Generating Synthetic Financial Time-Series Data Through Generative Adversarial Network

Methodologies And Comparison Among Different Deep Learning Structures

Grant Sawyer*[1], Harold Yuan†[1], Kaushik Tallam‡[1], Mario Nicolo' De Matteis§[1], and Tristan Roemer¶[1]

[1]Carnegie Mellon University, MSCF, New York, USA

February 2025

Abstract

Generative Adversarial Networks (GANs) have emerged as a powerful tool for synthesizing realistic financial time series data by addressing challenges such as data scarcity, privacy, and cost. This paper explores the integration of attention mechanisms, convolutional neural networks (CNNs), and long short-term memory (LSTM) layers within a GAN framework to generate synthetic financial price data. We evaluate the performance of these architectures using three financial instruments: JPMorgan Chase (JPM), Apple Inc. (AAPL), and the MSCI ACWI Index (ACWI). The data spans over two decades of market data. Our results demonstrate that CNN enhanced GANs significantly improve the modeling of temporal dependencies and volatility clustering compared to Attention and LSTM based variants. The synthetic data exhibits statistical properties that align with real world financial time series which is validated by tests for distributional similarity and moments. These findings provide actionable insights for financial institutions seeking to augment datasets for backtesting, risk management, and machine learning applications.

Keywords: Generative Adversatial Networks, Financial Time-Series, Attention Layers

1 Introduction

Hedge Funds, Proprietary Trading Firms, and other financial institutions rely on large amounts of data to make informed decisions. However, obtaining high quality financial data can be challenging due to data scarcity, privacy, and cost. As a result, researchers developed generative models that can synthetically generate realistic financial time series data. These models can be used to augment existing datasets, generate new data for backtesting trading strategies, and simulate market conditions for risk management. Over the last few years, many different developments have been conducted in the field of synthetic financial time series data generation. In this paper, we provide a comprehensive overview of the methodologies and comparisons among different deep learning structures used to generate synthetic financial time series data. In particular, we aim to synthesize new price time series data. The data is intended to be used for

*gsawyer@andrew.cmu.edu
†zhongfay@andrew.cmu.edu
‡ktallam@andrew.cmu.edu
§mdematte@andrew.cmu.edu
¶troemer@andrew.cmu.edu

backtesting trading strategies, training machine learning models, and conducting stress tests and scenario analyses. By generating realistic synthetic data, we can provide financial institutions with a valuable resource to enhance their decision making processes, improve model robustness, and better prepare for adverse market conditions.

Given the significant impact and popularity of generative AI models, we decided to utilize one of the main pillars of Generative AI, the Generative Adversarial Network (GAN) develope by Goodfellow et al. (2014). GANs have been used mainly in the computer vision field, particularly in image generation. However, Iglesias, Talavera, and Díaz-Álvarez (2023) show that the ability to generate realistic data samples through adversarial training makes GANs a powerful tool for synthesizing financial time series data. This groundbreaking model has subsequently influenced research by demonstrating that adversarial training can generate realistic data samples. Eckerli and Osterrieder (2021) mentions different applications of GANs, including market manipulation, credit card fraud, portfolio optimization and market prediction.

Several other aspects are also significantly important in the realm of synthetic financial data. For example, robust data augmentation techniques can enhance model reliability, and careful evaluation of synthetic data against real world financial indicators is crucial for ensuring its practicality. Moreover, addressing the temporal dependencies and non-linear dynamics inherent in financial markets requires advanced modeling techniques that go beyond the original GAN framework. Put simply, we aim to find the distribution by which the prices have been generated. As highlighted by Cont (2001), understanding the empirical properties of asset returns, such as heavy tails and volatility clustering, is essential for developing realistic financial models. These properties must be carefully considered when designing synthetic data generation frameworks to ensure that the generated data accurately reflects the behavior observed in real financial markets.

We designed the neural networks for the GAN models by implementing three different types of layers: Attention, CNN, and LSTM. We decided to implement these three different types of layers as each offers unique advantages for capturing the complex dynamics of financial time series data.

