Generative Adversarial Networks Applied to Stock Market Price Prediction

Ahmet Burak Yıldırım
Department of Computer Science
Özyeğin University
Istanbul, Turkey
burak.yildirim@ozu.edu.tr

Egemen İşcan
Department of Computer Science
Özyeğin University
Istanbul, Turkey
egemen.iscan@ozu.edu.tr

***Abstract*—The GAN architecture which includes the generator and the discriminator networks in its structure is generally used for producing fake data. This fake data typically comes in the form of images, however, a less popular research area is their application to time series forecasting. In this research, we have used different GAN configurations for predicting stock market prices, more specifically, the XU100 index. Our aim was to predict mainly the close price along with some other features including the technical analysis indicators. In doing this, we have also tried to determine how this neural network architecture compares to a simple LSTM regressor baseline, in terms of forecast accuracy. While testing the models, we kept the Generator fixed and changed the Discriminator structure to obtain three different GAN models for the experiments. After testing with several variations, we have observed that the best performing model of all the candidates, had an LSTM-based structure in its generator and a fully connected layer in its discriminator. This model outperformed both the baseline model and the rest of the GANs. The results were promising. Further research in the area of applying GAN models for predicting the stock market prices and all kinds of assets is strongly encouraged.**

***Index Terms*—Generative Adversarial Networks, Stock Market Price, Regression, Deep Learning, Neural Networks**

I. Introduction

The Generative Adversarial Network (GAN) [1] is a deep neural network architecture used for generating fake data from a randomly selected feature vector z. In this architecture, an unsupervised adversarial learning algorithm is applied for learning the feature probability distribution $p(z)$ of the real data X existing in the training dataset. This process is done by the networks called the generator G and the discriminator D. The generator generates fake data from a given feature vector and the discriminator learns how to differentiate the real data from the fake ones. As the discriminator learns how to differentiate them, the generator learns to produce more realistic data by getting feedback from the discriminator. Producing better fakes makes the differentiation problem more complex and the discriminator learns how to differentiate the fakes and the reals better. This minimax game played by both networks is called adversarial learning. While the generator minimizes the loss function, the discriminator aims to maximize it. The Binary Cross-Entropy (BCE) loss function used in training GAN models is given in Equation 1. Ideally, the learning is completed when the discriminator is unable to differentiate the real and the fake data anymore. The structure of GAN is shown in Figure 1.

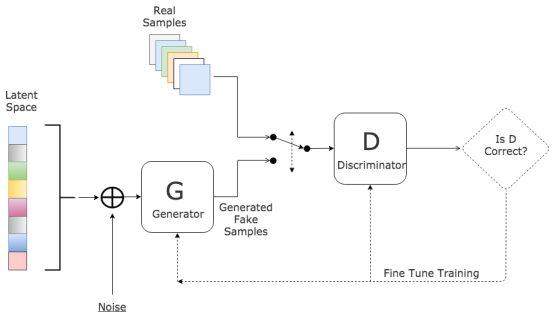


Fig. 1: Structure of Generative Adversarial Network
Retrieved from [2]: https://www.kdnuggets.com/wp-content/uploads/generative-adversarial-network.png

$$L_{BCE} = \sum_{x^t \in X} \log(D(x^t)) + \sum_{z^t \in p(z)} \log(1 - D(G(z^t))) \quad (1)$$

Since the GAN was introduced, there have been various studies in different areas to apply this network architecture in real-world scenarios such as generating fake images [3], producing realistic time series data [4], generating melody from song lyrics [5], and augmenting the training data [6]. In finance, the most-focused research areas of GANs are generating realistic time series data [7] and predicting the price of indices [8] [9] [10]. In this paper, different GAN models are constructed and trained on the daily price data of a stock market index to predict the price of the following day when the data of the past week are given. The performances of the constructed models are tested and the results are reported.

II. Methodology

The aim of this study was to determine the effect of applying the GAN architecture for producing stock market price forecasts. The BIST 100 (XU100) index is used for both the training of the models and testing their prediction performance. The daily historical data from 2000/01/05 to 2019/12/02 is extracted from the source provided by Investing.com[1]. The features used in training are selected as follows:

[1] https://www.investing.com/indices/ise-100-historical-data

- **Close Price:** The last transacted price of the index
- **Open Price:** The first price at the opening of the index
- **Highest Price:** The highest price of the index in a day
- **Lowest Price:** The lowest price of the index in a day
- **RSI 14:** Relative Strength Index of 14 days
- **SMA 5:** Simple Moving Average of 5 days
- **CUMLOGRET 1:** Cumulative Log Return of a single day
- **MACD (12, 26, 9):** Moving Average Convergence Divergence over 12 (2 weeks), 26 (1 month) and 9 (1.5 week) days

While the first 4 features are taken directly from the dataset, the rest of the technical indicators are derived by using the initial data. After obtaining the historical data belonging to the 8 features, all the rows that contain NaN values are removed and the values are normalized by using the Min-Max Scaler. 4252 rows remained after the preprocessing step. Finally, the data is converted to overlapping sequences of 7 days. The first 80% of the data is set as the training set while the rest of the data is split into half (10%-10%) as both the validation and the test sets, respectively.

