Financial Signal Processing Final Summary

Yung-Yi Lin, b10901059
Department of Electrical Engineering
National Taiwan University
Taiwan, Taipei
https://github.com/lin-1214/SPLab

Abstract

This semester's exploration of financial signal processing progressed from foundational statistical modeling to advanced machine learning and deep learning techniques. We began with traditional time series approaches, including Kalman and particle filters, ARMA, and ARIMA models, to analyze stock price dynamics. Building on this foundation, we applied recurrent neural networks such as RNNs, LSTMs, and GRUs to capture sequential dependencies in financial data. In the latter phase, we investigated advanced topics in financial technology, including enhancing LSTM models through sentiment analysis using large language models, utilizing Relational Temporal Graph Convolutional Networks to capture inter-stock relationships and temporal features, and applying generative learning techniques to mitigate the lack of sufficient historical data. These advanced approaches are increasingly vital in today's complex and data-constrained financial environments, where traditional models often fall short in capturing nonlinear patterns, dynamic relationships, and emerging information sources.

1 Introduction

Financial signal processing has become an increasingly vital field at the intersection of statistics, machine learning, and financial engineering. As global financial markets grow in complexity, traditional modeling techniques often struggle to keep pace with the nonlinear dynamics and external information sources that influence asset prices. This semester, our work aimed to explore both classical and contemporary approaches for modeling and forecasting stock price behavior, progressing from time-tested statistical methods to recent advances in deep learning and graph-based learning frameworks.

We began by studying traditional time series models such as ARMA and ARIMA, as well as state estimation techniques like the Kalman filter and particle filter. These methods provided a solid baseline for understanding the underlying trends and noise in stock price data. However, their reliance on linearity and stationarity motivated the exploration of more flexible modeling frameworks.

Next, we transitioned to machine learning approaches, employing recurrent neural networks (RNNs) including LSTM and GRU models. These architectures are particularly suited for capturing temporal dependencies and long-term patterns in financial data. Their ability to handle nonlinearity marked a significant improvement over conventional methods.

Building on this foundation, we delved into three advanced research directions that are increasingly relevant in today's data-driven and information-rich financial environment. First, we enhanced LSTM models with sentiment signals extracted from financial texts using large language models, aiming to incorporate qualitative market information. Second, we implemented Relational Temporal Graph Convolutional Networks (RT-GCN) to model inter-stock relationships and time-sensitive interactions

Preprint. Under review.

across assets. Finally, we applied generative learning techniques to address the challenge of limited labeled data, a common constraint in financial time series prediction.

Together, these explorations underscore a broader trend in financial technology: the shift from isolated, signal-level modeling to integrated, multi-modal approaches that leverage structure, context, and inference. In this report, we present a detailed overview of each stage of this progression and discuss their implications for future research and application.

2 Methodology Overview

2.1 Traditional Statistical Models

Traditional statistical models offer foundational tools for analyzing and forecasting financial time series. These models assume certain stochastic structures in the data and often provide interpretable and efficient estimations. In this section, we cover three key approaches: the Kalman filter, particle filter, and ARMA/ARIMA models.

2.1.1 Kalman Filter

The Kalman filter is an optimal recursive estimator for linear Gaussian state-space models. It is particularly useful in finance for estimating latent variables such as trends or volatility from noisy observations. The model is defined by:

$$x_t = Ax_{t-1} + w_t, \quad w_t \sim \mathcal{N}(0, Q) \tag{1}$$
$$z_t = Hx_t + v_t, \quad v_t \sim \mathcal{N}(0, R) \tag{2}$$

Here, x_t is the hidden state (e.g., the true value of a stock), z_t is the observed measurement (e.g., the market price), A is the state transition matrix, H is the observation matrix, and w_t, v_t represent process and observation noise, respectively. The filter operates in two steps:

Prediction step

$$\hat{x}_{t|t-1} = A\hat{x}_{t-1|t-1} \tag{3}$$
$$P_{t|t-1} = AP_{t-1|t-1}A^\top + Q \tag{4}$$

Update step

$$K_t = P_{t|t-1}H^\top (HP_{t|t-1}H^\top + R)^{-1} \tag{5}$$
$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(z_t - H\hat{x}_{t|t-1}) \tag{6}$$
$$P_{t|t} = (I - K_tH)P_{t|t-1} \tag{7}$$

This recursive algorithm efficiently updates the estimate $\hat{x}_{t|t}$ of the hidden state given all observations up to time t.

