layered perceptron inspired by the brain's neural network. Deep learning methods have been widely used to solve various problems such as image classification [2], speech recognition [3], sentiment analysis [4], and time series prediction [5,6,7,8].

There have been many studies that use deep learning models to forecast time series data. One deep learning architecture that is widely used for sequence data is a recurrent neural network (RNN). The RNN model has been applied to predict the S&P 500 stock price by using various features such as stock price, moving average, and volume [5]. In this study, it was shown that the performance of this model outperformed other method, the Kalman filter. This method has been applied to 50 other stock prices and produces better predictions than the previous methods.

One of the most popular variants of RNN is LSTM. LSTM's performance in predicting the stock market has been tested by [6]. In this paper the three models, basic RNN, GRU, and LSTM are tested to predict Google's stock price. The results of this study prove that LSTM outperforms the other two models.

Yang et al. [7] designed a multi-layered perceptron to predict the stock market in China, this model was trained using backpropagation and Adam as an optimizer. The results of this study indicate that the model offered is able to have good accuracy. [8] offers a deep learning model with a combination of Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). In this method, LSTM is used as the first layer whose output will be used as input from the next layer which is GRU. This method outperforms the performance of the previous methods.

Although deep learning has a good performance to solve these problems, deep learning must be trained using a lot of data or this method will experience overfitting [9]. To overcome the limitations of data, data augmentation techniques are used to train deep learning algorithms so that they have good performance. Data augmentation techniques are generally used to train image classification models by rotating, shifting and zooming an image. Training results using data augmentation show increased testing accuracy for image recognition [10]. But the same thing cannot be done to time series data, so the most frequent approach taken to do the data augmentation for time series is with Time Slicing and Time Warping [11].

The principle of Time Slicing for time series data augmentation is to divide a complete time-series data into small parts, that part will be input for the classifier [12]. But there is a weakness that is owned by this model that can damage the temporal correlation of time series. While Time Warping is an augmentation technique by speeding up or slowing down portions of the time series of data chosen at random so that the section will appear to be stretched. Parts that are stretched will not lose their distribution features. However, this technique cannot be used for purposes that require certain timescale such as astronomical data, and in some sense stock price data.

To realize a data augmentation method for time series data, [13] designed a data augmentation for time-series using Time-Conditional Generative Adversarial Network (T-CGAN). This method uses deconvolutional NN as a generator and convolutional NN as a discriminator, because its principle is based on CGAN [14] the input for training data is prearranged, in this study, the condition is the data timestamp. The data used in this study are synthetic and real-world data. This method is then compared to the Time Warping and Time Slicing techniques and always outperforms the other two methods. However, this method does not consider using the augmented data to improve the time series forecasting performance.

This paper offers a modified Deep Convolutional GAN (DCGAN) method [15] as a time series data augmentation method whose output along with the original data will be input to train the Deep Learning model for time series forecasting.

2. RESEARCH METHOD

This paper aims to design a time series data augmentation scheme using GAN and improve the performance of the classifier model by training it using the augmented data. In this section, the methods used in this paper will be explained: the GAN architecture, the deep learning model architecture for forecasting, the dataset used, and the evaluation method.

2.1. Proposed GAN model

GAN [16] is a Deep Learning model that is able to study data distribution by training two networks that compete with each other, namely generator and discriminator. The generator is a neural network that is trained to be able to generate fake data (synthetic) which has the same characteristics as the original data, while the discriminator is trained to be able to distinguish the original data and false data from the generator.

The principle of GAN is as follows, the generator must know the distribution of p_g owned by data x by using the mapping function $G(z;\theta_g)$ of a noise distribution $p_z(z)$ while θ_g is the parameter of the model. Whereas the discriminator $D(x;\theta_g)$ is the network whose purpose is to find out the probability of x being the original data and not the result of the generator. So that the two networks play a two-player minimax game with the value function $V(G, D)$ shown in equation (1) [16].