For the experiments, 1 simple regressor and 3 different GAN models with minor modifications are implemented in Python via the PyTorch library. The simple regressor model is chosen as a baseline to compare its prediction performance with the 3 different GAN models. In these GAN models, the generator structures were kept similar and the structure of the discriminators are designed by an artificial neural network (ANN), long short-term memory (LSTM), and gated recurrent units (GRU) architectures. While the ANN is a simple neural network, LSTM and GRU are recurrent neural network (RNN) architectures mostly used for processing the sequences of data. The differences between LSTM and GRU are the gate numbers inside the units and their working principles which are shown in Figure 2.

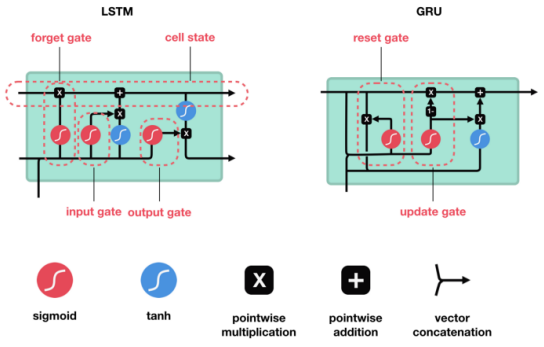


Fig. 2: LSTM vs. GRU
Retrieved from [11]:
https://miro.medium.com/max/2400/1*yBXV9o5q7L_CvY7quJt3WQ.png

LSTM architecture is used for the generator to predict the feature values of the next day from the given historical data of the past 6 days instead of using sampled z values from the Gaussian distribution as applied in the default GAN models.

For the hyperparameter tuning part; the hidden layer sizes, learning rate, number of epochs, and sequence length (number of timestamps used for training data sequences) were configured. In order to systematically compare and evaluate the models, the Root Mean Squared Error (RMSE) is chosen as an error metric (see Equation 2), and calculated for each feature in the validation set during the training phase.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - x_i)^2}{n}} \tag{2}$$

By the end of the training process, the best-performing models are obtained according to the lowest close price RMSE values on the validation dataset. Then, to show the prediction performance, the predictions of the model on the test dataset were plotted by using the Matplotlib library, and the RMSE values are reported.

III. Experimental Setup

For the experiments, the models mentioned in the Methodology section are implemented and their hyperparameters are tuned. Adam is selected as the stochastic gradient descent optimization algorithm and the number of epochs is set to $100k$. On each 500 epoch, the performances of the trained models are evaluated on the validation dataset. The sequence length of the historical data and the batch size are set to 7 and 64 respectively. In the GAN models, the learning rates were kept the same for both the generator and the discriminator. While the learning rate is set to 5.125×10^{-4} for the GAN models, it is set to 5×10^{-4} for the simple regressor baseline. The design details of the implemented models are given in the following sections.

A. Simple LSTM Regressor

LSTM network is used for the simple regressor and the model is set as a baseline to compare its performance with the GAN models. During the training, historical data of 6 days are fed into the model and the future values for the next day are produced as an output. The Mean Squared Error (see Equation 3) between the real values of the target and the predicted values is backpropagated during the training phase. This structure is given in Figure 3.

$$MSE = \frac{\sum_{i=1}^{n}(y_i - x_i)^2}{n} \tag{3}$$

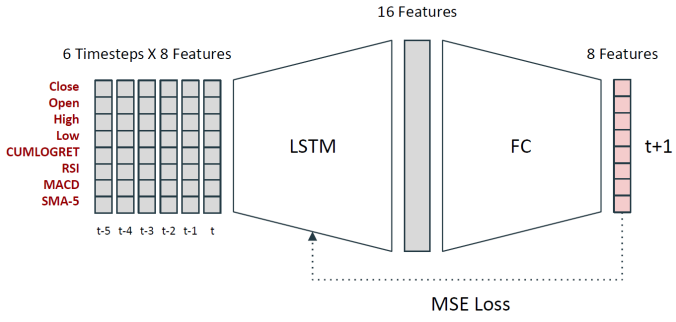


Fig. 3: Structure of Simple LSTM Regressor

B. GAN with ANN Discriminator

The configuration of the Simple LSTM Regressor network is adopted by the generator network of the GAN architecture. The discriminator network is designed as an ANN that takes a single vector of concatenated feature values of 7 days. For the fake data, the predicted feature vector is concatenated with the real feature vectors of the past 6 days and the resulting vector is labeled as fake. On the other hand, the real labels are the concatenated feature vectors of the 7 days taken from the training dataset. As it is explained in the Introduction, the discriminator learns how to differentiate these real data from fake ones. The BCE loss function is used for the training. The organization of the model with its hidden layer sizes is shown in Figure 4.

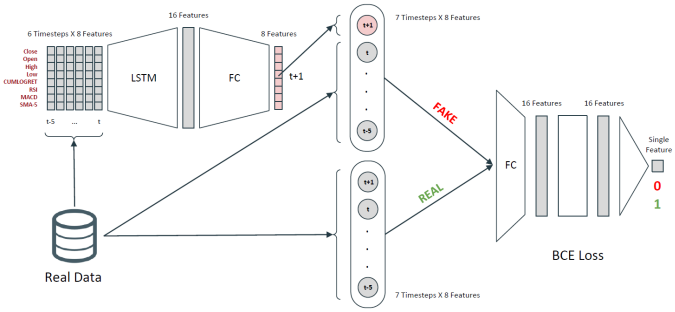


Fig. 4: Structure of GAN with ANN Discriminator

C. GAN with LSTM Discriminator

The generator design is kept similar to the GAN with ANN model however the discriminator is converted to LSTM network. Since the LSTM network takes sequential data as input, instead of creating a single vector by concatenating the feature vectors of the 7 days, the feature vectors are given as sequential data. The set-up for this model is shown in Figure 5.