2.1.2 Particle Filter

The particle filter extends the Kalman filter to nonlinear, non-Gaussian models using a Monte Carlo approach. It approximates the posterior distribution of the hidden states with a set of weighted particles $\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^N$. The particle filter involves three main steps:

Sampling (Prediction) will generate particles from the transition model:

$$x_t^{(i)} \sim p(x_t|x_{t-1}^{(i)})$$

Weighting (Update) assigns importance weights based on the likelihood of the observation:

$$w_t^{(i)} \propto p(z_t|x_t^{(i)})$$

Resampling resamples the particles according to their weights to avoid degeneracy.

This method is flexible and powerful for financial systems with complex dynamics, such as regime-switching or jump processes, but is computationally more intensive than the Kalman filter.

2.1.3 ARMA and ARIMA

ARMA (AutoRegressive Moving Average) and ARIMA (AutoRegressive Integrated Moving Average) models are classic time series models used for forecasting stationary and non-stationary data, respectively.

An ARMA(p, q) model is defined as:

$$y_t = \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t \tag{8}$$

where ϕ_i are the autoregressive coefficients, θ_j are the moving average coefficients, and ε_t is white noise.

For non-stationary time series, differencing is applied. An ARIMA(p, d, q) model is written as:

$$\Delta^d y_t = \sum_{i=1}^{p} \phi_i \Delta^d y_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t \tag{9}$$

where $\Delta^d y_t$ denotes the d-th order differenced series to remove trends or seasonality.

These models are interpretable and efficient for linear processes, often serving as baselines in financial forecasting. However, they lack the capacity to capture nonlinear dependencies or external signals.

2.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) and their variants are widely used for modeling sequential data, particularly time series. In financial signal processing, they are employed to capture temporal dependencies and patterns that traditional models often fail to represent. Unlike feedforward neural networks, RNNs maintain a hidden state that evolves over time, enabling the network to retain information across time steps.

2.2.1 RNN

A standard Recurrent Neural Network processes sequences by recursively updating its hidden state. Given a sequence of inputs $\{x_1, x_2, \ldots, x_T\}$, the RNN computes:

$$h_t = \tanh(W_{xh} x_t + W_{hh} h_{t-1} + b_h) \tag{10}$$

$$y_t = W_{hy} h_t + b_y \tag{11}$$

where:

- h_t is the hidden state at time t
- x_t is the input at time t
- y_t is the output at time t
- W_{xh}, W_{hh}, W_{hy} are weight matrices
- b_h, b_y are bias terms

The recurrent structure allows RNNs to retain memory of past inputs. However, in practice, standard RNNs suffer from the vanishing or exploding gradient problem, making them ineffective for capturing long-term dependencies—this is especially problematic in financial time series with seasonal or delayed effects.

2.2.2 LSTM

Long Short-Term Memory (LSTM) networks are designed to address the limitations of standard RNNs by introducing gating mechanisms that regulate the flow of information. An LSTM unit maintains a **cell state** c_t and uses three gates: input, forget, and output gates.

Given input x_t and previous hidden state h_{t-1} and cell state c_{t-1}, the LSTM computes:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{12}$$
$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{13}$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{14}$$
$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{15}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{16}$$
$$h_t = o_t \odot \tanh(c_t) \tag{17}$$

where σ denotes the sigmoid function and $\odot$ is the element-wise product. The gates control how much of the past cell state is retained (f_t), how much of the new candidate state is added (i_t), and how much of the cell state is exposed as output (o_t). This architecture allows LSTM to learn long-term dependencies effectively, which is crucial in financial series where trends, shocks, or cycles occur over extended horizons.

2.2.3 GRU

The Gated Recurrent Unit (GRU) is a simplified version of the LSTM that combines the input and forget gates into a single update gate and merges the cell and hidden states. This makes GRUs computationally lighter while still capable of learning long-term dependencies.