- **Attention Layers**: To capture and prioritize significant temporal patterns, thereby enabling the model to focus on key segments of the time series.
- **Convolutional Neural Networks (CNNs)**: To detect localized patterns and extract spatial features from financial time series data.
- **Long Short-Term Memory (LSTM) Networks**: To model long-range dependencies and effectively capture the sequential characteristics of financial data.

Through a comparative analysis, this study evaluates the performance of these architectures in generating synthetic financial time series data. In particular, we investigate how the integration of attention mechanisms can enhance the synthesis process by more accurately reflecting the intricate dynamics of financial markets.

The project code, datasets, and additional resources are available at our GitHub repository: `https://github.com/mariodematteis/ml-ii-final-project`.

2 Data and Variables of Interest

The purpose of this paper is to generate synthetic time series data from a Generative Adversarial Network (GAN) framework. To that end, it is important to carefully consider what financial instruments to use for training and testing the model so that it is both robust and easily generalized. The three instruments that we chose to analyze in this paper are JPMorgan Chase (JPM), Apple Inc. (AAPL), and the MSCI ACWI Index (ACWI). We specifically chose these equity securities because they allow us to assess the performance of our model across both individual equities and a broad market index, which when put together, helps us represent a wide array of market dynamics.

We sourced the time series data that was used in this paper from the Wharton Research Data Services (WRDS) platform, which provided access to very high quality daily historical data. We extracted data from January 1, 2000 to January 31, 2024 for tickers JPM and APPL. We chose a large time frame in order to cover multiple market cycles and include significant past economic events, which should potentially allow the model to capture the complex dynamics that are inherent in the financial markets over time. For example, the model will be able to see both periods of high volatility, like the 2008 financial crisis and the 2020 Covid-19 pandemic, as well as the periods of relative stability in between.

For the MSCI ACWI index, we decided to extract data from the iShares MSCI ACWI ETF which tracks the overall performance of the MSCI ACWI index. However, since this fund was created in 2008, we were only able to extract data from March 28, 2008 to January 31, 2024. Even though we have less time series data for the index compared to the individual stock data, because the index represents a well diversified portfolio of global equities, in both emerging and developed markets, the context it provides is important for evaluating the generalizability of the synthetic data that our model generates across different market conditions and geographic regions.

By using these datasets, we can comprehensively evaluate our GAN based approach. In the following sections, we will describe the preprocessing steps that were applied to the raw data, including cleaning, normalization, and the transformation procedures, that were used to make the data more suitable for model training.

3 Model

Generative Adversarial Networks (GANs) provide a framework to learn complex data distributions through a two-player minimax game involving two neural networks: a **generator** and a **discriminator**. The classical formulation of GANs is outlined below, after which we describe our enhanced generator architectures that incorporate attention, convolutional, and long short-term memory layers.

Setup and Notation

- **Real Data Distribution:** Let $p_{\text{data}}(x)$ denote the probability distribution of real data samples $x \in \mathcal{X}$.
- **Latent Space and Prior:** Define a latent space $\mathcal{Z}$ with a simple prior distribution $p_z(z)$ (e.g., a Gaussian or uniform distribution). A latent variable $z \sim p_z(z)$ is sampled and then transformed into the data space.
- **Generator:** The generator is a function $G : \mathcal{Z} \to \mathcal{X}$ parameterized by θ_G. It maps a latent variable z to a synthetic sample $G(z)$, thereby inducing an implicit distribution $p_g(x)$ over the data space.
- **Discriminator:** The discriminator is a function $D : \mathcal{X} \to [0, 1]$ parameterized by θ_D. It outputs a scalar representing the probability that a given sample x originates from the real data distribution $p_{\text{data}}(x)$ rather than from $p_g(x)$.