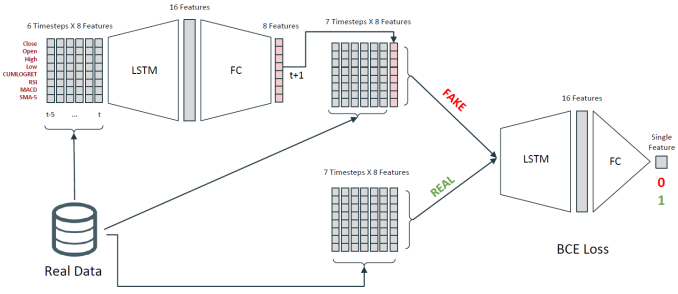


Fig. 5: Structure of GAN with LSTM Discriminator

D. GAN with GRU Discriminator

This model is identical with the GAN having an LSTM discriminator apart from the slight difference that the GRU is used instead of the LSTM network for the discriminator. The structure is shown in Figure 6.

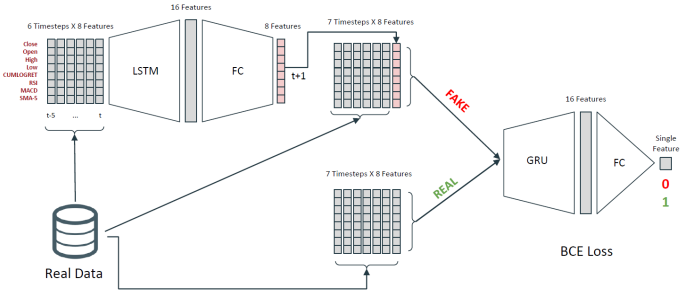


Fig. 6: Structure of GAN with GRU Discriminator

IV. Results and Discussion

According to the close price RMSE results on the validation dataset, the best performing weights for each model are determined. The epoch numbers of the models performed best and their close price RMSE results on validation and test dataset are given in Table I.

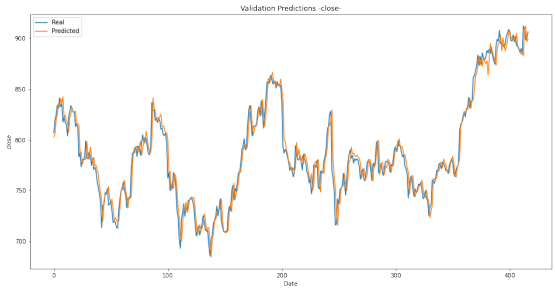
	Epoch	RMSE Validation	RMSE Test
Simple LSTM Regressor	37500	10.8510	50.2491
GAN with ANN Disc.	13500	15.8889	31.3936
GAN with LSTM Disc.	98000	13.9889	64.4466
GAN with GRU Disc.	12000	18.9767	66.5857

TABLE I: Overall performance results of the models

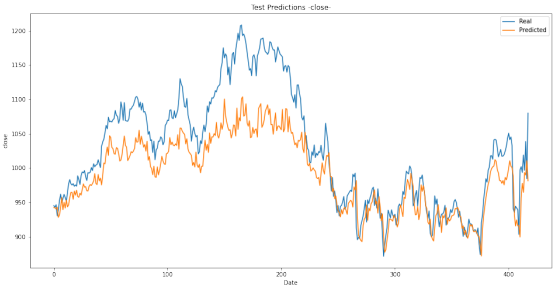
When the performance of all the models on the validation dataset is compared, the simple LSTM regressor, or the baseline, was the best-performing model. However, when the close price RMSE results on the test dataset are considered, they were actually worse than the validation errors. The prediction results of the model on validation and test dataset are shown in Figure 7. It can be concluded that, although the hyperparameters are tuned, the models are still overfitted since they could not predict the close price values in the test dataset which are higher than the values of the training set. The RMSE results (see Figure A.1) and the prediction (see Figure A.2) performances on each feature are given in Appendix A.

The GAN model outperforming the base model was the one having an ANN discriminator. Despite the model is ranked third according to the performance on the validation dataset, it achieved the best performance on the test dataset with a significant distinction. According to Figure 8, it may be observed that the model could generalize the regression model instead of overfitting the training dataset as the baseline model.

Observe that the GAN models having discriminator with RNN networks which are LSTM and GRU failed on RMSE results on the test dataset. The overall results of the GAN models with ANN, LSTM and GRU are given in Appendix B, C and D respectively.

