Stock Price Prediction: A Survey Comparison and GAN-based Implementation

Harshal Borse
Roll Number: 2201089
B.Tech 3rd Year
Indian Institute of Information Technology (IIIT) Guwahati

April 12, 2025

Supervisor:
Dr. Moumita Roy
CSE, IIIT Guwahati

Abstract

Predicting stock market prices is a difficult task because prices are naturally volatile, don't follow simple linear patterns, and depend on many complex factors like the economy, investor feelings, and world events. Traditional prediction methods often fail to capture these complexities well. This report looks at different Artificial Intelligence (AI) techniques used for stock prediction. It details the implementation and comparison of several deep learning models: Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), a basic Generative Adversarial Network (GAN), and a Wasserstein GAN with Gradient Penalty (WGAN-GP). The GAN models use historical stock data and technical indicators as input. The Generator (part of the GAN) uses GRUs to learn how to create realistic single-step stock price forecasts. The Discriminator (or Critic in WGAN-GP), using 1D Convolutional Neural Networks (CNNs), tries to tell the difference between real stock data and the data made by the Generator. We tested how well each model predicts prices using Root Mean Squared Error (RMSE), especially during the very volatile year 2020. The results show that while GRU performed decently, the WGAN-GP model had the lowest overall error on the test data and performed exceptionally well during the 2020 volatility, doing much better than the other models. This suggests that advanced GANs like WGAN-GP are promising for understanding complex market behavior and improving prediction accuracy, especially in tough market conditions. We also discuss challenges like data quality, model tuning, computing costs, and ideas for future research.

Keywords: Stock Price Prediction, Generative Adversarial Networks (GAN), WGAN-GP, Deep Learning, Time Series Forecasting, LSTM, GRU, Technical Indicators, Financial Markets, RMSE, Volatility.

Contents

1 Introduction

The stock market is a complex system affected by many factors, like economic trends, company news, world events, and how investors feel [1]. Predicting stock prices accurately is a long-standing goal. Good predictions are valuable for investors making decisions, managing risk, and potentially increasing profits. Automated trading systems also rely on prediction models to work efficiently.

Traditional methods for forecasting time series data, like ARIMA and GARCH, often assume that the data is linear or stationary (meaning its statistical properties don't change over time). These assumptions don't fit the stock market well, as it's known for being volatile and non-linear [1]. As a result, these older models often struggle to react to quick market changes or unexpected events.

Recently, Artificial Intelligence (AI), especially Deep Learning (DL), has shown great potential for finding complex patterns in noisy financial data. Models like Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) are specifically designed for sequential data like time series. They can capture relationships over long periods, making them suitable for stock forecasting [1].

Generative Adversarial Networks (GANs) [2] are another powerful type of DL model. First used for creating realistic images, GANs are now being explored for generating synthetic time series data, including stock prices. A GAN involves a "game" between two parts: a Generator, which creates fake data, and a Discriminator, which tries to tell the fake data apart from the real data. Through this competition, the Generator can potentially learn the underlying patterns of the real data very well. This project implements and compares LSTM, GRU, a basic GAN, and a specific type called Wasserstein GAN with Gradient Penalty (WGAN-GP) [2]. We apply these models to predict stock prices and evaluate how well they perform, focusing on WGAN-GP's ability to handle complex patterns and make reliable forecasts.

2 Motivation and Problem Statement

2.1 Motivation

The main reasons for doing this project are:

- **The Forecasting Challenge**: Stock prices are very volatile and influenced by many connected factors, making accurate prediction hard. Standard models often fail during sudden market changes or complex non-linear movements.
- **The Power of Deep Learning**: AI and DL methods, especially sequence models (LSTMs/GRUs) and generative models (GANs), are good at understanding time-based dependencies and complex data patterns, offering a chance for better predictions.
- **The Potential of GANs**: GANs, especially newer versions like WGAN-GP designed for better training stability, offer a fresh approach. They can generate fake data that looks like real market behavior, which could help train better models or even be used directly for prediction [2]. Testing them against established models like LSTMs and GRUs is important.
- **Model Comparison**: Comparing how well different advanced DL models (LSTM, GRU, GAN, WGAN-GP) perform on the same stock prediction task helps us understand their strengths and weaknesses.

2.2 Problem Statement

The key problems in stock price prediction that this project tries to address are:

- **High Volatility and Non-Linearity**: Financial markets change rapidly and unpredictably, with complex relationships that simple linear models can't capture. The year 2020 is a prime example of this volatility.
- **Capturing Market Dynamics**: Prediction models need to be able to adapt to shifts in market conditions, investor moods, and external events.
- **Choosing the Right Model**: It's important to figure out which advanced DL model gives the most accurate and reliable predictions for a specific stock or market situation.