The GRU computes the hidden state as follows:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \quad \text{(update gate)} \tag{18}$$
$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \quad \text{(reset gate)} \tag{19}$$
$$\tilde{h}_t = \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \tag{20}$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{21}$$

Here, the update gate z_t decides how much of the past information to keep, and the reset gate r_t determines how much of the past to forget when generating a new candidate state $\tilde{h}_t$. GRUs are often preferred for financial applications where training time and model simplicity are important, while still offering performance competitive with LSTMs.

2.3 Advanced FinTech Approaches

2.3.1 BERT for Stock Market Sentiment Analysis

News sentiment plays a pivotal role in influencing financial markets, particularly during volatile periods or before market openings. However, the massive volume and speed of news generation make it impractical for human analysts to process all information in real time. To address this challenge, this study integrates Bidirectional Encoder Representations from Transformers (BERT) for the task of stock market sentiment analysis.

BERT is a pre-trained language model based on the Transformer encoder architecture, which uses self-attention mechanisms to learn contextual representations of language bidirectionally. This property makes BERT especially effective in sentiment analysis, where context often determines polarity. The model is fine-tuned for a classification task by appending a softmax layer over the [CLS] token representation, which summarizes the entire document.

The research involves three key steps:

1. **Data Collection and Preprocessing:** Financial news articles from sources like CNBC, Forbes, and Business Insider were manually labeled as positive, neutral, or negative, creating a dataset of 582 documents. Tokenization was performed using WordPiece with a vocabulary of 30,000 subword tokens.

2. **Model Fine-tuning:** The BERT-Base model (12 layers, 768 hidden units, 12 attention heads) was fine-tuned using 10-fold cross-validation. This approach enables BERT to adapt its general language understanding capabilities to the financial sentiment domain.
3. **Market Integration:** The sentiment distribution of news articles published in the hours leading up to the Dow Jones Industrial (DJI) market open was correlated with same-day market movement. Sentiment polarity was used as a proxy for predicting the DJI trend.

In comparison with baseline methods—Naive Bayes, Support Vector Machines, and TextCNN—BERT achieved the highest performance, with an F1-score of 72.5% and ROC AUC of 0.87. Table 1 summarizes these results.

Table 1: 10-fold Cross-Validation Performance Comparison

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1-score**
Naive Bayes (tf-idf)	0.610	0.607	0.568	0.542
SVM (tf-idf)	0.624	0.631	0.595	0.578
TextCNN	0.739	0.703	0.500	0.569
BERT	**0.825**	**0.750**	**0.713**	**0.725**

The study also conducted a time-series alignment between the sentiment scores and DJI fluctuations. By comparing sentiment polarity five hours before market open to daily stock index changes, the method correctly predicted market direction in 69% of the test cases. This highlights the potential of BERT-based sentiment signals as a supplementary indicator in stock market forecasting pipelines.

Despite limitations due to dataset size and market noise, this work demonstrates that integrating large language models into financial analysis pipelines can provide meaningful predictive signals, especially in latency-sensitive decision-making contexts.

2.3.2 Relational Temporal Graph Convolutional Networks for Ranking-Based Stock Prediction

Traditional stock prediction models often treat stocks as independent time series, failing to capture the interdependencies that exist across different firms in the financial market. Additionally, many methods approach prediction as a regression or classification problem, which is suboptimal when the true investment objective is to select top-performing stocks. To address these issues, the RT-GCN (Relational Temporal Graph Convolutional Network) model integrates both temporal evolution and inter-stock relations in a unified framework and formulates stock prediction as a ranking task.

The model constructs a *relation-temporal graph* $G^{RT} = (V, E)$, where each node represents a stock at a given time, and the edges represent either temporal continuity or relational links (e.g., same industry, supplier-customer). The features of each node include historical closing prices and moving averages. The graph consists of:

- **Relational edges**: connecting different stocks with known relations.
- **Temporal edges**: connecting the same stock across consecutive time steps.

The RT-GCN then applies a combination of graph convolution and temporal convolution to extract joint spatio-temporal features:

$$Z = D^{-\frac{1}{2}} \tilde{A} D^{-\frac{1}{2}} X \Theta, \tag{22}$$

where $\tilde{A} = A + I$ is the adjacency matrix with self-loops, D is the degree matrix, X is the input feature matrix, and Θ are learnable weights.