The Minimax Game

The GAN framework is formulated as a two-player minimax game with the value function

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

- **Discriminator's Objective:** For a fixed generator G, the discriminator D is trained to maximize the probability of correctly classifying real data and generated data:

$$\mathbb{E}_{x \sim p_{\text{data}}(x)} \big[\log D(x) \big] + \mathbb{E}_{x \sim p_g(x)} \big[\log(1 - D(x)) \big].$$

- **Generator's Objective:** Simultaneously, the generator G is trained to minimize the same objective (i.e., to "fool" D) by generating samples $G(z)$ that maximize the discriminator's misclassification:

$$\min_G \mathbb{E}_{z \sim p_z(z)} \big[\log(1 - D(G(z))) \big].$$

Optimal Discriminator and Jensen-Shannon Divergence

For any fixed generator G, the optimal discriminator $D_G^*(x)$ is obtained by maximizing the value function pointwise:

$$f(D(x)) = p_{\text{data}}(x) \log D(x) + p_g(x) \log(1 - D(x)).$$

Taking the derivative with respect to $D(x)$ and setting it to zero leads to:

$$\frac{p_{\text{data}}(x)}{D(x)} - \frac{p_g(x)}{1 - D(x)} = 0 \quad \Longrightarrow \quad D_G^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)}.$$

Substituting D_G^* back into the value function yields:

$$V(G, D_G^*) = -\log(4) + 2\,\text{JSD}\big(p_{\text{data}} \,\|\, p_g\big),$$

where JSD denotes the Jensen-Shannon Divergence. Minimizing $V(G, D_G^*)$ with respect to G forces $p_g(x)$ to converge to $p_{\text{data}}(x)$.

Training Procedure

In practice, GAN training alternates between:

1. **Discriminator Update:** Maximize $V(D, G)$ with respect to θ_D, keeping θ_G fixed.
2. **Generator Update:** Minimize $V(D, G)$ (or a modified loss, such as maximizing $\log D(G(z))$ for stronger gradients) with respect to θ_G, while keeping θ_D fixed.

These updates are typically performed using stochastic gradient descent or its variants. In our specific case, for each of the model trained we used the Adam optimizer with a learning rate of 0.0002 for both the Discriminator and the Generator.

Enhanced Generator Architectures

While the classical GAN framework provides a robust foundation for generative modeling, financial time series data pose unique challenges due to their complex temporal dependencies and nonlinear structures. To address these issues, we extend the generator architecture by incorporating advanced neural network components, as detailed below.

Attention Layers

Attention mechanisms have revolutionized various fields, particularly natural language processing, by allowing models to dynamically focus on different parts of the input sequence Vaswani et al. (2023). In our generator, we incorporate a Self-Attention layer inspired by this work. The attention mechanism computes a weighted sum of feature representations across time steps, enabling the model to prioritize significant temporal patterns. This approach is particularly beneficial for capturing long-range dependencies and subtle patterns in financial data. In quantitative finance, attention-based models have garnered significant interest; for example, Qin et al. (2017) introduced a dual-stage attention-based recurrent neural network for time series prediction, while Chen and Ge (2019) leveraged attention mechanisms for enhanced stock price forecasting. Our implementation integrates a series of deconvolutional layers followed by a Self-Attention module, thereby enhancing the model's capability to generate realistic and coherent time series data.

Convolutional Neural Networks (CNNs)

CNNs are well-suited for detecting localized patterns and extracting spatial features from time series data. By applying convolutional filters along the temporal dimension, CNNs can capture short-term trends and volatility clusters, which are crucial for modeling the non-stationary characteristics of financial markets. Our generator architecture includes deconvolutional layers that progressively upsample the latent representation, allowing the model to generate detailed and realistic time series data. The effectiveness of CNNs in time series forecasting and anomaly detection has been demonstrated in various studies, including LeCun, Bengio, and Hinton (2015). Moreover, a growing body of literature has shown that CNNs excel in quantitative finance applications. For instance, Dixon (2017) illustrated their utility in market trend forecasting, while Liu and Wu (2023) and Zhang, Zohren, and Roberts (2019) demonstrated their capabilities in option pricing, risk management, volatility prediction, and anomaly detection in high-frequency trading data. These studies underscore the versatility and robustness of CNN architectures in capturing complex, localized patterns within financial datasets.

