# Stock Price Prediction Using Generative Adversarial Networks and Sentiment Analysis ( FinBERT and VADER )

Omkar Chavan

*Roll No 210280, chavan21@iitk.ac.in*

BTech, Mechanical Engineering,
Indian Institute of Technology, Kanpur

## ABSTRACT

Full papers are expected to be a minimum of 6 pages and a maximum of 8 pages in length. Short Papers (Posters) are limited to a minimum of 2 pages and a maximum of 4 pages. Please use this as a template for your paper. There is more formatting detail given below, if needed.

## 1. INTRODUCTION

Stock market forecasting is a fundamental area of research in financial analytics, enabling investors to anticipate market trends and make informed decisions. Despite significant advancements in machine learning and computational techniques, accurately predicting stock price movements remains challenging due to the complex and volatile nature of financial markets. Traditional models, such as Long Short-Term Memory (LSTM) networks, have shown promise in capturing temporal dependencies within historical price data. However, these models often fail to account for the role of market sentiment—a critical driver of stock prices during volatile periods.

Market sentiment reflects the collective emotions and opinions of investors, often derived from news articles, social media, and financial reports. Sentiment analysis techniques, such as the Valence Aware Dictionary for Sentiment Reasoning (VADER) and FinBERT—a financial variant of the BERT language model—offer powerful tools to quantify these sentiments. While VADER provides quick and interpretable sentiment scoring, FinBERT offers deeper contextual understanding, making it more suitable for financial text.

This study explores the integration of sentiment analysis with predictive models to enhance stock price forecasting. Initially, LSTM models were employed as a baseline to analyze the impact of sentiment scores from VADER and FinBERT. While effective in reducing error metrics such as Mean Squared Error (MSE) and Root Mean Squared Error (RMSE), the LSTM approach demonstrated limitations in capturing nonlinear relationships and generating realistic price sequences. To address these challenges, a Generative Adversarial Network (GAN) framework was introduced, leveraging sentiment scores to generate synthetic stock price predictions. However, traditional GANs are often unstable during training, and their loss metrics do not reliably correlate with prediction quality. To mitigate these issues, the study further incorporates the **Wasserstein GAN with Gradient Penalty (WGAN-GP)**, an advanced GAN architecture that improves stability and ensures better convergence.

The contributions of this research are fourfold:

1. **Sentiment Integration**: The study incorporates sentiment scores derived from VADER and FinBERT to evaluate their impact on stock price prediction accuracy.
2. **Model Comparison**: LSTM, GAN, and WGAN-GP models are benchmarked using key performance metrics, highlighting their strengths and limitations.
3. **FinBERT vs. VADER Comparison**: The study reveals that FinBERT consistently outperforms VADER in reducing prediction errors across all frameworks, demonstrating its suitability for financial sentiment analysis.

By combining sentiment analysis with advanced deep learning techniques, this research highlights the potential for improved stock price forecasting. The results emphasize the value of contextual sentiment analysis and robust generative architectures like WGAN-GP in enhancing prediction models, paving the way for innovative approaches to financial forecasting.

## 2. METHODOLOGY

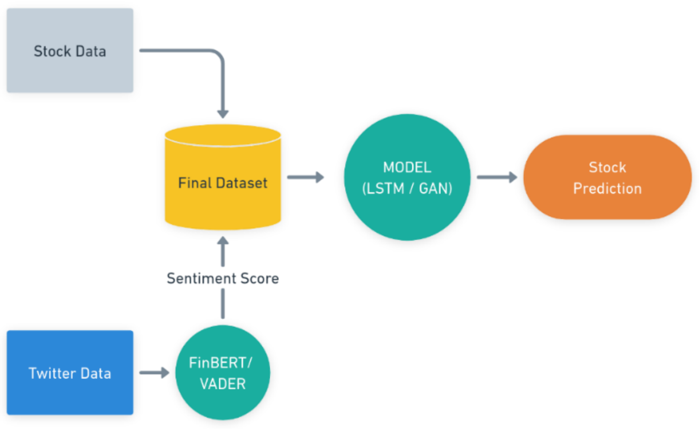

Figure 1: Project Structure

This project focuses on predicting stock prices by integrating sentiment analysis with predictive models, employing Long Short-Term Memory (LSTM) networks, Generative Adversarial Networks (GANs), and Wasserstein GANs with Gradient Penalty (WGAN-GP). The analysis was performed using stock data for **Amazon (AMZN)**, spanning the date range **September 30, 2021, to September 30, 2022.** The methodology involves data collection, sentiment analysis, data preprocessing, model development, and evaluation.