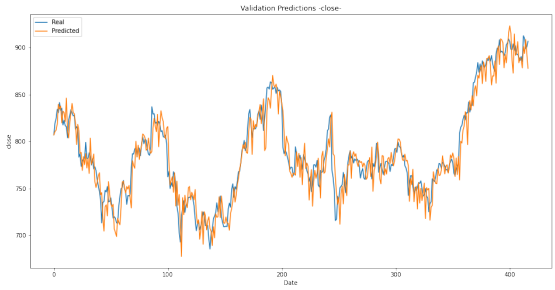


(a) Validation Data

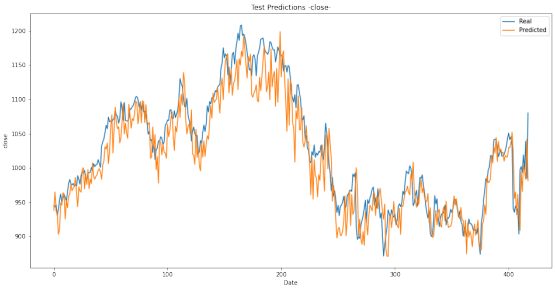


(b) Test Data

Fig. 7: Prediction performance of the simple LSTM Regressor



(a) Validation Data



(b) Test Data

Fig. 8: Prediction performance of the GAN with ANN discriminator

Since the GANs are highly sensitive to even the smallest modifications in the hyperparameters due to the adversarial learning algorithm, further tuning them may create some room for improvement. The experiments indicate that using the ANN network increased the performance of the GAN models. Another consequence is that using models with greater complexity which may be obtained by increasing the layer sizes may result in overfitting. Despite the close price range of the validation dataset is closer to the values in the training dataset, these overfitted models even have failed significantly on the validation dataset.

V. Conclusion

This research demonstrates the possibility to use the GAN architecture for predicting the price of the assets. One implication of this statement is that GANs may also be used to generate forecasts, ”potentially” on any data in the form of time series. According to the results of the experiments, constructing a simple LSTM regressor, base model, could not perform well on the historical data of the XU100 index. It could learn the relation between the feature values inside the training data. However, when the range of the values is increased in the future (validation) data, the predictions could not result higher than the value ranges on the training dataset. Thus, it is concluded that the model is overfitted. On the other hand, various GAN models having different types of discriminator networks such as ANN, LSTM, and GRU are tested by keeping the generator structure fixed. The results showed that despite the base model outperformed all the models according to the close price RMSE results on the validation dataset, the GAN model having ANN discriminator beat the base model on the test dataset significantly. The performance difference is caused by the higher value ranges existing in the test dataset. Thus, it can be stated that the GAN model could learn how to generalize the data better than the LSTM regressor. The rest of the GAN models having LSTM and GRU discriminator failed to predict the price data. Since the GAN models are very sensitive to hyperparameters due to adversarial learning, further tuning the parameters may increase their performance. However, in the experiments, this hasn’t been the case. Lastly, the experiments showed that using more complex networks for the generator or the discriminator models resulted in higher close price RMSE values because of the overfitting issue.

VI. Future Work

While drawing conclusions from this study, it is crucial to keep in mind that these results rely on one data source and a single baseline. To confirm that these results are not misleading in some way, more experimentation with a wide range of data sources belonging to different assets, indices, or even targets that are irrelevant to the stock market, should be done. Additionally, it is evident that the model performances still have some room for improvement. This may be achieved by inspecting in detail how the different hyperparameters affect the training or testing the models with different features. These

features may include additional technical analysis indicators such as the volume, the stochastic oscillator, the Aaron oscillator, etc. Improving the implementation of the models would lead to decreased RMSE values and better results. It may also be helpful to investigate the correlation between the inflation rate and the price of an index. Furthermore, the factors which cause poor prediction performance and overfitting should be determined and fixed.

VII. GitHub Repository

All the datasets, models, their implementations, and the results are given in the following GitHub repository: https://github.com/abyildirim/Stock-Market-Price-Prediction

VIII. Acknowledgements

We wish to acknowledge the guidance provided by Dr. Emre Sefer at each step of this study.

References

[1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio (2014). Generative Adversarial Networks. Retrieved from https://arxiv.org/pdf/1406.2661.pdf

[2] Al Gharakhanian (2017). Generative Adversarial Networks – Hot Topic in Machine Learning. Retrieved from https://www.kdnuggets.com/2017/01/generative-adversarial-networks-hot-topic-machine-learning.html

[3] Alec Radford, Luke Metz, Soumith Chintala (2016). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. Retrieved from https://arxiv.org/pdf/1511.06434.pdf

[4] Cristóbal Esteban, Stephanie L. Hyland, Gunnar Rätsch (2017). Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs. Retrieved from https://arxiv.org/pdf/1706.02633.pdf

[5] Yi Yu, Abhishek Srivastava, Simon Canales (2019). Conditional LSTM-GAN for Melody Generation from Lyrics. Retrieved from https://arxiv.org/pdf/1908.05551.pdf

[6] Ngoc-Trung Tran, Viet-Hung Tran, Ngoc-Bao Nguyen, Trung-Kien Nguyen, Ngai-Man Cheung (2020). On Data Augmentation for GAN Training. Retrieved from https://arxiv.org/pdf/2006.05338.pdf

[7] Jinsung Yoon, Daniel Jarrett, Mihaela van der Schaar (2019). Time-series Generative Adversarial Networks. Retrieved from https://papers.nips.cc/paper/2019/file/c9efe5f26cd17ba6216bbe2a7d26d490-Paper.pdf