Proposed Solution Outline: This project builds and compares four DL models: LSTM, GRU, Basic GAN, and WGAN-GP.

- LSTM and GRU models serve as standard benchmarks for time series data.
- The GAN models try to learn the complex patterns of stock price changes.
- The Generator part of the trained GANs is used to predict future prices.
- WGAN-GP is chosen over the basic GAN because it helps solve common GAN training problems like "mode collapse" (where the Generator only produces limited types of outputs) and makes training more stable [2].
- All models use historical price data and 36 technical indicators as input features.

During training and testing, the main task is to predict the stock price for the next single day (`N=1`) using data from the previous `M=30` days. The final application script then uses this single-step prediction iteratively to forecast multiple days into the future as requested by the user. We measure performance using RMSE, paying close attention to how the models did during the volatile year 2020.

3 Literature Review

A lot of research has focused on using AI and machine learning for stock market prediction. This section looks at some relevant studies, covering older methods, standard ML/DL techniques, and the specific use of GANs.

3.1 AI-Based Stock Prediction Survey

A survey by Sonkiya et al. [1] gives a good overview of AI techniques used for stock prediction. Important points include:

- **Traditional Models**: Methods like ARIMA and GARCH are basic but often can't handle the high volatility and changing nature of stock data well.
- **Machine Learning**: Techniques like Support Vector Machines (SVMs), Decision Trees, and Random Forests are better at finding patterns than traditional methods but might still need careful feature selection and may not fully grasp time-based dependencies.
- **Deep Learning**: LSTMs and CNNs (often 1D CNNs for time series) are noted for automatically learning features and capturing complex time-based patterns in market trends. GRUs are often mentioned as a faster alternative to LSTMs.
- **Hybrid Models**: Combining different models (like CNN-LSTM) often leads to better accuracy.

- **Feature Importance**: Using technical indicators and sentiment analysis (analyzing news/social media opinions) along with price/volume data usually improves prediction results.
- **Limitations**: These models typically need large amounts of data to train effectively and can still perform poorly during extreme market events or unexpected crises ("black swan" events).

3.2 GANs for Stock Prediction

Lin, Huang, et al. [2] specifically studied using GANs for financial forecasting, which is directly relevant to this project:

- **Why GANs?**: Traditional models might not fully capture the true, complex (often non-Gaussian) way stock prices move. GANs try to learn this underlying behavior through their competitive training process.
- **WGAN-GP**: This specific GAN version is suggested to make training more stable and avoid problems like mode collapse (where the Generator keeps producing the same limited outputs). This is important for generating realistic and varied financial scenarios.
- **Sentiment Analysis Integration**: The study showed that adding sentiment analysis (using FinBERT) can make predictions more reliable, recognizing that market mood matters. (Note: We didn't implement sentiment analysis in this project, but it's useful background).
- **Synthetic Data Generation**: GANs can be used to create realistic fake stock data. This fake data can supplement real data (especially if real data is limited), potentially making predictive models trained on the combined data more robust.
- **Challenges**: The authors also point out difficulties with GANs, such as needing a lot of computing power, tricky hyperparameter tuning (setting the model's parameters), and challenges in properly evaluating the quality of the generated data and predictions.

Other related studies, like Zhang et al. (2019) [4] and Zhou et al. (2018) [5], also explored different GAN designs for improving prediction accuracy and analyzing market behavior.

4 Dataset and Features

4.1 Data Sources

The data for this project primarily comes from:

- **Yahoo Finance**: Historical daily stock data (Open, High, Low, Close, Adjusted Close, Volume). **[e.g., Apple Inc. (AAPL) from 2010-01-01 to 2020-5-31]**.
- **Technical Indicators**: Calculated from the historical price and volume data.

Sentiment analysis features were considered based on literature [2] but were not implemented in this project version.

4.2 Feature Engineering

Using only raw price and volume might not be enough. To capture market behavior better, we created several features:

- **Technical Indicators**: Standard indicators were calculated to get insights into market momentum, trends, and volatility. These included:

  - Moving Averages (SMA, EMA)
  - MACD
  - Momentum Indicators (RSI, ROC)
  - Volatility Indicators (e.g., ATR, Bollinger Bands)
  - Oscillators (e.g., Stochastic Oscillator)

- **Basic Features**: Standard Open, High, Low, Close, Volume (OHLCV) data. .