---

### *1. Data Collection*

- **Stock Data**:
  - Historical stock prices for Amazon were retrieved using the `yfinance` library. Key features included Open, High, Low, Close, and Volume prices for the specified date range.
- **Twitter Data**:
  - Tweets mentioning Amazon-related keywords were scraped using the Twikit API. The collected data included tweet content, timestamp, and metadata, which were used for sentiment analysis.

---

### *2. Sentiment Analysis*

- Sentiment analysis was conducted to capture investor emotions and opinions reflected in the tweets:
  - **VADER (Valence Aware Dictionary for Sentiment Reasoning)**:
    - A rule-based model that provides sentiment polarity (positive, negative, neutral) and compound scores. These scores were aggregated daily to align with the stock data.
  - **FinBERT**:
    - A transformer-based NLP model fine-tuned on financial text. It generates more nuanced sentiment scores (positive, neutral, and negative) suitable for financial contexts.

The sentiment scores from both models were used as features to enhance stock price predictions.

---

## 4. Data Preprocessing

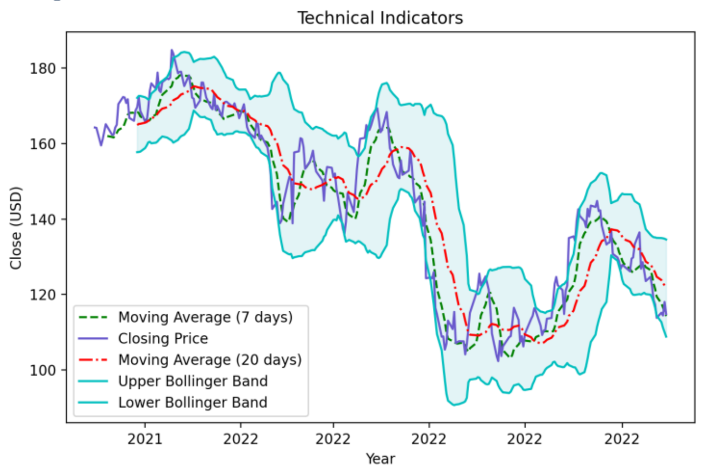

Figure 2: Stock Closing Price with Technical Indicators

### *3.1 Data Integration*

- **Stock Data**:
    - Historical stock prices, including `Open`, `Close`, `High`, `Low`, and `Volume`, were merged with sentiment scores using the **'Date'** column to ensure temporal alignment.
- **Sentiment Data**:
    - Daily aggregated sentiment scores (positive, neutral, negative) from FinBERT analysis were synchronized with stock data.

### *3.2 Technical Indicators*

To enhance feature representation, the following technical indicators were computed:

- **Moving Averages**:
    - 7-day (`MA7`) and 20-day (`MA20`) simple moving averages for short- and medium-term trend analysis.
- **MACD (Moving Average Convergence Divergence)**:
    - Captured the difference between the 26-day and 12-day exponential moving averages of `Close` and `Open` prices.
- **Bollinger Bands**:
    - Constructed using the 20-day standard deviation around the `MA20` to define upper and lower price bounds.
- **Exponential Moving Average (EMA)**:
    - Highlighted smoothed trends in stock prices.
- **Log-Momentum**:
    - Logarithmic transformation of price momentum to detect market shifts.

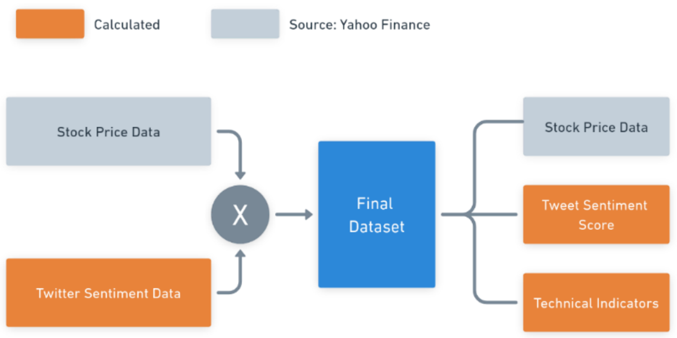

Figure 3: Structure of Dataset

### *3.3 Handling Missing Values*

- Missing sentiment scores, arising from non-trading days or limited news coverage, were imputed with **neutral values**, ensuring data consistency without introducing biases.