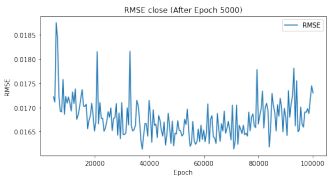
[8] Ricardo Alberto Carrillo Romero (2019). Generative Adversarial Network for Stock Market price Prediction. Retrieved from https://cs230.stanford.edu/projects_fall_2019/reports/26259829.pdf

[9] Kevin Thomas (2019). Time Series Prediction for Stock Price and Opioid Incident Location. Retrieved from https://core.ac.uk/download/pdf/240229094.pdf

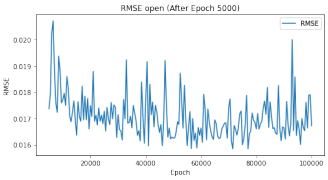
[10] Kang Zhang, Guoqiang Zhong, Junyu Dong, Shengke Wang, Yong Wang (2019). Stock Market Prediction Based on Generative Adversarial Network. Retrieved from https://www.researchgate.net/publication/331002527_Stock_Market_Prediction_Based_on_Generative_Adversarial_Network

[11] Michael Phi (2018). Illustrated Guide to LSTM's and GRU's: A step by step explanation. Retrieved from https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

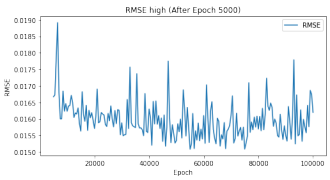
APPENDIX A
OUTCOMES OF THE SIMPLE LSTM REGRESSOR



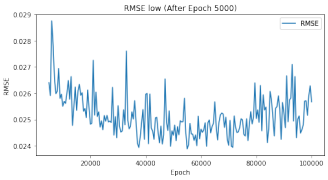
(a) Close Price



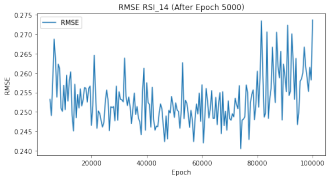
(b) Open Price



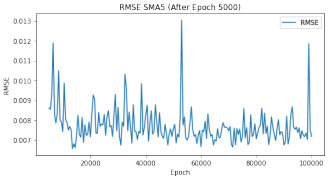
(c) Highest Price



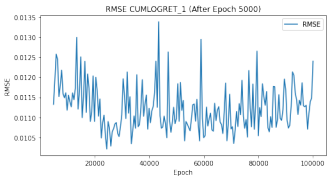
(d) Lowest Price



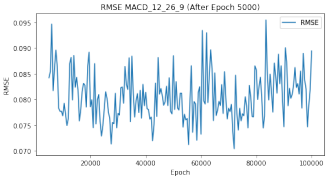
(e) Relative Strength Index (RSI_14)



(f) Simple Moving Average (SMA_5)

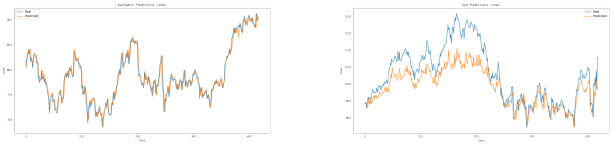


(g) Cumulative Log Return (CUMLOGRET_1)

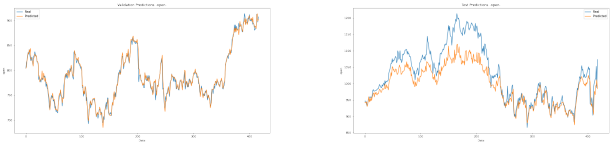


(h) Moving Average Convergence / Divergence (MACD_12_26_9))

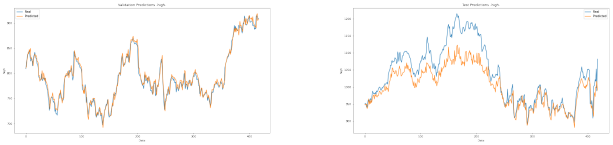
Fig. A.1: RMSE values of each training feature on normalized validation data



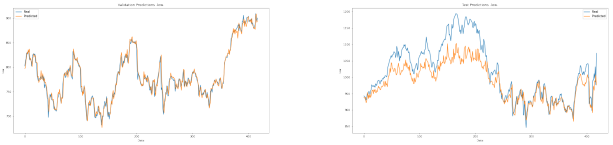
(a) Close Price



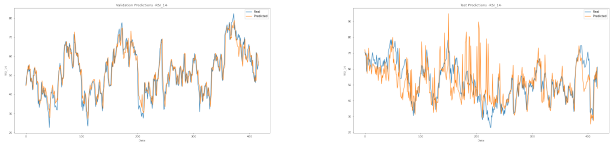
(b) Open Price



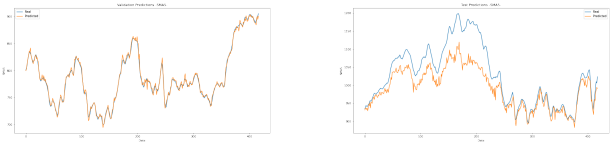
(c) Highest Price



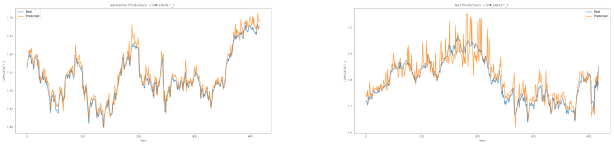
(d) Lowest Price



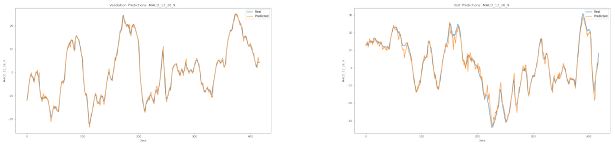
(e) Relative Strength Index (RSI_14)