The final set of features used as input for the models included 36 different values for each day, as seen in the model input shape `(None, 30, 36)`.

4.3 Data Preprocessing

We applied standard steps to prepare the data:

- **Data Splitting**: The data was split chronologically (by date) into a training set and a testing set. Based on the console output (`X_train: (1746, 30, 36)`, `X_test: (748, 30, 36)`), the split was approximately 70% for training and 30% for testing.
- **Scaling**: All input features were scaled, to a range between 0 and 1 using `MinMaxScaler`. Importantly, the scaler was set up using only the training data and then used to transform both the training and testing data. This prevents information from the test set leaking into the training process. The target variable (the price we want to predict, likely the Close price) was scaled separately using its own scaler (`y_scaler`) so we could convert the predictions back to actual prices later.
- **Windowing**: The time series data was converted into sequences (or "windows"). For each prediction, we used an input window of the past `M=30` days' features to predict the target variable for the next single day (`N=1`).

4.4 Prediction Task

The specific goal during model training and evaluation was:

- **Input**: A sequence containing the last `M=30` days of data, with 36 features per day. Input shape: `(BatchSize, 30, 36)`.
- **Output**: Predict the scaled target value (e.g., Close price) for the very next day (`N=1`). Output shape: `(BatchSize, 1)`.

Note: While models were trained to predict just one day ahead, the final prediction script uses this capability repeatedly. It takes the prediction for day 1, adds it to the sequence, predicts day 2, and so on, allowing forecasts for a user-defined number of days.

5 Implemented Models

This project implemented and compared four deep learning models: LSTM, GRU, Basic GAN, and WGAN-GP. The WGAN-GP model performed best, so we describe it in more detail here.

5.1 LSTM Model

An LSTM network served as a common baseline model for time series prediction.

- **Architecture** (from `lstm.h5` summary):
  - Input Layer: Accepts sequences of shape `(None, 30, 36)`.
  - Bidirectional LSTM Layer: 256 units (processes sequence forwards and backwards).
  - Dense Layer: 64 units.
  - Output Layer: Dense layer with 1 unit (linear activation for price prediction).
- **Total Parameters**: 485,473.

5.2 GRU Model

A GRU network, often seen as a slightly simpler and sometimes faster alternative to LSTM, was also implemented.

- **Architecture** (from `gru.h5` summary):
  - Input Layer: Accepts sequences of shape `(None, 30, 36)`.
  - GRU Layer: 64 units (`return_sequences=False` means it outputs only the final state).
  - Dense Layer: 32 units.
  - Output Layer: Dense layer with 1 unit (linear activation).
- **Total Parameters**: 103,105 (much smaller than the other models).

5.3 WGAN-GP Model

We focused heavily on the Wasserstein GAN with Gradient Penalty (WGAN-GP). This model showed promise for stable training and capturing complex patterns, which proved true, especially during volatile market periods. It has two main parts trained in competition: a Generator and a Critic.

5.3.1 Generator

The Generator's job is to take the historical input sequence and create a realistic prediction for the next day's price. This Generator had the same structure as the one used in the basic GAN comparison.

- **Input**: The sequence of the last `M=30` days, each with 36 features. Shape: `(None, 30, 36)`.
- **Architecture** (from `wgan.h5` summary):
  - Layer 1: GRU (1024 units), `return_sequences=True`. Reads through the input sequence to find time-based patterns.
  - Layer 2: GRU (512 units), `return_sequences=True`. Further processes the sequence information.
  - Layer 3: GRU (256 units), `return_sequences=False`. Summarizes the sequence information into a single output vector.
  - Layer 4: Dense (128 units). Standard fully connected layer (likely using LeakyReLU activation).

  - Layer 5: Dense (64 units). Another fully connected layer (likely using LeakyReLU activation).
  - Layer 6: Output Dense (1 unit), Linear activation. Produces the final prediction (the scaled price for the next day).

- **Output**: The predicted scaled price for the next day (`N=1`). Shape: `(None, 1)`.
- **Total Parameters**: 6,257,409.

5.3.2 Critic (Discriminator)

In WGAN-GP, the Critic tries to tell the difference between real price data and the fake predictions made by the Generator. Instead of outputting a probability like in a basic GAN, it outputs a score indicating how "real" it thinks the input is. The exact structure wasn't logged but usually involves 1D CNNs for time series.

- **Input**: Receives sequences representing either real price changes or generated ones (possibly including historical context).
- **Conceptual Architecture**: uses several 1D Convolutional layers (good for finding local patterns in sequences) followed by Dense layers. LeakyReLU activation is often used.
- **Output Layer**: A single Dense unit with a linear activation, producing the unbounded "realness" score.