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log \mathrm{D}(x)] + \mathbb{E}_{z\sim P_z(z)}[\log(1-D(G(z)))] \quad (1)$$

In this paper, the GAN model used was inspired by [15], which is a GAN model that uses a deep Convolutional Neural Network (CNN) for the generator and discriminator models. Unlike DCGAN, the GAN model in this paper does not use 2-dimensional CNN but use 1-dimensional CNN instead, because the input of the model is time-series data. In addition, the GAN model in this paper is unconditional GAN, unlike T-CGAN which is a conditional GAN, so that in this paper the input from the generator and discriminator is not conditioned or randomly carried out.

GAN model will be trained as many as 5000 epochs, the number is based on experiments that have been done, the number of epochs that is too much causes the data generated by the generator to be very similar to the original data, this is undesirable because it will be useless if one train the classifier model with very identical data, however in this paper there are no performance metrics that are used to determine the quality of the data generated by the generator as RMSE and MAE is solely based on the error between data.

In this paper, all deep learning modeling uses the Keras framework in the Python programming language. The proposed GAN architecture is shown in Figure 1.

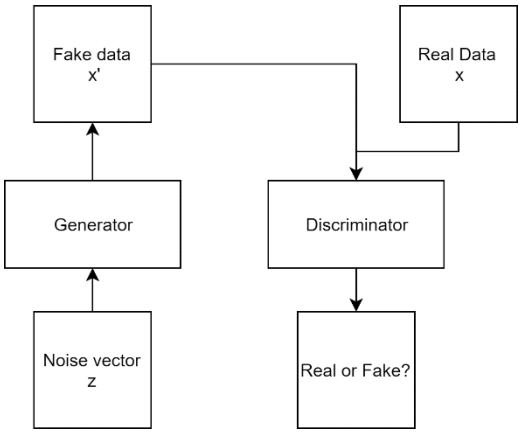

Figure 1. GAN scheme

In this paper, there are several main components for the design of generator and discriminator: batch normalization, rectified linear units (ReLu), Leaky Relu, and 1-dimensional CNN.

a. Batch Normalization (BN)

Training a deep learning model is difficult because the distribution of each input layer always changes during the training process, this causes a slowdown in the training process because it requires a low learning rate. Batch normalization [17] normalizes each input layer, allowing the use of a higher learning rate which makes the training process faster. In this study batch normalization layer is used only for the generator model as suggested by [15].

b. Rectified Linear Unit (ReLU)

ReLu is a non-saturated activation function unlike tanh and sigmoid. The advantage of ReLu compared to tanh and sigmoid is that ReLu can overcome the problem of vanishing gradient and accelerate convergence [18]. ReLu can be represented as equation (2) [16].

$$y_i = \begin{cases} x_i & if\ x_i \geq 0 \\ 0 & if\ x_i < 0 \end{cases} \quad (2)$$

In this paper, ReLu is used as an activation function for the generator model as suggested by [15].

c. Leaky ReLu

Leaky ReLu is one variant of ReLu which sacrifices hard zero sparsity which has the potential to make the model more robust when doing optimization [19]. Mathematically Leaky ReLu is shown in equation (3) [16].

$$y_i = \begin{cases} x_i & if\ x_i \geq 0 \\ \frac{x_i}{a_i} & if\ x_i < 0 \end{cases} \quad (3)$$

Where a_i is a constant in range $(1, +\infty)$ so that the output of Leaky ReLu cannot be zero. In this paper, Leaky ReLu is used as an activation function for the discriminator and the value of a_i is set at 0.2 as suggested by [15].

d. 1-Dimensional CNN

What distinguishes the model used in this paper from DCGAN [15] is the use of 1-D CNN which is used instead of 2-D CNN on the GAN model in order to process time-series data. 1-D CNN was chosen because it is easier to train than RNN like LSTM or GRU, but still has good data generation results. In this paper, the 1-D CNN kernel size is 5 to increase the coverage of the 1-D CNN model [20].