(f) Simple Moving Average (SMA_5)



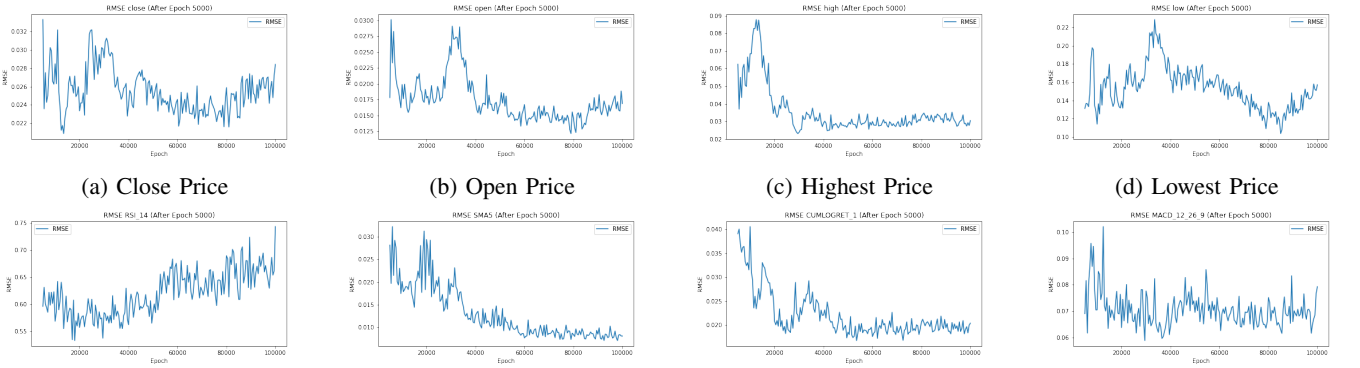
(g) Cumulative Log Return (CUMLOGRET_1)



(h) Moving Average Convergence / Divergence (MACD_12_26_9))

Fig. A.2: Predicted values of each training feature on validation and test dataset

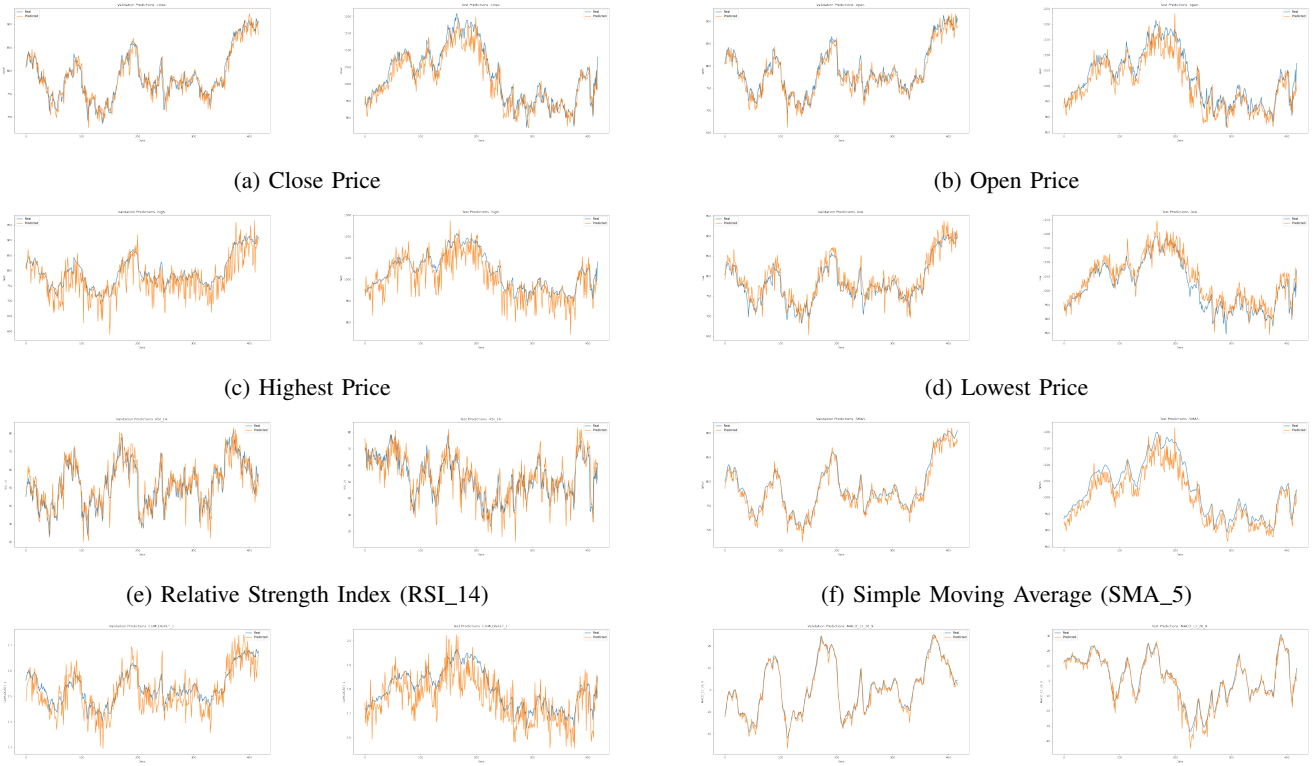
APPENDIX B
OUTCOMES OF THE GAN WITH ANN DISCRIMINATOR



(a) Close Price (b) Open Price (c) Highest Price (d) Lowest Price

(e) Relative Strength Index (RSI_14) (f) Simple Moving Average (SMA_5) (g) Cumulative Log Return (CUMLOGRET_1) (h) Moving Average Convergence / Divergence (MACD_12_26_9))