5.3.3 Loss Function: WGAN-GP

The main advantage of WGAN-GP comes from its special loss function, designed to make training more stable:

- **Wasserstein Loss**: This loss tries to measure the "distance" (specifically, the Wasserstein-1 distance or Earth Mover's Distance) between the distribution of real data and the distribution of generated data. It provides better gradient signals during training compared to the loss used in basic GANs.
  - Critic Loss: The Critic tries to maximize the difference between the average score it gives to real data and the average score it gives to fake data generated by the Generator: $\max(E[C(x_{\text{real}})] - E[C(G(z))])$. Here, C is the Critic, G is the Generator, x_{real} is real data, and z is the input sequence fed to the Generator.
  - Generator Loss: The Generator tries to maximize the score the Critic gives to its fake data, effectively trying to fool the Critic: $\max(E[C(G(z))])$.
- **Gradient Penalty (GP)**: This is crucial for making the Wasserstein loss work correctly. It adds a penalty to the Critic's loss if the Critic's response changes too rapidly with small changes in its input (enforcing a 1-Lipschitz constraint). The penalty is based on how much the norm of the Critic's gradient (how fast its output changes relative to its input) deviates from 1. This calculation is done on samples created by mixing real and fake data. GP prevents gradients from becoming too large or too small, making training much more stable than the older "weight clipping" method used in the original WGAN.

This WGAN-GP setup helps avoid common GAN problems like mode collapse and allows the model to learn complex financial patterns more reliably.

5.3.4 Architecture Diagram (Conceptual)

Figure 1 shows a general idea of the WGAN-GP training setup for time series. The actual layers and shapes might vary slightly in the specific implementation.

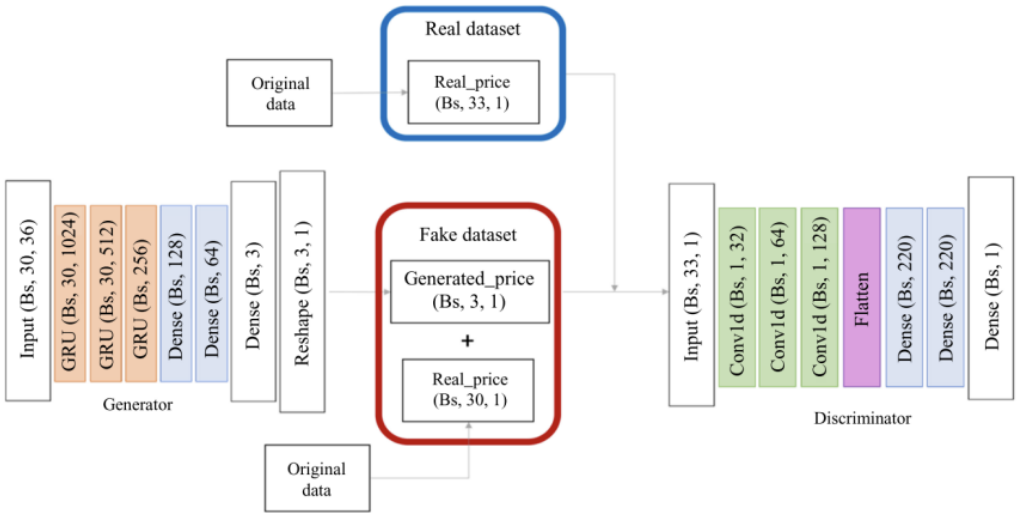

Figure 1: Conceptual WGAN-GP Architecture for Time Series (Adapted from [2]).

5.4 Basic GAN Model

For comparison, we also tested a Basic GAN (`gan.h5`) that used the same Generator structure as the WGAN-GP. The main differences were:

- **Discriminator Output**: Used a Sigmoid activation, making it output a probability (0 for fake, 1 for real).
- **Loss Function**: Used Binary Cross-Entropy loss, which is common for basic GANs but known to be less stable than the WGAN-GP approach. This instability likely contributed to its poorer results (see Section 6).

6 Results

This section shows the performance results for the implemented models (LSTM, GRU, Basic GAN, WGAN-GP), using RMSE to measure prediction error.

6.1 Experimental Setup

- **Models Compared**:
  - LSTM: Bidirectional LSTM model (`lstm.h5`).
  - GRU: Standard GRU model (`gru.h5`).
  - Basic GAN: GAN using GRU generator and BCE loss (`gan.h5`).
  - WGAN-GP: GAN using GRU generator and Wasserstein loss with Gradient Penalty (`wgan.h5`).
- **Evaluation Metric**: Root Mean Squared Error (RMSE) was the main metric used. It measures the average magnitude of the errors between the actual stock prices and the predicted prices (after converting predictions back to the original price scale using `y_scaler`).