Long Short-Term Memory (LSTM) Networks

LSTM networks are designed to capture long-term dependencies in sequential data, making them ideal for modeling the temporal evolution of financial time series. The gating mechanisms in LSTMs allow the network to retain relevant information over extended periods, addressing issues such as vanishing gradients. Our LSTM-based generator includes fully connected layers to transform the latent space into a suitable input for the LSTM layers, followed by an output layer that generates the final time series. The use of LSTMs in financial modeling has been extensively studied, as highlighted by Hochreiter and Schmidhuber (1997).

While LSTMs have historically been a cornerstone in Natural Language Processing (NLP) for tasks such as language modeling and machine translation (Fischer and Krauss (2018) and Sutskever, Vinyals, and Le (2014)), the emergence of attention-based mechanisms—introduced by Bahdanau, Cho, and Bengio (2016) and further advanced by Vaswani et al. (2023)—has largely supplanted them in that field.

In contrast, within quantitative finance, LSTMs remain a robust tool. Their ability to model sequential dependencies has proven valuable in a range of applications, including market trend forecasting (Fischer and Krauss (2018)), risk management (Bao, Yue, and Rao (2017)), and volatility prediction, where capturing temporal nuances is critical.

Integration within the GAN Framework

In our proposed model, these enhanced architectural components are integrated within the generator network while the discriminator maintains a conventional architecture. The overall training procedure remains consistent with the classical GAN framework, with the generator updates now encompassing additional hyperparameters related to the attention heads, convolutional filter sizes, and LSTM layer configurations.

The evaluation provides insights into how these components improve the generator's ability to model the intricate, nonlinear, and temporal structures present in real financial time series.

4 Results and Evaluations

As can be seen in the appendix, the CNN implementation performs better than the other two models. Visually, the log returns of the generated data look very similar to the log returns of the historical data. We can note, however, the model does not adequately capture the market dynamics for JP Morgan. The results tend to not be very aligned with the true observed distribution of returns for this stock.

Analyzing the results from a more statitical perspective, like the Jensen-Shannon divergence, the model based on the convolutional layers shows best results. Again, Apple and the MSCI ACWI Index yield better results than JPM. Looking at the Jensen-Shannon Divergence for CNN (JSD, min. of 0 and max. of $ln(2) = 0.693$) we get a score in the order of 10^{-1} which is very good. JPM on the otherhand is quite high with a score of 0.251.

The generated distribution for JPM has considerably fatter tails, especially a fatter left tail which would explain why all the generated time series plotted for JPM have a drastic negative spike in one concentrated area. This may have been caused by a large drawdown during a short time period in the historical data that the model overemphasized during training, or perhaps a data error. The LSTM model performs worst according to JSD and produces too narrow of a distribution. This may indicate that there was some sort of structural mistake in applying that architecture to our data. Perhaps, after tuning the architecture more appropriately for our data and retraining, we would achieve better results. The attention model manages to produce better distributions than the LSTM, however, the distributions are also not strongly aligned with historical data. Specifically for ACWI, the Attention architecture does quite okay (visually) with a JSD of 0.222. Overall, both visually and numerically, the CNN GAN strongly outperforms the other architectures used.

It is also interesting to look at autocorrelation for our time series which is another important metric of market dynamics. In our historical data, JPM and ACWI showed evidence of autocorrelation, while AAPL did not. This can be seen in the Ljung-Box tests in the tables in the appendix. Our generated data showed autocorrelation for all 3 tickers in all 3 models. This means that it either correctly generates autocorrelated data for 2/3 tickers, or, the architecture and methodology potentially always generates autocorrelated data.

5 Conclusion

In conclusion, our investigation into enhanced GAN architectures for synthesizing financial time series data demonstrates that incorporating convolutional layers yields superior performance in capturing the distributional characteristics of asset returns compared to LSTM and attention based models. While the CNN based model successfully replicates the historical return distributions for instruments such as AAPL and the MSCI ACWI Index, discrepancies observed for JPM highlight areas for further refinement. More work should be done to explore additional tuning of the architectures and to consider hybrid approaches that combine the strengths of attention, CNNs, and LSTMs to achieve even closer alignment with observed market data. These improvements could provide financial institutions with robust synthetic datasets for applications in backtesting, risk management, and machine learning research.

Acknowledgments

We would like to thank CodeWilling for providing the computational resources that enabled us to train and evaluate our models effectively.