Fig. B.1: RMSE values of each training feature on normalized validation data



(a) Close Price (b) Open Price

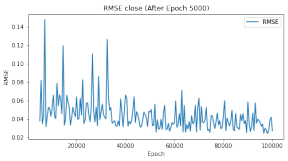
(c) Highest Price (d) Lowest Price

(e) Relative Strength Index (RSI_14) (f) Simple Moving Average (SMA_5)

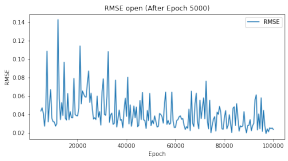
(g) Cumulative Log Return (CUMLOGRET_1) (h) Moving Average Convergence / Divergence (MACD_12_26_9))

Fig. B.2: Predicted values of each training feature on validation and test dataset

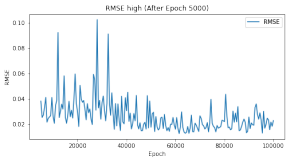
APPENDIX C
OUTCOMES OF THE GAN WITH LSTM DISCRIMINATOR



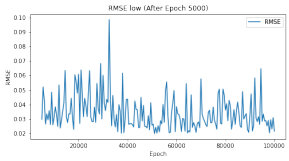
(a) Close Price



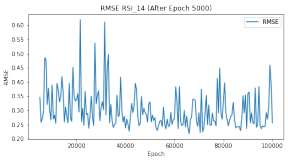
(b) Open Price



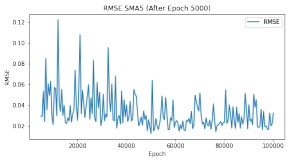
(c) Highest Price



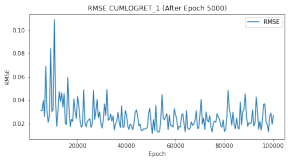
(d) Lowest Price



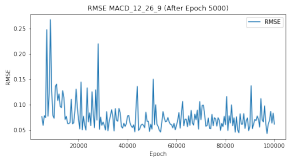
(e) Relative Strength Index (RSI_14)



(f) Simple Moving Average (SMA_5)

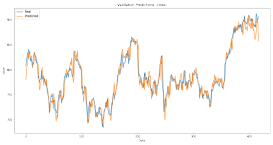


(g) Cumulative Log Return (CUMLOGRET_1)



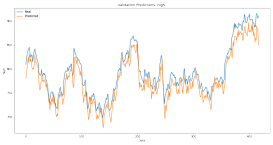
(h) Moving Average Convergence / Divergence (MACD_12_26_9))

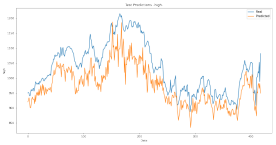
Fig. C.1: RMSE values of each training feature on normalized validation data



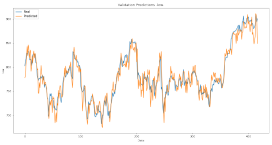
(a) Close Price

(b) Open Price



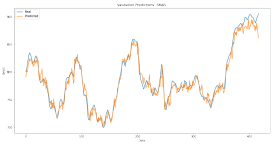


(c) Highest Price

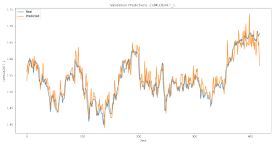


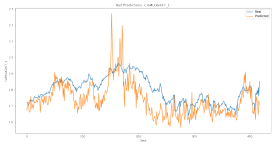
(d) Lowest Price

(e) Relative Strength Index (RSI_14)

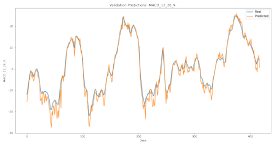


(f) Simple Moving Average (SMA_5)