Three relation-aware strategies are proposed:

- **Uniform strategy**: treats all relation types equally.
- **Weight-based strategy**: learns a separate importance weight for each relation type.
- **Time-sensitive strategy**: combines dot-product similarity of node features with relational weights to adapt to temporal dynamics.

Temporal dependencies are modeled via 1D causal convolutions, defined as:

$$O(x_t) = \sum_{i=0}^{k-1} F(i) \cdot x_{t-k+i}, \tag{23}$$

where F is a filter of width k, applied only to current and past inputs, ensuring no future data leakage.

After extracting features, an average pooling layer summarizes the representation for each stock, followed by a fully connected layer to produce a ranking score. The model is trained using a combination of regression loss and pairwise ranking loss:

$$\tau_{\text{reg}} = \|\hat{r}_{t+1} - r_{t+1}\|^2, \tag{24}$$

$$\tau_{\text{rank}} = \sum_{i,j} \max\left(0, -(\hat{r}_i - \hat{r}_j)(r_i - r_j)\right), \tag{25}$$

$$\tau = \tau_{\text{reg}} + \alpha\tau_{\text{rank}} + \lambda\|\beta\|_2^2, \tag{26}$$

where $\hat{r}$ are predicted scores, r are ground-truth returns, and α, λ are regularization weights.

Empirical evaluation on NASDAQ, NYSE, and CSI datasets shows that RT-GCN outperforms state-of-the-art baselines in both return prediction accuracy and computational efficiency. Notably, the time-sensitive strategy achieves the best performance by dynamically adapting relationship weights across time. Compared to models like RSR and RankLSTM, RT-GCN provides faster training and better investment returns by jointly modeling relational and temporal signals in a single pass.

This approach marks a significant advancement in financial time series modeling, particularly in portfolio selection contexts where understanding inter-stock influence is crucial for maximizing returns.

2.3.3 FTS-Diffusion: Generative Learning for Financial Time Series with Irregular and Scale-Invariant Patterns

Data scarcity and irregularity in financial time series remain major challenges in applying deep learning to finance. To address this, the FTS-Diffusion framework introduces a generative architecture that models and synthesizes financial time series with two key characteristics: irregular recurrence and scale-invariance. These properties reflect the reality that financial patterns often recur at inconsistent intervals and with variable magnitudes and durations—unlike many natural time series such as ECG or weather data.

FTS-Diffusion decomposes the generation of realistic synthetic data into three interconnected modules:

Pattern Recognition. The first module employs a novel *Scale-Invariant Subsequence Clustering* (SISC) algorithm, which segments time series into variable-length segments and clusters them into K scale-invariant patterns. These patterns preserve shape similarity across different durations and magnitudes by using dynamic time warping (DTW) as the core distance metric:

$$\text{DTW}(x, y) = \min_{A \in \mathcal{A}} \langle A, \Delta(x, y) \rangle, \tag{27}$$

where $\mathcal{A}$ is the set of valid alignments and $\Delta(x, y)$ is the pointwise distance matrix between sequences x and y.

Pattern Generation. The second module generates new segments from learned patterns using a combination of a *scaling autoencoder* and a *pattern-conditioned diffusion network*. The diffusion model gradually corrupts a latent representation with Gaussian noise and learns to reverse the process:

$$q(x_i|x_{i-1}) = \mathcal{N}(x_i; \sqrt{1-\beta}(x_{i-1} - p), \beta I), \tag{28}$$

$$p_\theta(x_{i-1}|x_i) = \mathcal{N}(x_{i-1}; \mu_\theta(x_i, i, p), \beta I), \tag{29}$$

where p is the reference pattern and μ_θ is a neural network estimating the denoising direction. The training loss includes reconstruction loss for the autoencoder and ELBO-like diffusion loss:

$$\mathcal{L}(\theta) = \mathbb{E}[\|x_m - \hat{x}_m\|_2^2] + \mathbb{E}[\|\epsilon - \epsilon_\theta(x^i, i, p)\|_2^2]. \tag{30}$$

Pattern Evolution. The third module models the temporal transitions between patterns using a Markov chain in a learned latent space. Each state (p, α, β) corresponds to a pattern with duration α and magnitude β, and the next state is predicted as:

$$(p_{m+1}, \alpha_{m+1}, \beta_{m+1}) = \phi(p_m, \alpha_m, \beta_m), \tag{31}$$

where ϕ is a neural network trained to minimize cross-entropy and MSE losses across pattern and scaling transitions.