References

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (May 19, 2016). *Neural Machine Translation by Jointly Learning to Align and Translate.* DOI: 10.48550/arXiv.1409.0473. arXiv: 1409.0473[cs]. URL: http://arxiv.org/abs/1409.0473 (visited on 02/25/2025).

Bao, Wei, Jun Yue, and Yulei Rao (July 14, 2017). "A deep learning framework for financial time series using stacked autoencoders and long-short term memory". In: *PLOS ONE* 12.7. Publisher: Public Library of Science, e0180944. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0180944. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0180944 (visited on 02/25/2025).

Chen, Shun and Lei Ge (Sept. 2, 2019). “Exploring the attention mechanism in LSTM-based Hong Kong stock price movement prediction”. In: *Quantitative Finance*. Publisher: Routledge. ISSN: 1469-7688. URL: `https://www.tandfonline.com/doi/abs/10.1080/14697688.2019.1622287` (visited on 02/25/2025).

Dixon, Matthew F. (Dec. 5, 2017). *A High Frequency Trade Execution Model for Supervised Learning*. DOI: `10.48550/arXiv.1710.03870`. arXiv: `1710.03870[q-fin]`. URL: `http://arxiv.org/abs/1710.03870` (visited on 02/25/2025).

Eckerli, Florian and Joerg Osterrieder (2021). “Generative Adversarial Networks in finance: an overview”. eng. In.

Fischer, Thomas and Christopher Krauss (Oct. 16, 2018). “Deep learning with long short-term memory networks for financial market predictions”. In: *European Journal of Operational Research* 270.2, pp. 654–669. ISSN: 0377-2217. DOI: `10.1016/j.ejor.2017.11.054`. URL: `https://www.sciencedirect.com/science/article/pii/S0377221717310652` (visited on 02/25/2025).

Goodfellow, Ian J. et al. (June 10, 2014). *Generative Adversarial Networks*. DOI: `10.48550/arXiv.1406.2661`. arXiv: `1406.2661[stat]`. URL: `http://arxiv.org/abs/1406.2661` (visited on 02/25/2025).

Hochreiter, Sepp and Jürgen Schmidhuber (Nov. 15, 1997). “Long Short-Term Memory”. In: *Neural Computation* 9.8, pp. 1735–1780. ISSN: 0899-7667. DOI: `10.1162/neco.1997.9.8.1735`. URL: `https://doi.org/10.1162/neco.1997.9.8.1735` (visited on 02/25/2025).

Iglesias, Guillermo, Edgar Talavera, and Alberto Díaz-Álvarez (May 2023). “A survey on GANs for computer vision: Recent research, analysis and taxonomy”. In: *Computer Science Review* 48, p. 100553. ISSN: 15740137. DOI: `10.1016/j.cosrev.2023.100553`. arXiv: `2203.11242[cs]`. URL: `http://arxiv.org/abs/2203.11242` (visited on 02/25/2025).

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (May 2015). “Deep learning”. In: *Nature* 521.7553. Publisher: Nature Publishing Group, pp. 436–444. ISSN: 1476-4687. DOI: `10.1038/nature14539`. URL: `https://www.nature.com/articles/nature14539` (visited on 02/25/2025).

Liu, David and Yu Wu (Aug. 2023). “Option pricing using deep convolutional neural networks enhanced by technical indicators”. In: *2023 IEEE 9th International Conference on Cloud Computing and Intelligent Systems (CCIS)*. 2023 IEEE 9th International Conference on Cloud Computing and Intelligent Systems (CCIS). ISSN: 2376-595X, pp. 143–147. DOI: `10.1109/CCIS59572.2023.10262865`. URL: `https://ieeexplore.ieee.org/document/10262865` (visited on 02/25/2025).

Qin, Yao et al. (Aug. 14, 2017). *A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction*. DOI: `10.48550/arXiv.1704.02971`. arXiv: `1704.02971[cs]`. URL: `http://arxiv.org/abs/1704.02971` (visited on 02/25/2025).