### *3.4 Feature Scaling and Normalization*

- All numerical features, including stock prices, sentiment scores, and technical indicators, were scaled using **MinMaxScaler**, normalizing values between 0 and 1 to facilitate model convergence.

### *3.5 Dataset Splitting*

- Data was divided into training (80%) and testing (20%) subsets, ensuring chronological order to avoid data leakage during model evaluation.

### *3.6 Feature Engineering*

- Combined historical stock prices, sentiment scores, and technical indicators formed the input dataset, with the `Close` price as the target variable for prediction.

---

## *4. Model Development*

### 4.1. LSTM

- LSTM was used as the baseline model to capture temporal dependencies in stock price data.
- **Architecture**:
  - Input Layer: Historical stock prices and sentiment scores.
  - Hidden Layers: Multiple LSTM layers with 50 epochs of training.
  - Output Layer: Predicted stock prices for the next time step.
- Two configurations were developed:
  1. LSTM with VADER sentiment scores.
  2. LSTM with FinBERT sentiment scores.

### 4.2. GAN

- GANs were introduced to overcome the limitations of LSTM in modeling complex, nonlinear relationships in stock price data.
- **Architecture**:
  - **Generator**:
    - Composed of five stacked LSTM layers with decreasing units (1024 to 64) and recurrent dropout for regularization.
    - Outputs synthetic stock price sequences based on input features and sentiment data.
  - **Discriminator**:
    - A series of Conv1D layers with Leaky ReLU activations to process sequential data.
    - Outputs a probability score indicating whether the input data is real or synthetic.
- **Training**:
  - The generator and discriminator were trained adversarially for 5000 epochs.
  - Sentiment scores (VADER and FinBERT) were included as conditional inputs.
- Two configurations were developed:
  1. GAN with VADER sentiment scores.
  2. GAN with FinBERT sentiment scores.

### *4.3 WGAN-GP*

- **Purpose**: WGAN-GP was implemented to improve the stability and convergence of GANs while addressing training instabilities such as mode collapse.
- **Architecture**:
    - **Generator**:
        - Replaced LSTM layers with GRU layers to capture sequential dependencies, with decreasing units (256 to 128).
        - Dense layers with L2 regularization refined the output.
    - **Discriminator (Critic)**:
        - Similar to the GAN Discriminator, composed of Conv1D layers with Leaky ReLU activations.
        - Outputs Wasserstein scores to distinguish between real and synthetic data.
- **Training**:
    - The generator and critic were trained alternately using the Wasserstein loss with gradient penalty for 1500 epochs.
    - Sentiment scores (VADER and FinBERT) were incorporated as features.
- **Configurations**:
    1. WGAN-GP with VADER sentiment scores.
    2. WGAN-GP with FinBERT sentiment scores.

---

## *5. Evaluation*

- **Metrics**:
    - **Mean Squared Error (MSE)**: Average squared differences between predicted and actual prices.
    - **Mean Absolute Percentage Error (MAPE)**: Prediction accuracy as a percentage.
    - **Root Mean Squared Error (RMSE)**: Magnitude of prediction errors.
- **Comparative Analysis**:
    - **Models**: LSTM vs. GAN vs. WGAN-GP frameworks.
    - **Sentiment Techniques**: VADER vs. FinBERT.

---

# 3. Theoretical Background

### *4.1 Recurrent Neural Networks (RNNs)*

Recurrent Neural Networks (RNNs) are a foundational class of neural networks designed for sequential data. Unlike feedforward networks, RNNs introduce a **feedback loop** through their hidden layers, allowing information to persist over time steps. This capability makes RNNs well-suited for tasks such as time-series prediction, speech recognition, and natural language processing.

**Mathematical Formulation of RNNs**: At each time step $t$:

$$h_t = \phi(W_h h_{t-1} + W_x x_t + b_h)$$

where:

- $h_t$: Hidden state at time step $t$.
- $h_{t-1}$ : Hidden state from the previous time step $t-1$.
- $x_t$ : Input vector at time step $t$.
- $W_h$: Weight matrix for the hidden state.
- $W_x$ : Weight matrix for the input.
- $b_h$: Bias vector for the hidden state.
- ϕ: Activation function, typically $tanh$ or σ (sigmoid).

RNNs suffer from the **vanishing gradient problem**, which limits their ability to learn long-term dependencies. To overcome these limitations, more advanced architectures like **Long Short-Term Memory (LSTM) networks** were developed.