Experimental Results. FTS-Diffusion was evaluated on S&P500, GOOG, and corn futures datasets. It significantly outperformed existing generative baselines such as RCGAN, TimeGAN, and CSDI in both data fidelity and predictive utility. When used for data augmentation, synthetic series from FTS-Diffusion reduced LSTM-based prediction error by up to 17.9%. It also preserved stylized facts like fat-tailed return distributions and slowly decaying autocorrelation.

This approach represents the first dedicated framework capable of capturing irregular and scale-invariant patterns in financial data. By decomposing the generation task into recognition, synthesis, and temporal evolution, FTS-Diffusion not only enhances prediction tasks under data constraints but also sets a foundation for more robust simulation and backtesting in financial applications.

3 Implementation and Evaluation

3.1 Kalman and Particle Filter

Figure 1 shows the Microsoft (MSFT) closing stock prices from 2011 to 2013, along with the estimates generated using Kalman and Particle filters. In both subplots, the blue line represents the actual closing price, while the red line shows the filtered estimate. The Kalman filter (top) closely follows the actual price trend, with a mean squared error (MSE) of 0.0333 and a root mean squared error (RMSE) of 0.1825. The Particle filter (bottom) produces an even more accurate estimate in this case, with a lower MSE of 0.0211 and RMSE of 0.1454, suggesting better performance in capturing the underlying dynamics, particularly during volatile or nonlinear periods. However, it is important to note that Particle filters do not universally outperform Kalman filters; in systems that are linear and governed by Gaussian noise, the Kalman filter remains optimal and more computationally efficient.

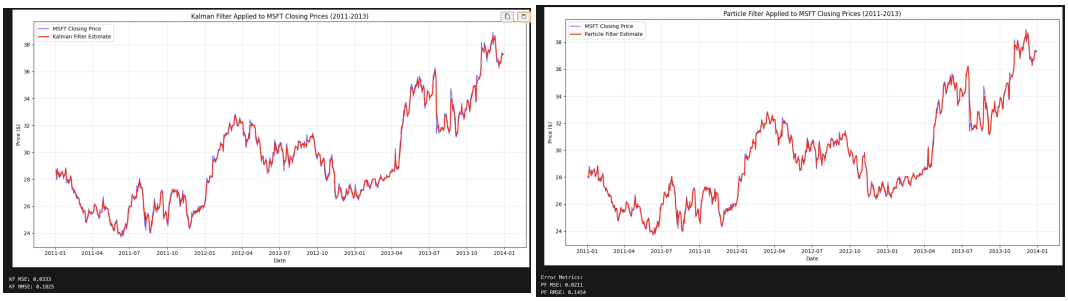

Figure 1: Result of Kalman and Particle filter

3.2 ARMA and ARIMA

Figure 2 shows the Microsoft (MSFT) closing stock prices from 2011 to 2013, along with the estimated values generated using an ARIMA(2,1,2) model. The actual closing prices are plotted in blue, while the ARIMA estimates are shown in green. Although the ARIMA model captures the general trend and cyclical behavior of the time series, its performance is less precise compared to the Kalman and Particle filters, especially during periods of rapid price movement or structural change. This is reflected in the higher error metrics, with a mean squared error (MSE) of 0.19 and a root mean squared error (RMSE) of 0.43. These results suggest that while ARIMA can be a useful baseline model for financial time series, its linear assumptions may limit its accuracy in highly volatile or nonlinear environments.

The choice of ARIMA over ARMA is motivated by the nonstationary nature of financial time series such as stock prices. ARMA models assume that the time series is stationary, meaning its mean

and variance remain constant over time. However, stock prices often exhibit trends and changing levels that violate this assumption. ARIMA models incorporate a differencing step, represented by the "I" component for "Integrated," which transforms a nonstationary series into a stationary one by subtracting previous values. In this case, using a differencing order of one helps to stabilize the mean of the series, making ARIMA more suitable for modeling and forecasting the underlying price behavior.