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le (Dec. 14, 2014). *Sequence to Sequence Learning with Neural Networks*. DOI: `10.48550/arXiv.1409.3215`. arXiv: `1409.3215[cs]`. URL: `http://arxiv.org/abs/1409.3215` (visited on 02/25/2025).

Vaswani, Ashish et al. (Aug. 2, 2023). *Attention Is All You Need*. DOI: `10.48550/arXiv.1706.03762`. arXiv: `1706.03762[cs]`. URL: `http://arxiv.org/abs/1706.03762` (visited on 02/25/2025).

Zhang, Zihao, Stefan Zohren, and Stephen Roberts (June 1, 2019). "DeepLOB: Deep Convolutional Neural Networks for Limit Order Books". In: *IEEE Transactions on Signal Processing* 67.11, pp. 3001–3012. ISSN: 1053-587X, 1941-0476. DOI: `10.1109/TSP.2019.2907260`. arXiv: `1808.03668[q-fin]`. URL: `http://arxiv.org/abs/1808.03668` (visited on 02/25/2025).

A Appendix

Metric	**Original**	**Generated**
Mean	0.000128	-0.000002
Std	0.040645	0.214407
Skewness	-25.111799	0.255764
Kurtosis	1039.042700	46.728140
Ljung-Box t-statistic	10.673040	272346.309774
Ljung-Box p-value	0.383552	0.000000

Table 1: AAPL Moments Comparison for the Attention-Based GAN. Jensen–Shannon Divergence: 0.580034

Metric	**Original**	**Generated**
Mean	0.000279	0.000001
Std	0.012906	0.079548
Skewness	-0.556542	-0.786310
Kurtosis	11.866263	135.567522
Ljung-Box t-statistic	55.307795	1234.989861
Ljung-Box p-value	0.000000	0.000000

Table 2: ACWI Moments Comparison for the Attention-Based GAN. Jensen–Shannon Divergence: 0.221720

Metric	**Original**	**Generated**
Mean	0.000189	-0.000002
Std	0.104471	0.198215
Skewness	-0.021890	0.098549
Kurtosis	106.470019	30.823050
Ljung-Box t-statistic	1442.181508	247959.786055
Ljung-Box p-value	0.000000	0.000000

Table 3: JPM Moments Comparison for the Attention-Based GAN. Jensen–Shannon Divergence: 0.504067

Metric	Original	Generated
Mean	0.000128	0.000003
Std	0.040645	0.180111
Skewness	-25.111799	-0.640448
Kurtosis	1039.042700	158.044933
Ljung-Box t-statistic	10.673040	347.086390
Ljung-Box p-value	0.383552	0.000000

Table 4: AAPL Moments Comparison for the CNN-Based GAN. Jensen–Shannon Divergence: 0.060796

Metric	Original	Generated
Mean	0.000279	-0.000001
Std	0.012906	0.082028
Skewness	-0.556542	-0.880490
Kurtosis	11.866263	143.158931
Ljung-Box t-statistic	55.307795	264.902105
Ljung-Box p-value	0.000000	0.000000

Table 5: ACWI Moments Comparison for the CNN-Based GAN. Jensen–Shannon Divergence: 0.065261

Metric	Original	Generated
Mean	0.000189	-0.000001
Std	0.104471	0.174060
Skewness	-0.021890	1.247102
Kurtosis	106.470019	60.163179
Ljung-Box t-statistic	1442.181508	64169.776953
Ljung-Box p-value	0.000000	0.000000

Table 6: JPM Moments Comparison for the CNN-Based GAN. Jensen–Shannon Divergence: 0.251107

Metric	Original	Generated
Mean	0.000128	-0.000005
Std	0.040645	0.168739
Skewness	-25.111799	0.647168
Kurtosis	1039.042700	150.256361
Ljung-Box t-statistic	10.673040	1412.511109
Ljung-Box p-value	0.383552	0.000000

Table 7: AAPL Moments Comparison for the LSTM-Based GAN. Jensen–Shannon Divergence: 0.608428

Metric	Original	Generated
Mean	0.000279	-0.000000
Std	0.012906	0.084201
Skewness	-0.556542	2.924553
Kurtosis	11.866263	80.666600
Ljung-Box t-statistic	55.307795	3032.228573
Ljung-Box p-value	0.000000	0.000000