---

### *4.2 Long Short-Term Memory (LSTM) Networks*

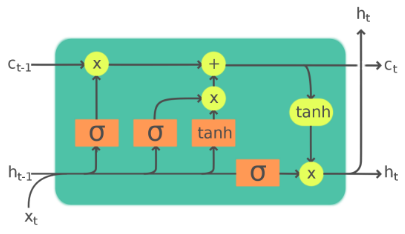

Figure 4: LSTM Cell

Recurrent Neural Networks (RNNs) are commonly used for tasks involving sequential data, such as stock price prediction. However, traditional RNNs face challenges in capturing long-term dependencies due to the **vanishing gradient problem**, which makes training ineffective as the sequence length increases. Long Short-Term Memory (LSTM) networks address this limitation by introducing a specialized cell structure with **gating mechanisms** to regulate the flow of information.

### *LSTM Architecture*

An LSTM cell consists of three primary gates: the **Forget Gate**, the **Input Gate**, and the **Output Gate**, which work together to update the cell state $C_t$ and the hidden state $h_t$ at each time step $t$. The functionality of each gate is as follows:

#### 1. Forget Gate

The forget gate determines which parts of the previous cell state $C_{t-1}$ to retain or discard. The output of the forget gate $f_t$ is computed as:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

#### 2. Input Gate

The input gate decides which new information should be added to the current cell state. It involves two components:

1. A sigmoid layer to determine relevance:
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

2. A hyperbolic tangent layer to create candidate values:
$$\widetilde{C_t} = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

#### 3. Cell State Update

The cell state $C_t$ is updated by combining the retained information from $C_{t-1}$ with new information from the input gate:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \widetilde{C_t}$$

### 4. Output Gate

The output gate determines which parts of the updated cell state $C_t$ contribute to the hidden state $C_t$. It involves:

1. A sigmoid function to calculate the output gate activation:
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

2. A scaled version of the cell state:
$$h_t = o_t \cdot \tanh(C_t)$$

---

## 4.3 Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs), introduced by Ian Goodfellow in 2014, are a class of deep learning models designed for generating realistic data distributions. GANs consist of two neural networks: the **Generator** and the **Discriminator**, which compete in an adversarial framework. The Generator synthesizes data to resemble real samples, while the Discriminator attempts to distinguish between real and synthetic data.

This adversarial training mechanism enables GANs to learn complex data distributions, making them particularly effective for tasks such as time-series forecasting.

### *GAN Architecture*

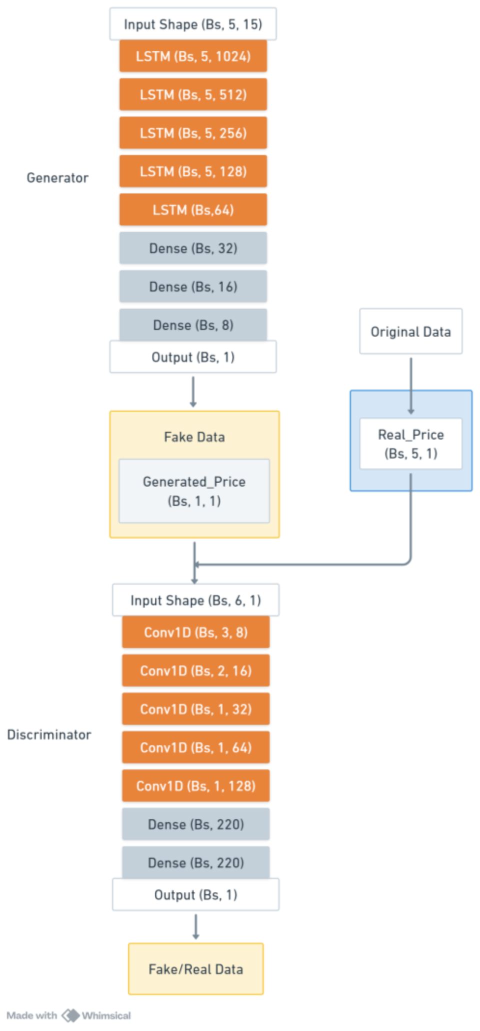

Figure 5: GAN Architecture

The Generator and Discriminator networks in GANs have opposing objectives, represented by a **minimax loss function**. The optimization process can be formulated as:

$$\min_G \max_D E_{x\sim p_{\mathrm{dt}}}[\log D(x)] + E_{z\sim p_z(z)}\left[\log\left(1 - D\big(G(z)\big)\right)\right]$$

**Components**:

1. **Generator** ($G$):
   - Takes a random noise vector $z$ and generates synthetic stock prices.
   - Learns to produce outputs $G(z)$ that are indistinguishable from real stock prices.
2. **Discriminator** ($D$):
   - Takes an input $x$ and predicts the probability that x is real.
   - Learns to correctly classify real data (x) and generated data ($G(z)$).