e. Generator architecture

The generator's purpose is to produce a time-series data that has the same characteristics and distribution as the original data, so that the discriminator is unable to distinguish between the original data and the data generated by the generator. The Input of the generator model is in the form of a z-vector with a

dimension of 100, z-vector contains random numbers in range (-1, 1) which are uniformly distributed. The generator model in this paper has five 1-D CNN layers and five BN layers, one fully connected layer that is used to receive z-vectors, sigmoid is used as an activation function for the output layer, and RMSProp is used as an optimizer with a learning rate equal to 1e-4 and the decay rate is equal to 3e-8. The generator architecture can be seen in Figure 2.

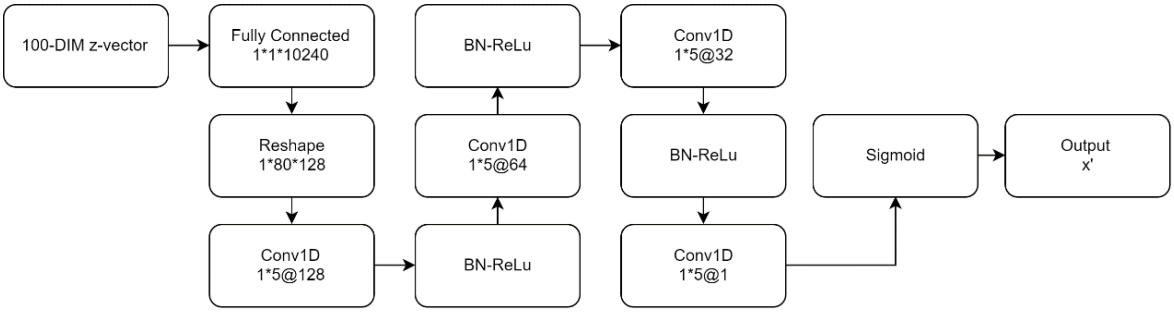

Figure 2 Generator architecture

The first three layers of Conv1D use stride equal to 2 and 1 for the last Conv1D. It aims to change the z-vector which is a vector containing 100 uniformly distributed numbers to fit the desired time series data size. In this paper, the time-series data generated by the generator is 20 sequences which are equal to the number of look back that will be the input of the classifier model for the prediction.

Data generated from the trained generator are random, therefore time-stamp matching needs to be done first to form a complete time-series data. In this paper time-stamp matching uses MSE as a metric to match the results of the data generator with real-time series data. But if there are no sequences that match the given real data on a particular time-stamp then flatline 0 will be added to the final generation result.

f. Discriminator architecture

The discriminator is used to distinguishing between real data and fake data. As shown in equation (1) generator and discriminator are playing the minimax game, the purpose of the discriminator is to make sure that the generator can generate data that is similar to real data. The discriminator model is similar to the image classifier model using CNN, there are four 1-D CNN layers that are used as feature extractors. The input of the discriminator is the real data or the fake data and given randomly. The output of the discriminator is the probability of the input being a real number in the range (0,1) and in this paper, the discriminator uses binary-crossentropy as the loss function. Because it resembles an ordinary classifier, the discriminator has a fully connected layer as the last layer before the sigmoid activation function. Discriminator uses RMSProp as an optimizer with the learning rate equal to 2e-4 and a decay rate equal to 6e-8. The Illustration of discriminator architecture can be seen in Figure 3.