Table 8: ACWI Moments Comparison for the LSTM-Based GAN. Jensen–Shannon Divergence: 0.601836

Metric	Original	Generated
Mean	0.000189	0.000002
Std	0.104471	0.120360
Skewness	-0.021890	-0.977749
Kurtosis	106.470019	132.980503
Ljung-Box t-statistic	1442.181508	6376.462484
Ljung-Box p-value	0.000000	0.000000

Table 9: JPM Moments Comparison for the LSTM-Based GAN. Jensen–Shannon Divergence: 0.596089

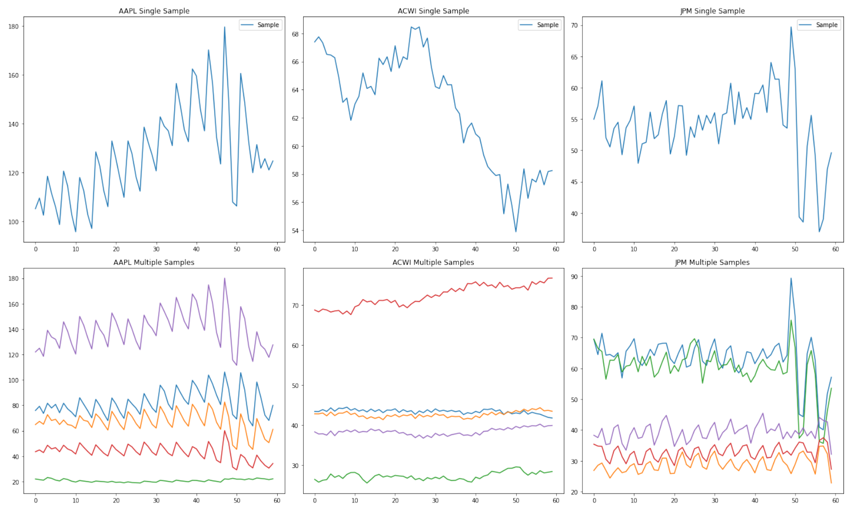

Figure 1: Comparison of Real vs. Generated Time Series for AAPL, ACWI and JPM using Attention-Based GAN.

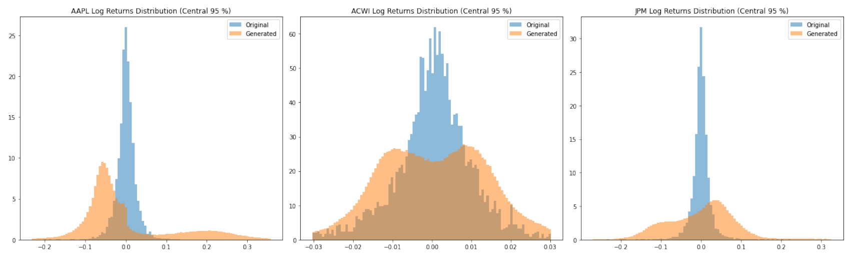

Figure 2: Comparison of Real vs. Generated Log-Return Distributions for AAPL, ACWI and JPM using Attention-Based GAN.