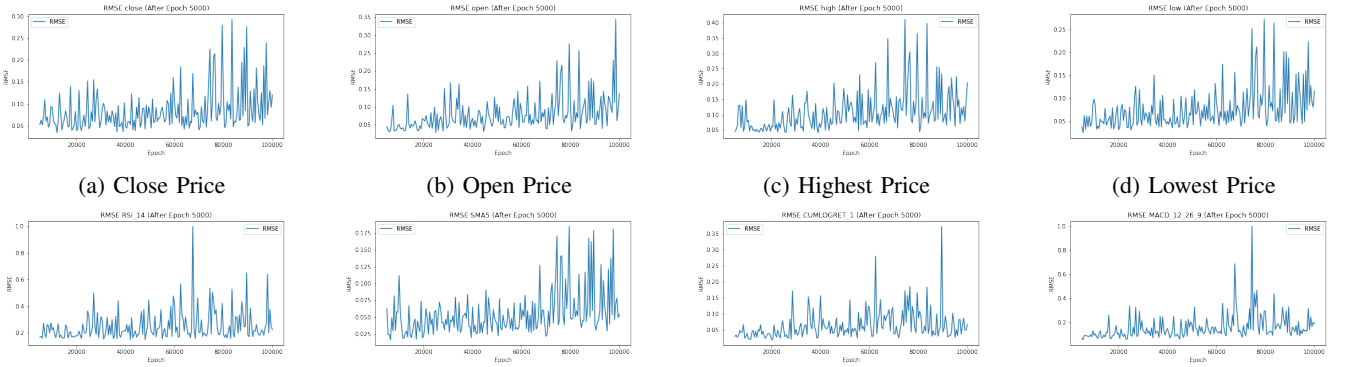
(g) Cumulative Log Return (CUMLOGRET_1)



(h) Moving Average Convergence / Divergence (MACD_12_26_9))

Fig. C.2: Predicted values of each training feature on validation and test dataset

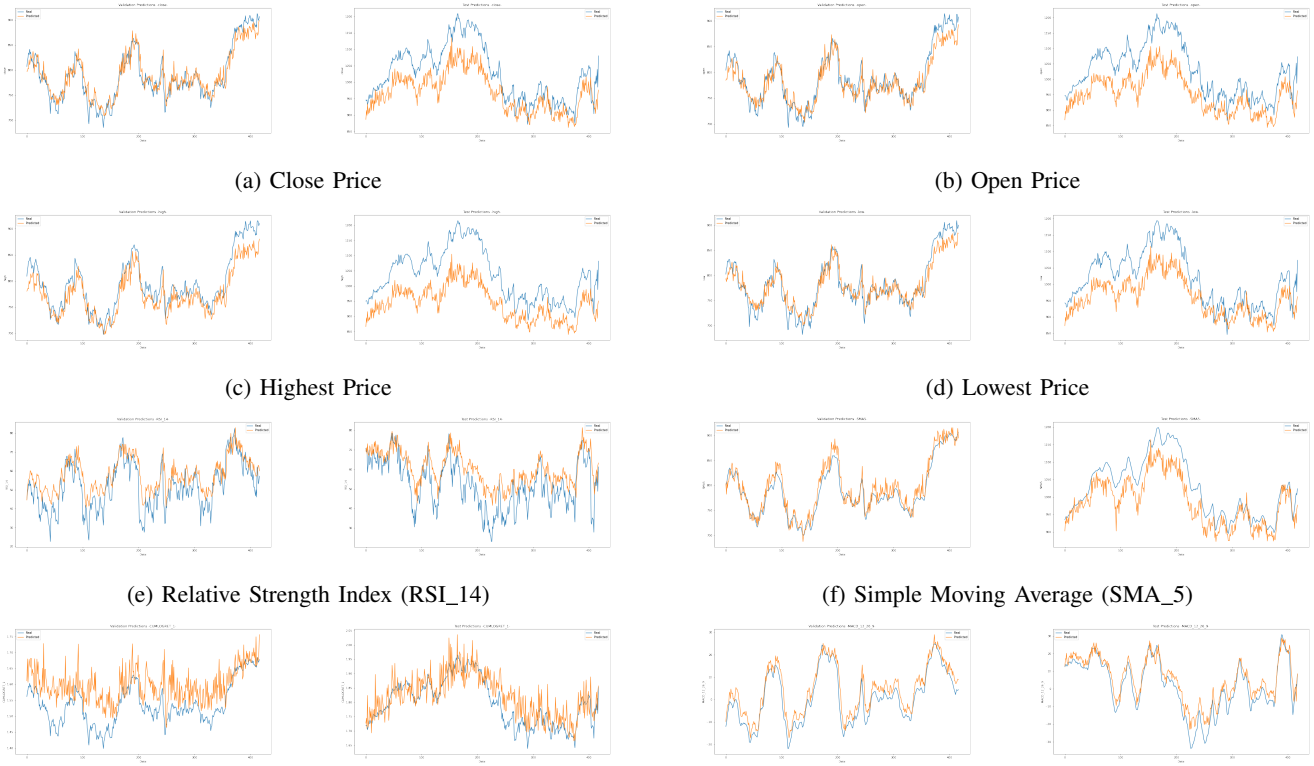
APPENDIX D
OUTCOMES OF THE GAN WITH GRU DISCRIMINATOR



(a) Close Price (b) Open Price (c) Highest Price (d) Lowest Price

(e) Relative Strength Index (RSI_14) (f) Simple Moving Average (SMA_5) (g) Cumulative Log Return (CUMLOGRET_1) (h) Moving Average Convergence / Divergence (MACD_12_26_9))

Fig. D.1: RMSE values of each training feature on normalized validation data



(a) Close Price (b) Open Price

(c) Highest Price (d) Lowest Price

(e) Relative Strength Index (RSI_14) (f) Simple Moving Average (SMA_5)

(g) Cumulative Log Return (CUMLOGRET_1) (h) Moving Average Convergence / Divergence (MACD_12_26_9))

Fig. D.2: Predicted values of each training feature on validation and test dataset