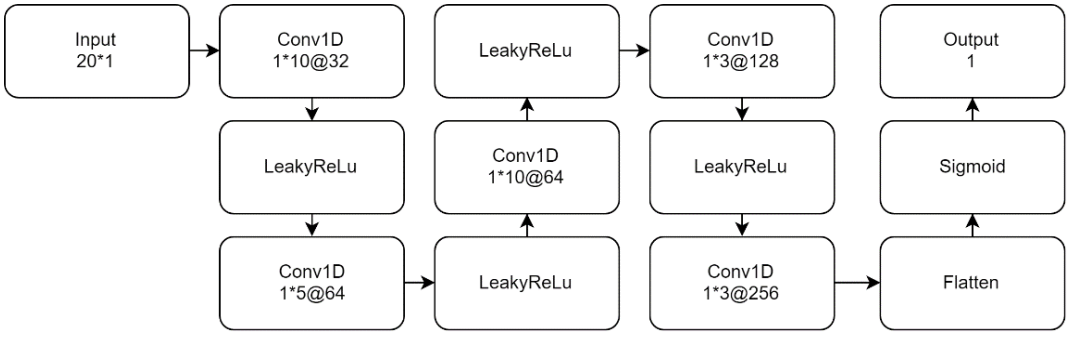

Figure 3. Discriminator architecture

2.2. Classifier Model

The combination of convolutional layers with recurrent layers such as LSTM has been shown to have performance that exceeds when using only one model [21,22]. The classifier model uses a combination of one layer 1D CNN and two LSTM layers. The purpose of this model is to predict the stock price at t + 1 by using previous sequence input by the size of look back. In this paper, the size of the look back or lag is 20. The model classifier uses RMSProp as its optimizer with a learning rate equal to 2e-4 and a decay rate equal to 6e-8. The

classifier model is trained using an early stopping scheme to prevent overfitting, the training process will stop if during 10 iterations there is no reduction in the minimum validation loss, with the maximum epoch value set at 100. The loss function used for the classifier model is a Huber loss that is more robust against outliers compared to MSE or MAE [23].

The LSTM layer is placed after the 1-D convolutional layer which has a kernel size equal to 5 and the stride is set to equal to 1. Both LSTM layers used in this paper use hyperbolic tangent (tanh) as the activation function which is the default activation function of LSTM on the Keras framework [24]. Drop-out [25] and recurrent drop-out [26] are also implemented at both LSTM layers to prevent overfitting. The illustration of classifier model architecture is shown in Figure 4.

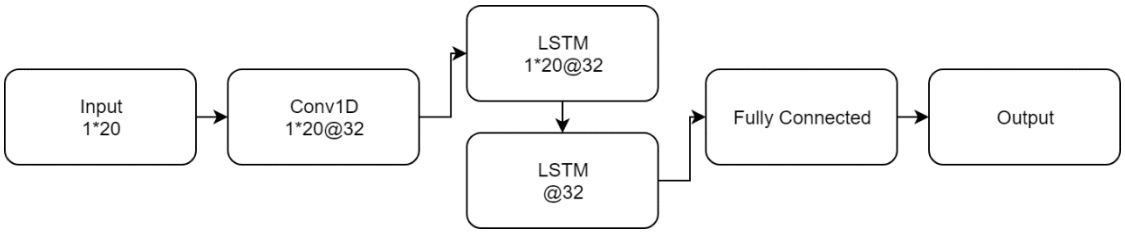

Figure 4. Classifier model architecture

2.3. Dataset

The dataset used in this paper is stock price data of Google (GOOG), Amazon (AMZN), and Facebook (FB). All three datasets are downloaded from Yahoo Finance and all three data contain stock prices between October, 4th 2018 and October, 4th 2019 with a total length of 252. The time series of data to be tested is univariate so only close prices will be used as a reference in this paper. Before being used, Minmax was used to normalize data in range (0,1). The normalized dataset was then divided into three parts: training, validation, and testing with the proportions of 40%, 40%, and 20% respectively. Data for validation is 40% of the real data and the farthest to the present time, followed by 40% for training data, newer than validation and older than testing, and 20% for testing data that is closest to the present time. The time-series data used in this paper is shown in Figure 5 - Figure 7.

Figure 5. GOOG stock price

Figure 6. AMZN stock price

Figure 7. FB stock price

2.4. Experiment scheme

This study tries to improve the performance of the classifier model by training the model with augmented data. The testing model is as follows, first the classifier model is only trained using the original data using the early stopping scheme as previously explained and tested for performance using testing data with RMSE and MAE performance metrics, after that the classifier model is re-trained using data generated from the generator GAN. The classifier performance that has been trained using original data and generator output data is evaluated once more using testing data. The experimental scheme is shown in Figure 8.

