Generative Adversarial Networks for Synthetic Financial Time-Series Generation

Sankalp Yadav

July 6, 2025

Abstract

The efficacy of quantitative financial models, particularly those for algorithmic trading and risk management, is fundamentally dependent on the volume and quality of available historical data. Backtesting strategies on limited historical data can lead to overfitting and poor out-of-sample performance. This paper details the implementation of a sophisticated generative model, TimeGAN, to create high-fidelity, realistic synthetic financial time-series data. I frame the problem as learning the underlying data generating process of real stock prices. The goal is to produce synthetic data that preserves the key statistical and temporal characteristics of the original series, such that it can serve as a viable augmentation for historical datasets. I implement the TimeGAN architecture, which uniquely combines adversarial training in a latent space with a supervised loss to capture temporal dynamics. The model is trained on historical daily price data from a diverse portfolio of US equities. I detail the iterative tuning process, including the crucial implementation of the Wasserstein GAN with Gradient Penalty (WGAN-GP) framework to overcome common training instabilities like mode collapse, and conclude by presenting a qualitative analysis of the generated data. This research serves as a comprehensive guide to the practical application of generative adversarial networks for robust financial data synthesis.

1 Introduction

Quantitative strategies are rigorously tested against historical market data before deployment. This process, known as **backtesting**, is the cornerstone of model validation in finance. However, this reliance on historical data presents a significant challenge: the available data is finite and represents only one possible realization of market behavior. Models that are highly optimized on this single historical path are prone to **overfitting**, exhibiting excellent performance in backtests but failing in live market conditions.

Furthermore, certain market regimes or "black swan" events are rare, leading to a scarcity of data for training models to be robust under such conditions. The ability to generate new, realistic financial data that preserves the essential characteristics of historical data is therefore of immense value. Such synthetic data can be used to augment historical datasets, allowing for more robust backtesting, stress testing, and the training of data-hungry machine learning models.

Generative Adversarial Networks (GANs), first introduced by **Goodfellow et al. (2014)**, provide a powerful, model-free framework for learning complex data distributions. A GAN consists of two neural networks, a **Generator** and a **Discriminator**, which are trained in a competitive, zero-sum game. The Generator learns to produce synthetic samples, while the Discriminator learns to distinguish these synthetic samples from real ones. Through this adversarial process, the Generator becomes progressively better at creating realistic data.

While standard GANs excel at generating independent samples like images, they are not inherently suited for sequential data like time series, which possess strong temporal correlations. The **TimeGAN** architecture, proposed by **Yoon et al. (2019)**, specifically addresses this by incorporating an autoencoder and a supervised loss to explicitly learn the temporal dynamics of the data within a latent space.

My contribution is to provide a detailed, step-by-step implementation and analysis of the TimeGAN framework for generating synthetic stock price data. I document the entire process, from data preprocessing to addressing critical training challenges like mode collapse by implementing advanced techniques such as the Wasserstein GAN with Gradient Penalty (WGAN-GP). The final result is a trained model capable of producing high-quality synthetic data that visually captures the characteristics of real stock price movements.

2 Data and Preprocessing

The model was trained on a diverse set of historical daily stock prices to ensure it learns a wide range of market dynamics.

1. **Data Source**: Daily price data for 12 highly liquid US equities from a variety of sectors (Technology, Healthcare, Consumer Discretionary) was downloaded using the 'yfinance' library. The selected tickers include AAPL, NVDA, MSFT, JNJ, PFE, AMZN, and TSLA.

2. **Time Frame**: The data covers a 10-year period from January 1, 2015, to December 31, 2024, encompassing multiple market regimes.

3. **Preprocessing**: To prepare the data for the neural network, two key preprocessing steps were performed:

   - **Normalization**: The 'Adjusted Close' price for each stock was scaled to the range [0, 1] using a 'MinMaxScaler'. This is essential for the stability of neural network training.
   - **Sequencing**: The continuous time series was transformed into a dataset of overlapping sequences of a fixed length (e.g., 24 timesteps). This creates the '(samples, timesteps, features)' structure required by recurrent neural networks.

The processed data for each ticker was saved locally, allowing for efficient loading during the training phase.

3 The Generative Model as a Minimax Game

The theoretical foundation of a GAN is a two-player minimax game between the Generator (G) and the Discriminator (D). The goal is to find a Nash equilibrium where the generator produces data indistinguishable from real data, and the discriminator is unable to tell the difference.