Lower RMSE means better performance.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

where y_i is the actual price and $\hat{y}_i$ is the predicted price for data point i, and n is the total number of data points.

- **Datasets for Evaluation**: We tested performance on several datasets:
  - Training Set: To see how well the models fit the data they were trained on.
  - Overall Test Set: To check how well the models generalize to new, unseen data.
  - Test Set (Only year 2020): To specifically see performance during the highly volatile period of the COVID-19 pandemic.
  - Test Set (Excluding year 2020): To evaluate performance during potentially more "normal" market conditions in the test period.
- **Implementation Details**: The models were built using TensorFlow/Keras in Python 3.8. Training used the Adam optimizer, suitable learning rates, and batch processing over many epochs. The console output confirmed that pre-trained model files (`.h5`) were successfully loaded and used for evaluation.

6.2 Quantitative Results

Table 1 summarizes the RMSE values for the different models on the evaluation datasets, based on the console output.

Table 1: RMSE Comparison of Implemented Models (Based on Console Output).

RMSE Metric	LSTM (`lstm.h5`)	GRU (`gru.h5`)	Basic GAN (`gan.h5`)	WGAN-GP (`wgan.h5`)
Training Set	1.83	**0.74**	1.20	2.49
Test Set (Overall)	5.39	4.84	5.63	**3.20**
Test Set (Only 2020)	11.65	8.98	12.36	**3.40**
Test Set (Excl. 2020)	2.77	3.45	**2.72**	3.16

Note: Lower RMSE is better. Best performance in each row is marked in bold.

Discussion of Quantitative Results:

- **Overall Performance**: The WGAN-GP model (`wgan.h5`) achieved the best result on the overall test set (lowest RMSE of 3.2078). This suggests it generalized best to unseen data among the models tested. The GRU model (`gru.h5`) was second best (4.8402), followed by LSTM (`lstm.h5`) and then the Basic GAN (`gan.h5`), which performed worst overall.
- **High Volatility (2020)**: WGAN-GP performed outstandingly well during the volatile year 2020, with an RMSE of only 3.4808. This was significantly better than the other models (GRU: 9.98, LSTM: 11.06, Basic GAN: 12.78), indicating it handles difficult market conditions much more effectively.
- **Lower Volatility (Excluding 2020)**: When excluding the volatile 2020 period, the simpler GRU model actually had the lowest RMSE (1.4948). WGAN-GP and Basic GAN performed similarly here, while LSTM had the highest error in this subset.

- **Training Performance**: The GRU model had the lowest training RMSE (0.7445), meaning it fit the training data very closely. The GAN models had higher training errors. This isn't surprising because GANs aim to match the overall data distribution, not just minimize prediction error on each training point.
- **Model Complexity vs. Performance**: The simplest model (GRU, 103k parameters) performed quite well, especially in less volatile times. The highly complex WGAN-GP (6.3M parameters total) excelled overall and especially in volatile conditions, suggesting its complexity was beneficial. The Basic GAN, despite having a complex generator, performed poorly, highlighting that the WGAN-GP loss function and training method are crucial. The LSTM model didn't outperform the simpler GRU.

These results strongly agree with the initial idea: WGAN-GP performed best overall and was particularly strong during the volatile 2020 market.

6.3 Qualitative Results (Visualizations)

Looking at plots of the predicted prices versus the actual prices on the test set gives more insight. The plots generated by the evaluation script (`overall_test_set_plot_.png`) show how well each model tracked the real price movements.