3. RESULT AND ANALYSIS

This study aims to improve the performance of time-series prediction using augmented training data generated by GAN. The evaluation uses two metrics, namely RMSE and MAE. Real and false data obtained from the generator can be seen as comparisons in Figure 9 - Figure 10 for GOOG, AMZN, and FB respectively. While testing comparisons can be seen in Figure 11 - Figure 13 and the evaluation results table can be seen in Table 1.

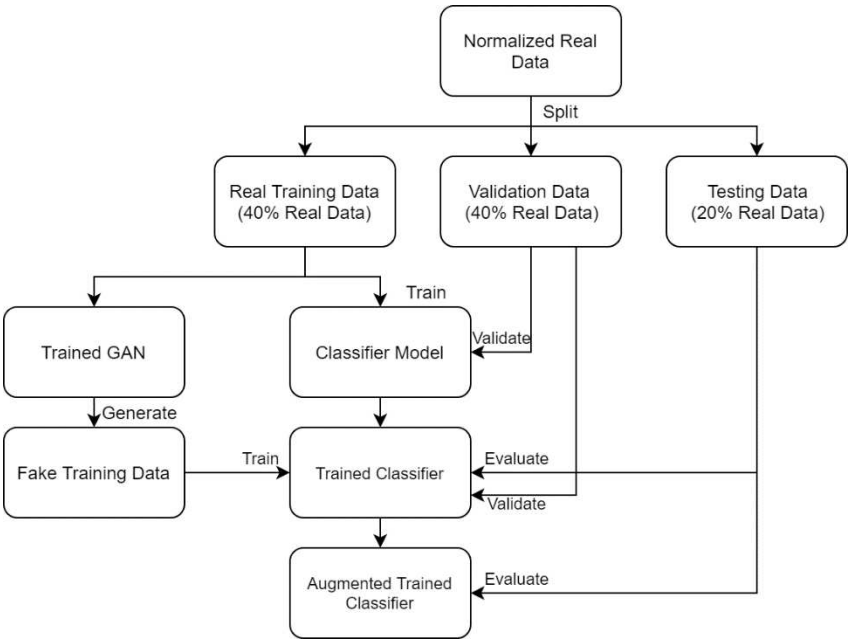

Figure 8. Experiment Scheme

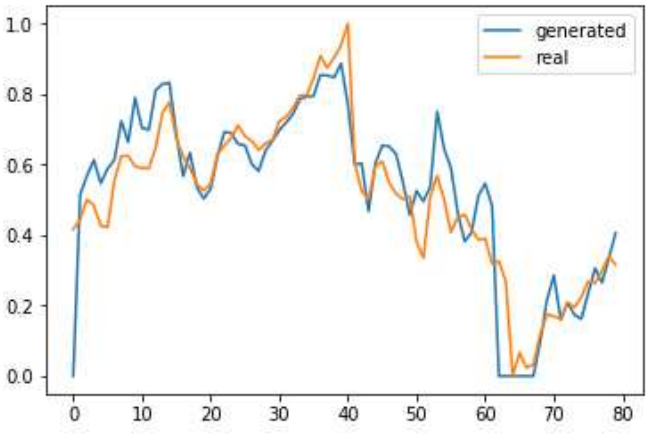

Figure 9. Fake and real comparison on GOOG train data

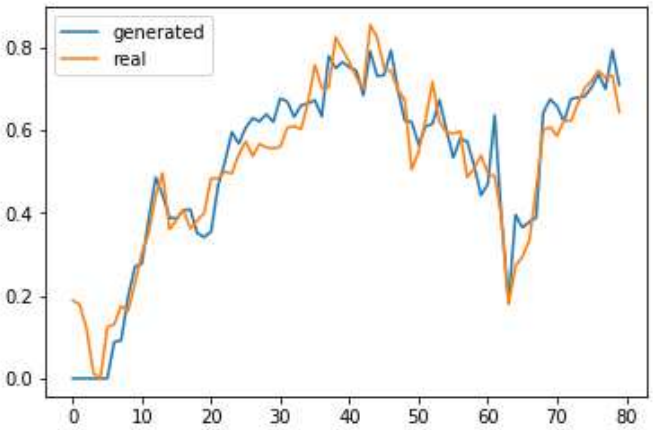

Figure 10. Fake and real comparison on AMZN train data

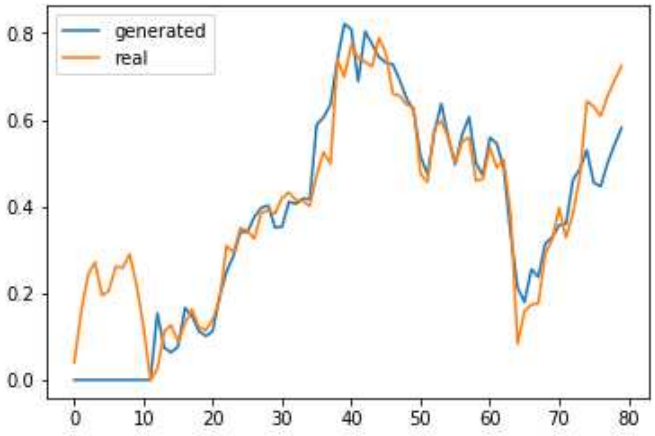

Figure 11. Fake and real comparison on FB train data

As can be seen, the GAN model used in this paper is able to generate data that is similar to the original data, although there are still some parts that are only flatlining 0, which indicates that there is no data generated by generators that successfully mimic that part.