**Discriminator Loss**: The Discriminator aims to maximize its ability to differentiate between real and synthetic data:

$$\mathcal{L}_{\mathcal{D}} = -E_{x\sim p_{\mathrm{dt}}}[\log D(x)] - E_{z\sim p_z(z)}\left[\log\left(1 - D\big(G(z)\big)\right)\right]$$

**Generator Loss**: The Generator attempts to minimize the Discriminator's ability to distinguish synthetic data:

$$\mathcal{L}_{\mathcal{G}} = -E_{z\sim p_z(z)}\left[\log D\left(G(z)\right)\right]$$

where,

- $D(x)$: Probability that $x$ is real data.
- $G(z)$ : Synthetic data generated by the Generator.
- $p_{data}$ : Distribution of real data (stock prices).
- $\mathrm{p_z(z)}$: Distribution of random noise.
- $\mathcal{L}_{\mathcal{D}}, \mathcal{L}_{\mathcal{G}}$ : Loss functions for the Discriminator and Generator, respectively.

### *Training Workflow*

1. **Generator**:
   - Generates synthetic stock price sequences from random noise zzz.
   - Employs LSTM layers to model temporal dependencies in the data.
2. **Discriminator**:
   - Evaluates the authenticity of the sequences generated by the Generator.
   - Uses Conv1D layers to process sequential data and distinguish real from fake sequences.
3. **Adversarial Training**:
   - The Generator and Discriminator are trained alternately:
     - The Generator is updated to minimize $\mathcal{L}_{\mathcal{G}}$, making its outputs more realistic.
     - The Discriminator is updated to minimize $\mathcal{L}_{\mathcal{D}}$ , improving its ability to distinguish real and fake data.

### *Advantages of GANs for Stock Price Prediction*

- **Modelling Nonlinear Dependencies**: GANs excel at capturing complex, nonlinear relationships in financial time series.
- **Synthetic Data Generation**: GANs can produce realistic stock price sequences that align with observed trends.
- **Adversarial Learning**: The competitive training process ensures that both the Generator and Discriminator improve iteratively.

## 4.4 Wasserstein GAN with Gradient Penalty (WGAN-GP)

Wasserstein GAN with Gradient Penalty (WGAN-GP) is an extension of the traditional GAN framework designed to address common training instabilities such as mode collapse and vanishing gradients. Unlike standard GANs, WGAN-GP replaces the Discriminator with a **Critic** that scores data samples based on their Wasserstein distance from the real data distribution. This approach improves the training process by ensuring better convergence and stability.

### *WGAN-GP Architecture*

**Figure 5: WGAN-GP Architecture**
The Generator and Critic networks in WGAN-GP are trained using the Wasserstein loss function. The gradient penalty is applied to enforce Lipschitz continuity in the Critic, ensuring stable training. The optimization process can be formulated as:

$$\min_G \max_C E_{x\sim P_r}[C(x)] - E_{\tilde{x}\sim P_g}[C(\tilde{x})] + \lambda \cdot E_{\hat{x}\sim P_{\hat{x}}}[(|\nabla_{\hat{x}} C(\hat{x})|_2 - 1)^2]$$

where:

- $C(x)$:Critic output (realness score) for real data $x$.
- $\tilde{x} = G(z)$:Synthetic data generated by the Generator from noise $z$.
- $P_r$:Real data distribution.
- $P_g$:Generated data distribution.
- $\hat{x}$:Linear interpolation between real and fake data.
- $\lambda$:Gradient penalty coefficient.