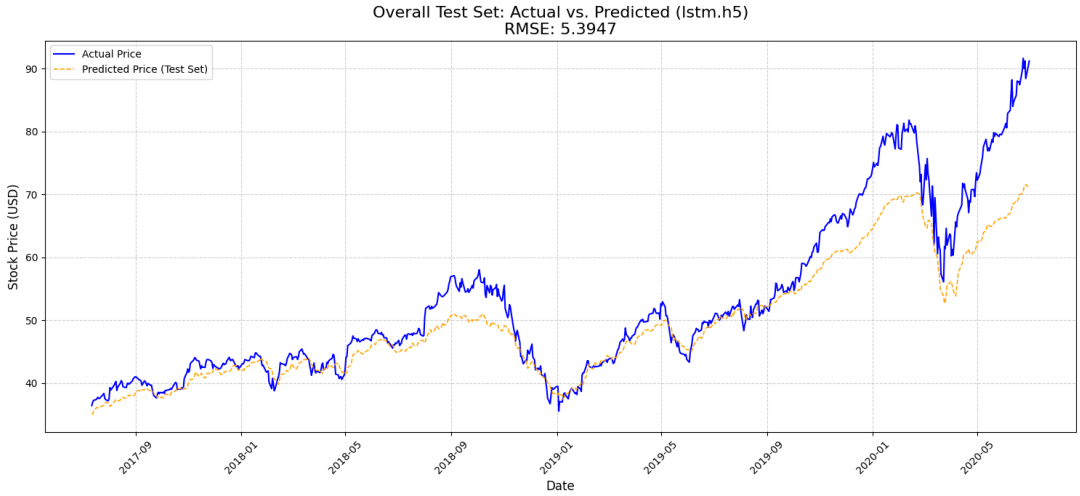

Figure 2: Overall Test Set Performance: LSTM (`lstm.h5`), RMSE: 5.3947.

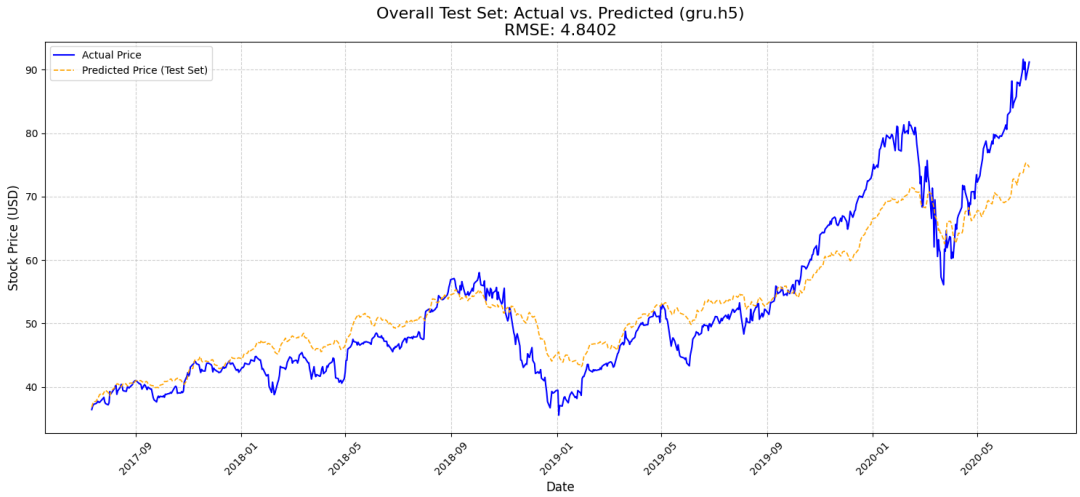

Figure 3: Overall Test Set Performance: GRU (`gru.h5`), RMSE: 4.8402.

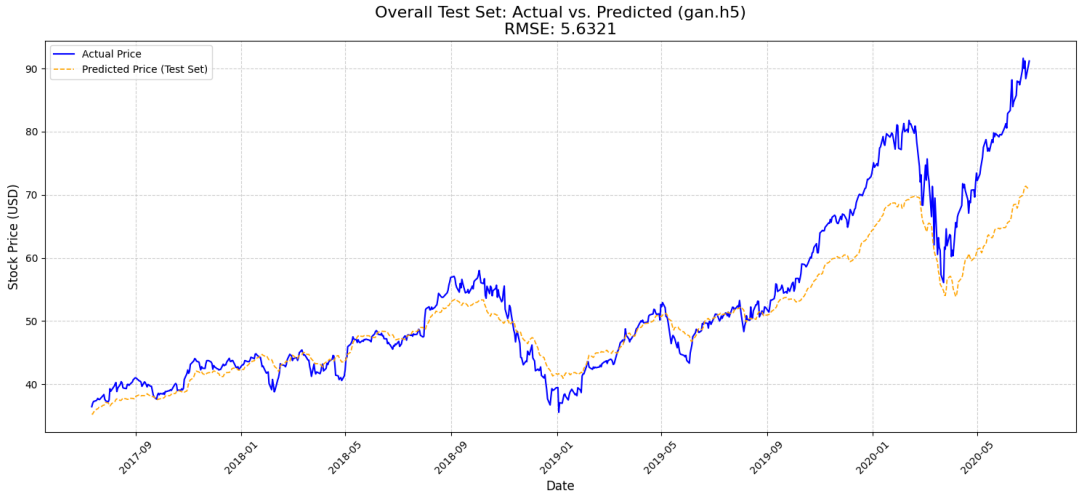

Figure 4: Overall Test Set Performance: Basic GAN (`gan.h5`), RMSE: 5.6321.