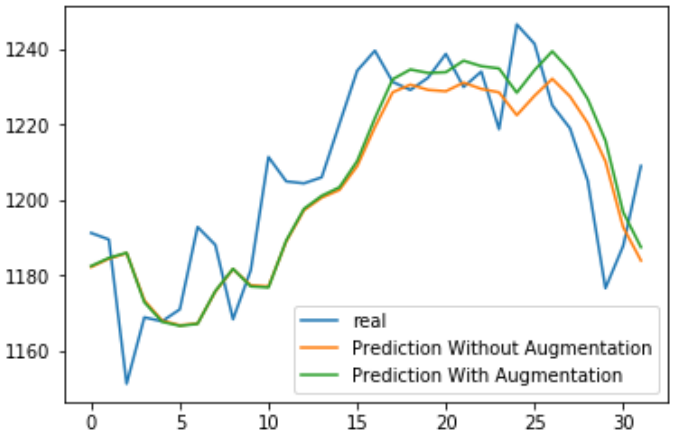

Figure 12. Prediction comparison on GOOG data

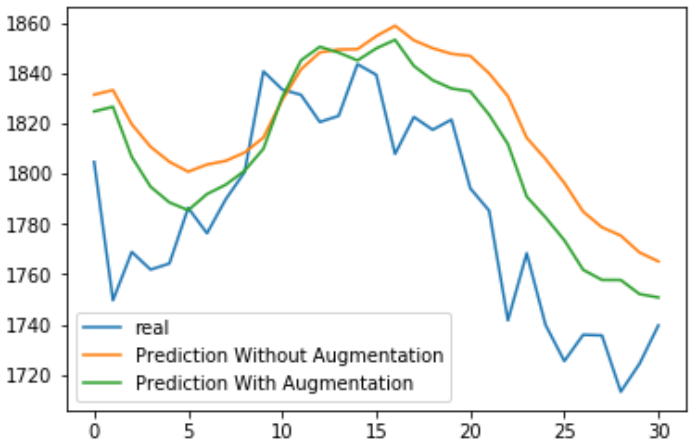

Figure 13. Prediction comparison on AMZN data

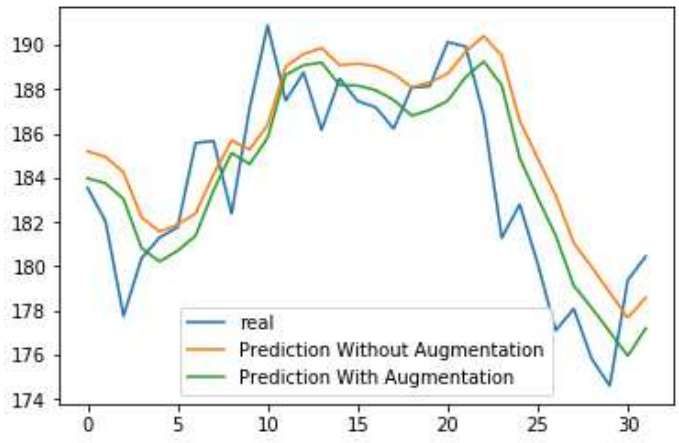

Figure 14. Prediction comparison on FB data

Table 1. Performance Metrics Comparison for Testing Data

Metrics	Real	Augmented
	GOOG	
RMSE	16.7859	16.6982
MAE	12.9452	13.2827
	AMZN	
RMSE	43.7396	32.1612
MAE	37.8935	26.4249
	FB	
RMSE	3.2645	2.7457
MAE	2.6124	2.2497

A fairly high prediction performance improvement can be seen for AMZN datasets that have a lower 26.47% and 30.27% RMSE and MAE respectively compared to just using real data. The same can also be seen for the FB dataset, which has a reduction of 15.84% and 13.88% for RMSE and MAE respectively. However, for the GOOG dataset, there were no significant changes in RMSE and MAE. There was a 0.52% decrease in RMSE and a 2.62% increase in MAE compared to just using the real data.

From the result, it can be concluded that using augmented time series data can generally improve the performance of the classifier compare to just using the real data only. The increase or decrease in classifier performance is certainly determined by the quality of fake data generated by the generator, in this paper fake time-series data used for augmentation are selected visually, which have similarities but are not exactly the same as the original data, therefore the number of epochs chosen is only 5000 compared to tens of thousands of epochs carried out by [13,15] because they have different goals than this paper.