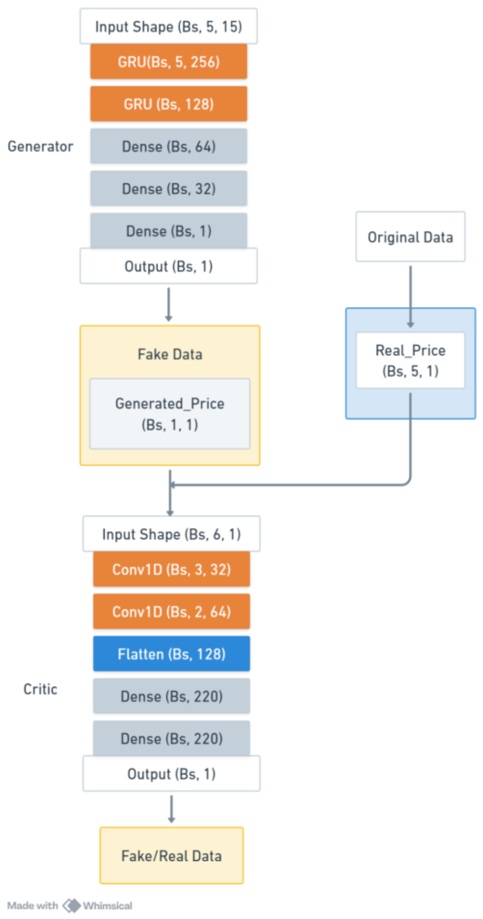

Figure 6: WGAN-GP Architecture

### *Components:*

1. **Generator** (G):
   - Takes a random noise vector z and generates synthetic stock prices G(z)
   - Learns to produce outputs indistinguishable from real stock prices by minimizing the Critic loss.
2. **Critic** ($C$):
   - Evaluates the "realness" of both real and synthetic data.
   - Outputs a score $C(x)$ for real data and $C(\tilde{x})$ for synthetic data.
   - Enforces Lipschitz continuity through gradient penalty.

### *Lipschitz Continuity*

A function $f$ is Lipschitz continuous if there exists a constant $K$ such that:

$$|f(x_1) - f(x_2)| \leq K \cdot |x_1 - x_2|$$

### *Gradient Penalty*

To enforce Lipschitz continuity, WGAN-GP applies a **gradient penalty** to the Critic. This penalty ensures that the gradient norm of the Critic's output with respect to its inputs is close to 1:

$$GP = E_{\hat{x} \sim P_{\hat{x}}}[(|\nabla_{\hat{x}} C(\hat{x})|_2 - 1)^2]$$

### *Loss Functions*

1. **WGAN-GP Loss Function**:
$$\min_G \max_C E_{x \sim P_r}[C(x)] - E_{\tilde{x} \sim P_g}[C(\tilde{x})] + \lambda \cdot E_{\hat{x} \sim P_{\hat{x}}}[(|\nabla_{\hat{x}} C(\hat{x})|_2 - 1)^2]$$
2. **Critic Loss**:
$$\mathcal{L}_{\mathcal{C}} = E_{\tilde{x} \sim P_g}[C(\tilde{x})] - E_{x \sim P_r}[C(x)] + \lambda \cdot E_{\hat{x} \sim P_{\hat{x}}}[(|\nabla_{\hat{x}} C(\hat{x})|_2 - 1)^2]$$

- Maximizes the distance between real and generated data distributions while penalizing gradients to maintain Lipschitz continuity.

3. **Generator Loss**:
$$\mathcal{L}_{\mathcal{G}} = -E_{\tilde{x} \sim P_g}[C(\tilde{x})]$$

- $D(x)$Minimizes the Critic's ability to distinguish real from synthetic data, forcing the Generator to improve.

### *Training Workflow*

1. **Generator**:
   - Generates synthetic stock price sequences from random noise zzz.
   - Employs GRU layers to capture temporal dependencies in stock price data.
2. **Critic**:
   - Evaluates both real and synthetic data samples.
   - Enforces Lipschitz continuity through gradient penalty during training.
3. **Adversarial Training**:
   - The Generator and Critic are trained alternately:
     - The Generator is updated to minimize $\mathcal{L}_{\mathcal{G}}$ making its outputs more realistic.
     - The Critic is updated to maximize $\mathcal{L}_{\mathcal{C}}$, improving its scoring ability.

### *Advantages of WGAN-GP for Stock Price Prediction*

1. **Improved Stability**: The Wasserstein loss with gradient penalty ensures stable training and mitigates mode collapse.
2. **Better Generalization**: The model learns a more realistic data distribution by directly optimizing the Wasserstein distance.
3. **Synthetic Data Generation**: Produces realistic stock price sequences aligned with observed trends.
4. **Adversarial Learning**: The iterative training process refines both the Generator and Critic, leading to improved results over time.

---

## 4.5 Sentiment Analysis Techniques: VADER and FinBERT

Sentiment analysis serves as a crucial component in this study, providing insights into market sentiment, which significantly influences stock price movements. This section highlights two distinct sentiment analysis techniques, **VADER** and **FinBERT**, used to quantify market sentiment from social media and financial texts.