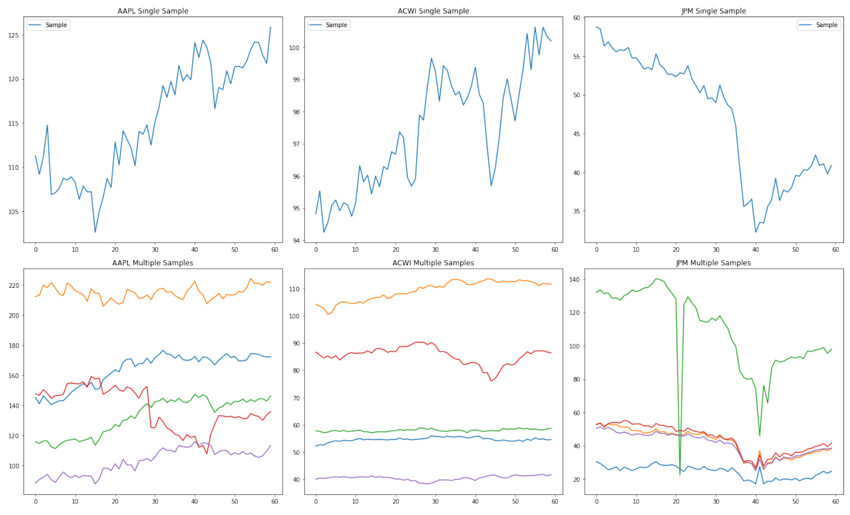

Figure 3: Comparison of Real vs. Generated Time Series for AAPL, ACWI and JPM using CNN-Based GAN.

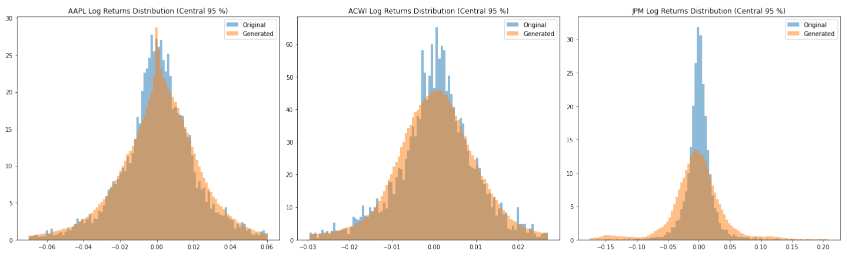

Figure 4: Comparison of Real vs. Generated Log-Return Distributions for AAPL, ACWI and JPM using CNN-Based GAN.

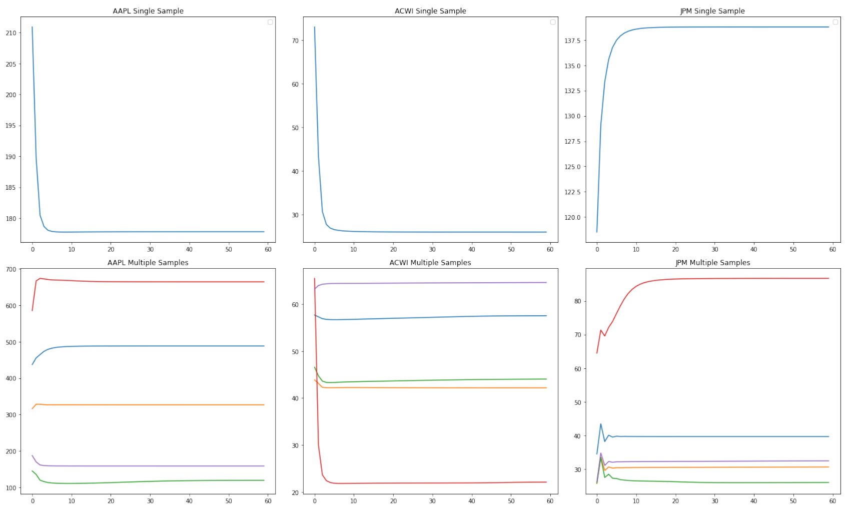

Figure 5: Comparison of Real vs. Generated Time Series for AAPL, ACWI and JPM using LSTM-Based GAN.

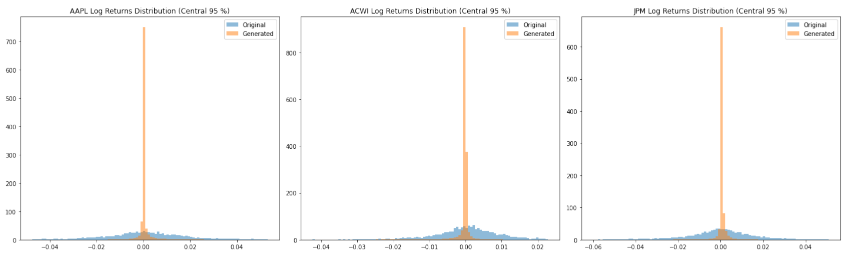

Figure 6: Comparison of Real vs. Generated Log-Return Distributions for AAPL, ACWI and JPM using LSTM-Based GAN.