The selection of fake time-series data for augmentation in this paper clearly has a weakness that is not based on quantitative metrics, however, this study only aims to show the potential of time-series data augmentation to improve classifier performance for time-series predictions.

4. CONCLUSION

This paper proposes a scheme to train a classifier model for predicting time series data using augmented time-series data generated using GAN. The Evaluation shows that the classifier model trained using augmented data has better performance in terms of lower RMSE and MAE for the AMZN and FB stock price, however, the evaluation using GOOG stock price dataset the proposed methodology does not show a significant change in RMSE that is 0.52% lower and even the MAE value is increased slightly by 2.62% compared to just using the real data.

REFERENCES

[1]. D. Shah, H. Isah, and F. Zulkernine, "Stock Market Analysis: A Review and Taxonomy of Prediction Techniques," International Journal of Financial Studies, vol. 7, no. 2, pp. 1–22, 2019.

[2]. X. Wu, D. Sahoo, and S. C. H. Hoi, "Recent Advances in Deep Learning for Object Detection," arXiv:1908.03673 [cs], Aug. 2019.

[3]. A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, and K. Shaalan, "Speech Recognition Using Deep Neural Networks: A Systematic Review," IEEE Access, vol. 7, pp. 19143–19165, 2019.

[4]. L. Zhang, S. Wang, and B. Liu, "Deep Learning for Sentiment Analysis : A Survey," arXiv:1801.07883 [cs, stat], Jan. 2018.