### *1. VADER (Valence Aware Dictionary and sEntiment Reasoner)*

VADER is a lexicon and rule-based sentiment analysis tool designed for analyzing short, informal texts such as tweets and social media posts. Its computational efficiency and simplicity make it particularly suitable for processing large datasets of high-volume, noisy data. VADER assigns valence scores to words using a predefined dictionary, where each word is associated with a sentiment polarity—positive, negative, or neutral. Sentiment is calculated as a weighted average of these scores, with a compound score (ranging from -1 to 1) summarizing the overall sentiment of the text. VADER's primary strengths lie in its lightweight nature, clear interpretability, and ability to handle informal text effectively, including slang, emojis, and casual language commonly found on social media platforms.

### *2. FinBERT*

FinBERT is a transformer-based model specifically fine-tuned for sentiment analysis in the financial domain. Built on the robust architecture of BERT, it leverages self-attention mechanisms to capture complex relationships between words in financial texts, such as earnings reports, news articles, and market analyses. FinBERT classifies sentiment into three categories—positive, neutral, and negative—while also providing confidence scores for each classification. Its ability to understand the context of words sets it apart from lexicon-based methods, making it effective for analyzing nuanced financial language. Fine-tuned on large-scale financial corpora, FinBERT demonstrates high accuracy and reliability in domain-specific sentiment analysis, offering valuable insights into market sentiment from structured financial texts.

## 5. Results and Discussions

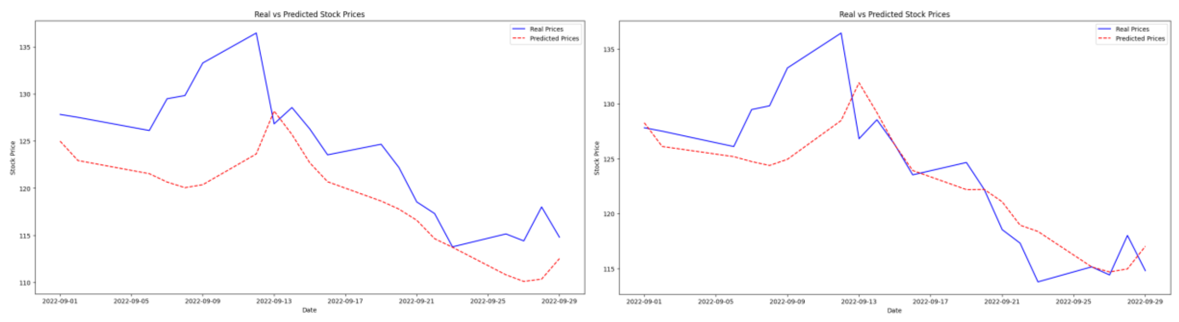

Figure 6: LSTM Prediction (1. with VADER and 2. With FinBERT)

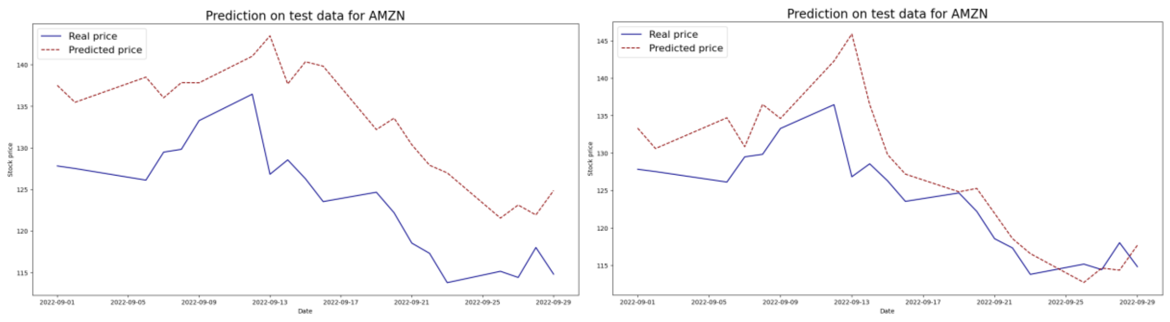

Figure 7: Basic GAN Prediction (1. with VADER and 2. With FinBERT)