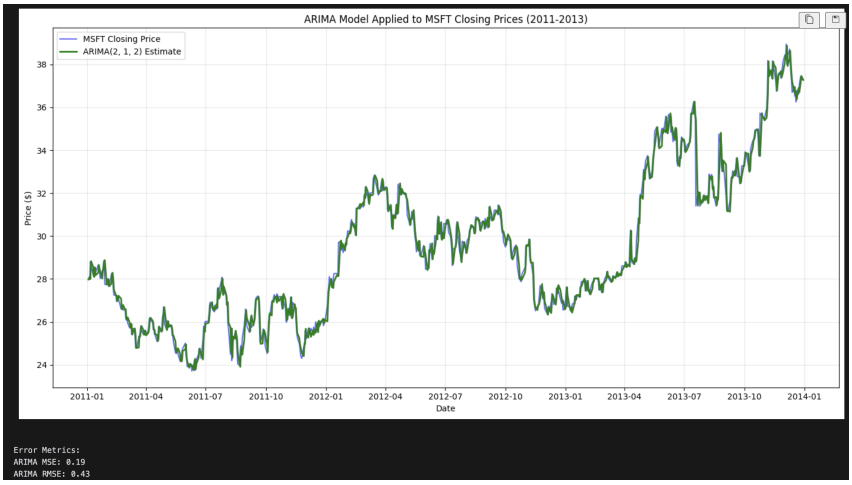

Figure 2: Result of ARIMA

3.3 RNN

Figure 3 shows the training and validation loss of the vanilla RNN model, as well as its prediction on the test set of MSFT closing prices. The training converges quickly with both losses stabilizing near zero, indicating no overfitting. However, the prediction plot demonstrates that the RNN underperforms in terms of accuracy, especially in capturing upward trends. The final test MSE is 2.0962 and the RMSE is 1.4478, the highest among all tested models.

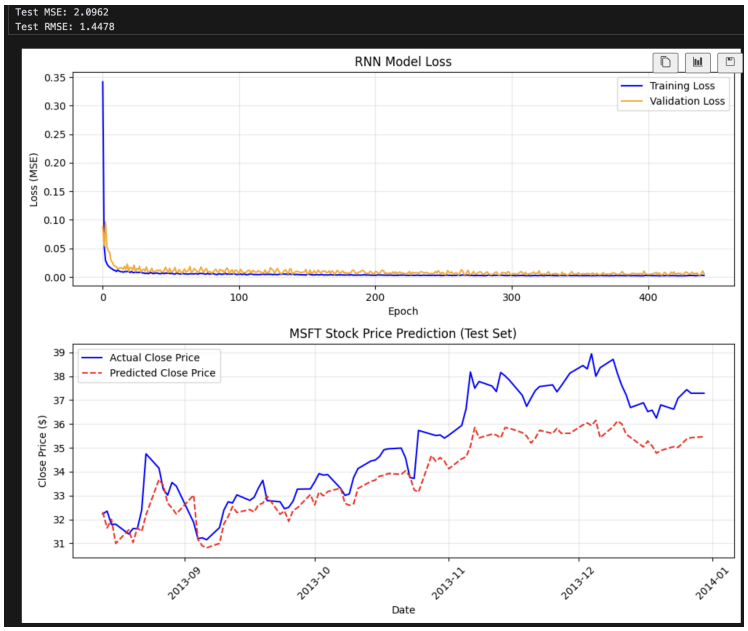

Figure 3: Result of Vanilla RNN

3.4 LSTM

Figure 4 presents the training results and predictions of the LSTM model. Compared to the RNN, the LSTM exhibits better generalization, with both training and validation losses decreasing more smoothly. On the test set, the model captures the overall trend more accurately but still fails to track some of the sharp increases in price. The LSTM achieves a lower test MSE of 1.6283 and an RMSE of 1.2761, indicating improved performance over the vanilla RNN.