[5]. Bernal, Armando, Sam Fok, and Rohit Pidaparthi, ”Financial Market Time Series Prediction with Recurrent Neural Networks”. State College: Citeseer. 2012
[6]. Di Persio, Luca, and Oleksandr Honchar. "Recurrent neural networks approach to the financial forecast of Google assets." International Journal of Mathematics and Computers in simulation 11. 2017
[7]. B. Yang, Z. Gong, and W. Yang, “Stock market index prediction using deep neural network ensemble,” in 2017 36th Chinese Control Conference (CCC), pp. 3882–3887, 2017.
[8]. M. Hossain, R. Karim, R. Thulasiram, N. Bruce, and Y. Wang, “Hybrid Deep Learning Model for Stock Price Prediction,” 2018, pp. 1837–1844.
[9]. B. Ghojogh and M. Crowley, “The Theory Behind Overfitting, Cross Validation, Regularization, Bagging, and Boosting: Tutorial,” arXiv:1905.12787 [cs, stat], May 2019.
[10]. L. Perez and J. Wang, “The Effectiveness of Data Augmentation in Image Classification using Deep Learning,” arXiv:1712.04621 [cs], Dec. 2017.
[11]. A. L. Guennec, S. Malinowski, and R. Tavenard, “Data Augmentation for Time Series Classification using Convolutional Neural Networks,” 2016.
[12]. Z. Cui, W. Chen, and Y. Chen, “Multi-Scale Convolutional Neural Networks for Time Series Classification,” arXiv:1603.06995 [cs], Mar. 2016.
[13]. G. Ramponi, P. Protopapas, M. Brambilla, and R. Janssen, “T-CGAN: Conditional Generative Adversarial Network for Data Augmentation in Noisy Time Series with Irregular Sampling,” arXiv:1811.08295 [cs, stat], Nov. 2018.
[14]. M. Mirza and S. Osindero, “Conditional Generative Adversarial Nets,” arXiv:1411.1784 [cs, stat], Nov. 2014.
[15]. A. Radford, L. Metz, and S. Chintala, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks,” arXiv:1511.06434 [cs], Nov. 2015.
[16]. I. J. Goodfellow et al., “Generative Adversarial Networks,” arXiv:1406.2661 [cs, stat], Jun. 2014.
[17]. S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” arXiv:1502.03167 [cs], Feb. 2015.
[18]. B. Xu, N. Wang, T. Chen, and M. Li, “Empirical Evaluation of Rectified Activations in Convolutional Network,” arXiv:1505.00853 [cs, stat], May 2015.
[19]. Maas, Andrew L., Awni Y. Hannun, and Andrew Y. Ng. "Rectifier nonlinearities improve neural network acoustic models." In Proc. icml, vol. 30, no. 1, p. 3. 2013.
[20]. Atienza, Rowel. “Advanced Deep Learning with Keras.” Packt Publishing. 2018
[21]. T. Kim and H. Y. Kim, “Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data,” PLOS ONE, vol. 14, no. 2, p. e0212320, Feb. 2019.
[22]. T.-Y. Kim and S.-B. Cho, “Predicting residential energy consumption using CNN-LSTM neural networks,” Energy, vol. 182, pp. 72–81, Sep. 2019.
[23]. Niu, Jiayi, Jing Chen, and Yitian Xu. "Twin support vector regression with Huber loss." Journal of Intelligent & Fuzzy Systems 32, no. 6 pp 4247-4258. 2017
[24]. F. Chollet, Deep Learning with Python, 1st ed. Greenwich, CT, USA: Manning Publications Co., 2017.
[25]. G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” arXiv:1207.0580 [cs], Jul. 2012.
[26]. Y. Gal and Z. Ghahramani, “A Theoretically Grounded Application of Dropout in Recurrent Neural Networks,” arXiv:1512.05287 [stat], Dec. 2015.

BIBLIOGRAPHY OF AUTHORS

Julisa Bana Abraham was born in Surakarta, He received B.Eng. in nuclear engineering and M.Eng in information engineering from Universitas Gadjah Mada His research interest includes nuclear reactor safety, soft computing, deep learning, computer vision, and optimization.