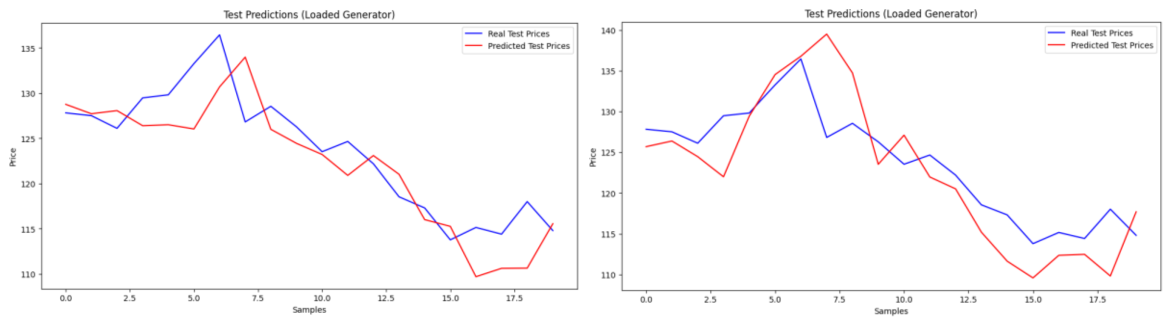

Figure 7: WGAN-GP Prediction (1. with VADER and 2. With FinBERT)

## *Performance Metrics*

The performance of the models was evaluated using three key metrics: Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE). The results for each configuration are presented below:

## *Table of Results*

| **Deep Learning Model** | **Sentiment Analysis Model** | **Epochs** | **MAPE** | **MSE** | **RMSE** |
|---|---|---|---|---|---|
| LSTM | VADER | 50 | 0.0398 | 37.6372 | 6.1349 |
| LSTM | FinBERT | 50 | 0.0207 | 13.2465 | 3.6396 |
| GAN | VADER | 5000 | 0.0787 | 106.4511 | 10.3175 |
| GAN | FinBERT | 5000 | 0.0344 | 35.2645 | 5.9384 |
| WGAN-GP | VADER | 1500 | 0.0248 | 14.7830 | 3.8449 |
| WGAN-GP | FinBERT | 1500 | 0.0298 | 22.2314 | 4.7150 |

## *Discussions*

LSTM models effectively capture temporal dependencies in stock price data. Incorporating FinBERT sentiment scores significantly improves performance over VADER, as evidenced by lower error metrics across all three metrics. This highlights the value of FinBERT's domain-specific sentiment analysis for financial forecasting.

While GANs introduce the ability to model nonlinear relationships and generate synthetic stock price sequences, their performance using VADER sentiment scores was less effective compared to FinBERT. The GAN model with FinBERT sentiment integration demonstrates significant improvement over VADER, reducing MSE by ~67% and RMSE by ~42%.

The WGAN-GP framework enhances training stability and reduces mode collapse through gradient penalty. While WGAN-GP achieves slightly better results than LSTM with VADER, the FinBERT-based configuration shows mixed results compared to LSTM. However, WGAN-GP demonstrates a balance between performance and the ability to model complex data distributions, making it more robust for generating realistic predictions.

### *4. Comparative Analysis*

A comparative analysis of all models highlights the following key insights:

- **FinBERT vs. VADER**:
  Across all configurations (LSTM, GAN, WGAN-GP), FinBERT consistently outperforms VADER, demonstrating its superior ability to capture nuanced sentiment in financial data.
- **GAN vs. WGAN-GP**:
  WGAN-GP provides more stable training and achieves better results than GAN with VADER sentiment scores. However, its performance with FinBERT sentiment scores does not surpass LSTM.
- **Best Model**:
  LSTM with FinBERT achieved the lowest MSE (13.2465) and RMSE (3.6396), demonstrating its effectiveness in forecasting stock prices with sentiment analysis.

## 4. CONCLUSIONS

The results emphasize the importance of sentiment integration in stock price forecasting. FinBERT consistently outperforms VADER across all models, and while GAN and WGAN-GP bring robustness to modeling complex relationships, LSTM with FinBERT remains the most effective configuration for this dataset.

## 5. REFERENCES

- Arjovsky, M., Chintala, S., and Bottou, L., (2017), Wasserstein GAN, arXiv preprint, arXiv:1701.07875.

- Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., (2014), Generative Adversarial Nets, arXiv preprint, arXiv:1406.2661.

- Halder, S., (2022), FinBERT-LSTM: Deep Learning based stock price prediction using News Sentiment Analysis, arXiv:2211.07392

- Jain, J. K., & Agrawal, R. (2023). FB-GAN: A Novel Neural Sentiment-Enhanced Model for Stock Price Prediction.