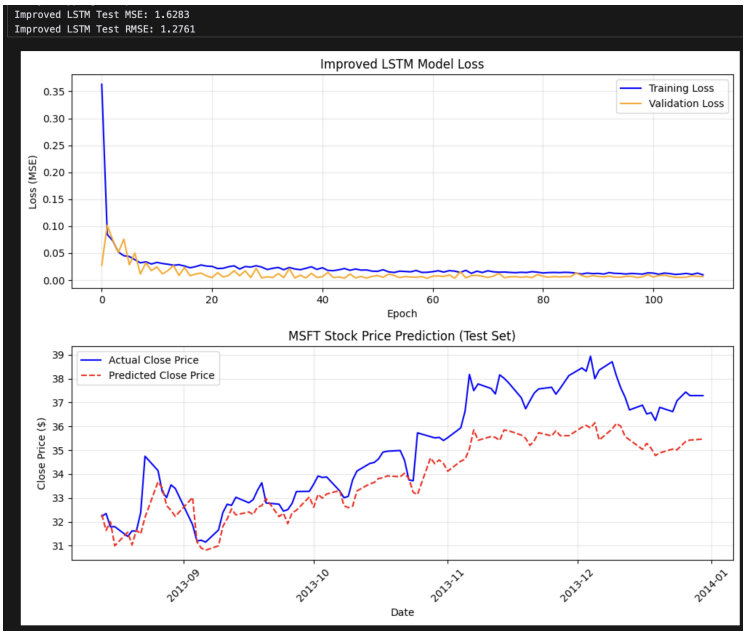

Figure 4: Result of Vanilla LSTM

3.5 GRU

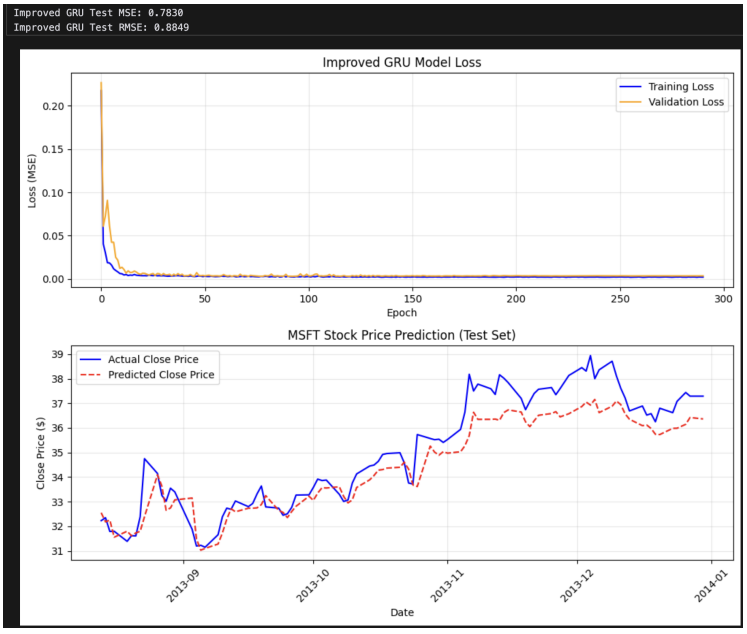

Figure 5: Result of Vanilla GRU

Figure 5 illustrates the loss curves and test predictions of the GRU model. The GRU achieves fast convergence and maintains stable training and validation losses. On the test set, the GRU closely follows the actual stock price movements, especially in regions with rapid fluctuations. Among the three models, GRU performs the best, with the lowest test MSE of 0.7830 and RMSE of 0.8849, suggesting that it captures temporal dependencies in the financial time series more effectively than RNN or LSTM.

3.6 Sentiment Analysis with FinBert

Figure 6 shows the result of integrating sentiment analysis with stock price prediction by combining FinBERT and the GRU model. Based on the previous comparison of deep learning architectures, the GRU model demonstrated the best performance in predicting MSFT stock prices, achieving the lowest test RMSE among all tested models. Therefore, we selected GRU as the base model to evaluate the impact of incorporating sentiment information.

The plot below displays the training and validation loss curves, both of which decrease smoothly and remain stable throughout the training process, indicating good convergence and generalization. The lower plot compares the actual and predicted closing prices on the test set. With the addition of sentiment features extracted from FinBERT, the model captures price movements more effectively, particularly around sharp changes and local fluctuations.

This improvement is quantitatively supported by a test MSE of 0.3710 and RMSE of 0.6091, which represent significant gains over the standalone GRU model. These results demonstrate that incorporating sentiment signals from financial news or text sources can substantially enhance the predictive accuracy of time series models in stock forecasting.