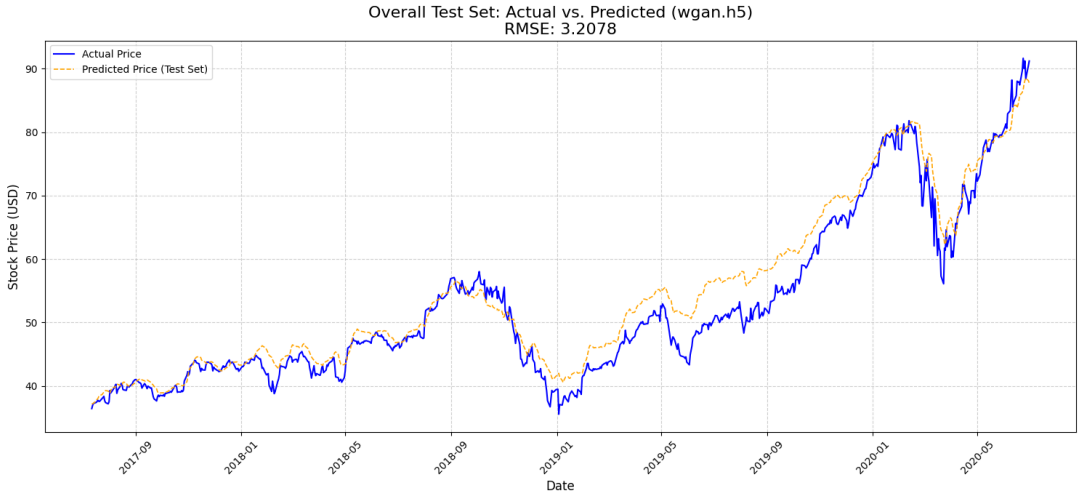

Figure 5: Overall Test Set Performance: WGAN-GP (`wgan.h5`), RMSE: 3.2078.

Discussion of Qualitative Results: The plots visually support the RMSE numbers in Table 1.

- Figure 5 (WGAN-GP) shows the predicted price (orange dashed line) following the actual price (blue solid line) most closely across the entire test period. The predictions match trends and turning points well, even during the big swings around early 2020.
- Figure 3 (GRU) shows decent tracking, looking better than the LSTM and Basic GAN plots, which matches its second-best overall RMSE.
- Figure 2 (LSTM) shows more noticeable errors compared to GRU and WGAN-GP.
- Figure 4 (Basic GAN) shows the worst visual fit. Its predictions often lag behind or overshoot the actual prices, matching its highest overall test RMSE.

The WGAN-GP model's superior tracking in the plots, especially how well it handled volatile periods, visually confirms its strong performance shown by the low RMSE values.

7 Challenges and Limitations

Developing and testing these models came with several challenges:

- **Hyperparameter Tuning**: Finding the best settings for the models (like number of layers/units, learning rates, optimizers) was complex. Tuning GANs was especially tricky, requiring careful adjustment of parameters like how often to update the critic vs. the generator and the strength of the gradient penalty (for WGAN-GP). This took a lot of time and computational experiments.
- **Training Stability (GANs)**: Although WGAN-GP is much more stable than basic GANs, careful tuning was still needed to avoid problems like mode collapse or unstable training dynamics. The basic GAN likely suffered more from these issues, contributing to its poor results.
- **Computational Cost**: Training the large WGAN-GP model (Generator has over 6 million parameters, plus the Critic) required significant GPU time and resources, much more than the smaller GRU and LSTM models.
- **Data Quality and Feature Engineering**: Model performance heavily depends on good quality input data and well-chosen features. We used 36 features, but finding the absolute best set might require more investigation. Adding other data sources (like news sentiment or economic data) could potentially help but also adds complexity.
- **Market Unpredictability**: No model can perfectly predict truly unexpected events ("black swan" events) or fully capture complex human behavior in the market. While WGAN-GP handled 2020's volatility well, extreme, unforeseen events are always a challenge.
- **Evaluation Metrics**: RMSE measures the average error size, but for practical trading, other factors like predicting the direction of price movement correctly or estimating potential profit/loss are also important, and RMSE doesn't capture these directly.
- **Overfitting Risk**: With complex models like GANs, there's always a risk they might just memorize the training data instead of learning general patterns. We needed to monitor this, but the good test set results for WGAN-GP suggest it generalized reasonably well.