The Generator, $G(z)$, maps a random noise vector z from a prior distribution $p_z(z)$ to the data space. The Discriminator, $D(x)$, outputs the probability that a sample x is real rather than synthetic. The objective function, $V(D, G)$, is defined as:

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

The discriminator D tries to maximize this objective by correctly identifying real samples ($D(x) \to 1$) and fake samples ($D(G(z)) \to 0$). The generator G tries to minimize the objective by producing samples that fool the discriminator ($D(G(z)) \to 1$).

4 Methodology: TimeGAN Architecture

To adapt the GAN framework for time-series data, I implemented the TimeGAN architecture, which has four key network components and a three-part loss function.

4.1 Network Components

1. **Encoder**: A recurrent neural network (RNN), specifically a GRU, that maps real time-series data X into a latent sequence representation H.

2. **Recovery**: An RNN that maps the latent sequence H back to the original data space, creating a reconstructed sequence $\tilde{X}$. The Encoder and Recovery together form an autoencoder.

3. **Generator**: An RNN that takes a sequence of random vectors Z and generates a synthetic latent sequence $\hat{H}$.

4. **Discriminator (Critic)**: An RNN that takes a latent sequence (either real, H, or synthetic, $\hat{H}$) and tries to distinguish between them.

4.2 Loss Functions

The model is trained by jointly optimizing three distinct loss functions:

1. **Reconstruction Loss**: The Mean Squared Error (MSE) between the original data X and the autoencoder's reconstruction $\tilde{X}$. This ensures the latent space preserves the necessary information.

$$\mathcal{L}_R = \mathbb{E}[||X - \tilde{X}||^2] \tag{2}$$

2. **Supervised Loss**: The MSE between the real latent sequence H and the "supervised" generated sequence $\hat{H}_S$ (where the generator is given H as input). This explicitly trains the generator to capture the temporal dynamics of the data.

$$\mathcal{L}_S = \mathbb{E}[||H - \hat{H}_S||^2] \tag{3}$$

3. **Adversarial Loss**: The loss from the minimax game between the Generator and the Discriminator in the latent space. To stabilize training, the standard GAN loss was replaced with the more robust **Wasserstein Loss with Gradient Penalty (WGAN-GP)**.

5 Experimental Design and Results

1. **Implementation**: All models were built using the TensorFlow and Keras libraries. The training process was encapsulated in a modular 'TimeGANTrainer' class.
2. **Training Phase**: The model was trained on a single stock's data (AAPL) for 500 epochs. An iterative tuning process was essential. Initial training runs suffered from **mode collapse**, where the generator produced only a limited variety of smooth, unrealistic samples.
3. **Overcoming Instability**: To achieve stable training and high-quality results, the following key modifications were implemented:
   - Regularization: A 'GaussianNoise' layer was added to the discriminator's input to prevent it from overpowering the generator.
   - WGAN-GP: The standard Binary Cross-Entropy loss was replaced with the Wasserstein loss and a gradient penalty. This was the most critical change, as it provided a stable gradient for the generator to learn from.
   - Hyperparameter Tuning: Learning rates and training ratios for the critic and generator were carefully tuned to maintain a competitive balance.
4. **Qualitative Evaluation**: The quality of the final model was assessed by visually comparing plots of real data sequences against synthetic sequences generated by the trained model. The final results demonstrated that the model was able to generate diverse samples that successfully captured the characteristic volatility, noise, and short-term trends of the real stock price data.

6 Conclusion

This paper has successfully detailed the end-to-end process of building, training, and tuning a TimeGAN model for generating synthetic financial time-series data. I demonstrated that a standard implementation is prone to instability but that these issues can be overcome by applying advanced techniques such as discriminator regularization and the WGAN-GP loss framework. The final model proved capable of producing high-fidelity synthetic data that is qualitatively similar to real stock price movements. This validates the use of generative adversarial networks as a powerful tool for data augmentation in quantitative finance, providing a path to more robust model backtesting and development.

7 Future Work

This research can be extended in several promising directions:

- **Quantitative Evaluation**: Implementing rigorous statistical tests (e.g., comparing distributions of returns, autocorrelation) and the "Train-on-Synthetic, Test-on-Real" (TSTR) methodology to quantitatively measure the quality of the synthetic data.
- **Multivariate Models**: Extending the framework to generate data for a portfolio of assets simultaneously, thereby learning the complex correlation structure between them.
- **Alternative Architectures**: Implementing and comparing the performance of other generative models, such as Transformers, which have shown great promise in sequence modeling tasks.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672-2680.

[2] J. Yoon, D. Jarrett, and M. van der Schaar, "Time-series Generative Adversarial Networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 5867-5877.

[3] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved Training of Wasserstein GANs," in *Advances in Neural Information Processing Systems*, 2017, pp. 5767-5777.