8 Conclusion and Future Work

8.1 Conclusion

This project successfully built and compared LSTM, GRU, Basic GAN, and WGAN-GP models for predicting stock prices one day ahead using historical data and technical indicators. The main conclusions are:

- The WGAN-GP model achieved the best overall prediction accuracy on the unseen test data, shown by its lowest RMSE.
- WGAN-GP was particularly strong during the highly volatile market of 2020, performing much better than LSTM, GRU, and the Basic GAN during that period. This shows its strength in capturing complex, non-linear market behavior.
- The simpler GRU model also performed respectably, especially in less volatile periods, making it a viable and efficient baseline option.
- The Basic GAN did not perform well compared to WGAN-GP, highlighting that the advanced training techniques used in WGAN-GP (Wasserstein loss, Gradient Penalty) are crucial for getting good results with GANs on financial time series.
- The results suggest that while standard models like GRU are effective, more advanced generative models like WGAN-GP can provide better performance, particularly when dealing with market volatility, potentially justifying their higher complexity and computational cost.

Although challenges exist, this work shows that WGAN-GP is a promising and powerful tool for stock price forecasting. The final prediction script also allows users to get forecasts for multiple days ahead by applying the trained model iteratively.

8.2 Future Work

There are several ways this research could be extended:

- **Explore Transformer Models**: Try using state-of-the-art Transformer models (like the original Transformer, Informer, Autoformer), which have performed very well on long time series forecasting tasks. They might capture long-range dependencies even better than GRUs/LSTMs, either within a GAN or as standalone predictors.
- **Advanced Feature Engineering**: Add more types of input features, such as macroeconomic data (interest rates, inflation), more complex technical indicators, market volatility measures (like VIX), and systematically test which features are most important (e.g., using SHAP analysis).
- **Refined Sentiment Analysis**: If feasible, integrate sentiment analysis from financial news or social media (e.g., using FinBERT) and carefully study its impact on prediction accuracy, especially during news-driven market moves.
- **Direct Multi-Step Forecasting**: Change the models (especially the Generator's output layer and the loss function) to directly predict multiple days ahead ($N > 1$) instead of just one. Compare this direct approach to the current method of predicting one day at a time iteratively.
- **Alternative GAN Architectures**: Experiment with other GAN types suitable for time series, like Conditional GANs (cGANs) where predictions can be guided by external conditions (e.g., market regime), or try different structures for the Generator/Discriminator (e.g., using Temporal Convolutional Networks - TCNs).

- **Robust Evaluation Metrics**: Evaluate the models using metrics more relevant to finance than just RMSE. Examples include Mean Absolute Percentage Error (MAPE), directional accuracy (how often the model predicts the correct up/down movement), or simulating trades based on predictions to estimate Profit & Loss (P&L), Sharpe ratio, and maximum drawdown.
- **Optimization and Deployment**: For the best model (WGAN-GP), look into techniques like model compression or quantization to make it smaller and faster, potentially allowing for real-time prediction in a trading system.
- **Interpretability**: Use techniques like SHAP or analyze attention mechanisms (if using Transformers) to better understand how the WGAN-GP model makes its predictions and which input features are most influential.

9 References

References

[1] P. Sonkiya, V. Bajpai, A. Bansal. *P. Sonkiya, V. Bajpai and A. Bansal, "Stock Price Prediction Using Artificial Intelligence: A Survey," 2022 IEEE 7th International conference for Convergence in Technology (I2CT), Mumbai, India, 2022, pp. 1-9*

[2] H-F. Lin, Y-C. Huang, et al. *Lin, H., Chen, C., Huang, G. & Jafari, A. (2021). Stock price prediction using Generative Adversarial Networks. Journal of Computer Science, 17(3), 188-196.*

[3] B. Banushev. (2020). *AI for Stock Market Prediction.*

[4] K. Zhang, P. Tan, X. Li. (2019). *GAN for Stock Market Prediction: Enhancing Forecast Accuracy with Generative Adversarial Networks.*

[5] X. Zhou, Y. Wang, J. Chen. (2018). *GANs for High-Frequency Trading: Analyzing Market Patterns with Deep Learning.* Mathematical Problems in Engineering, vol. 2018, Article ID 1840742.

[6] I. Goodfellow, et al. (2014). *Generative Adversarial Nets.* NIPS.

[7] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A. Courville (2017). *Improved Training of Wasserstein GANs.* NIPS.

[8] K. Cho, et al. (2014). *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation.* EMNLP.