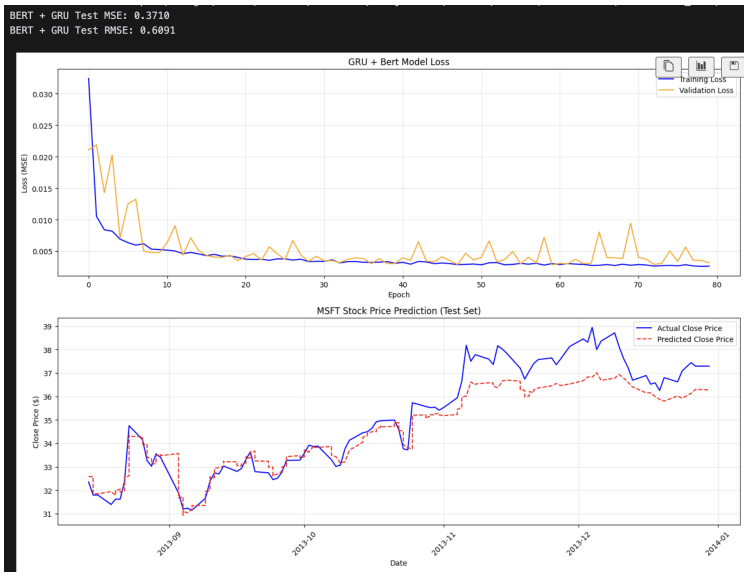

Figure 6: Result of GRU with the improvement of sentiment analysis

3.7 Comparison of Machine Learning Models

Figure 7 shows the overall comparison of the machine learning models evaluated in this study based on their test RMSE values. The plot summarizes the performance of traditional models such as ARIMA, neural network-based models including vanilla RNN, LSTM, and GRU, as well as the sentiment-enhanced GRU model with FinBERT.

Among the baseline models, GRU achieves the best accuracy, outperforming both RNN and LSTM due to its more efficient gating mechanism that captures long-term dependencies in time series data. Furthermore, when sentiment analysis is incorporated using FinBERT, the GRU model sees a significant boost in predictive performance, achieving the lowest RMSE overall. This demonstrates the value of combining financial textual sentiment with temporal price data.

The comparison highlights the progression from statistical to deep learning models and finally to hybrid models, underscoring how each layer of complexity contributes to improved forecasting accuracy in financial time series.

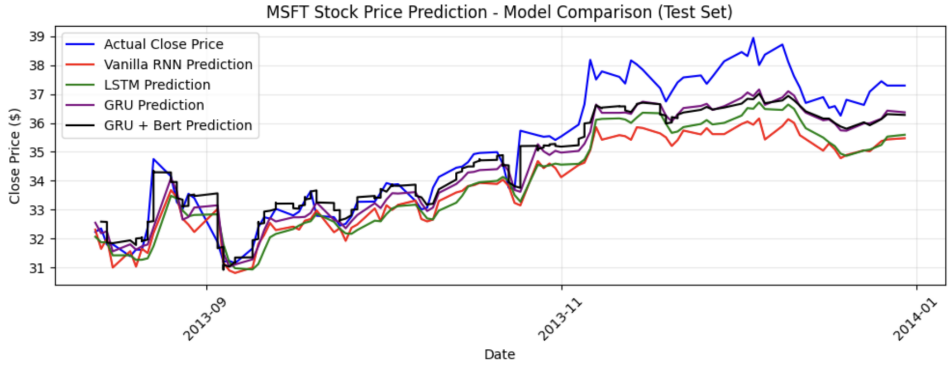

Figure 7: Overall comparison of ML models

3.8 RT-GCN

Figure 8 shows the structure of RT-GCN (Relational-Temporal Graph Convolutional Network), which integrates both relational and temporal information to model complex dependencies in stock price movement. The model first constructs a relation-temporal graph that captures interactions between different stocks over time, augmented by external resources such as news or metadata.

Within the RT-GCN framework, two types of convolutions are applied: relational convolution captures structural relationships among stocks at a single time slice, while temporal convolution extracts evolving patterns across time. These relational-temporal features are then pooled and passed through a prediction layer to generate ranking scores for investment decisions or trend forecasts.

This architecture is particularly suited for financial forecasting tasks involving multiple correlated assets. The implementation of RT-GCN is publicly available at `https://github.com/zhengzetao/RTGCN`.

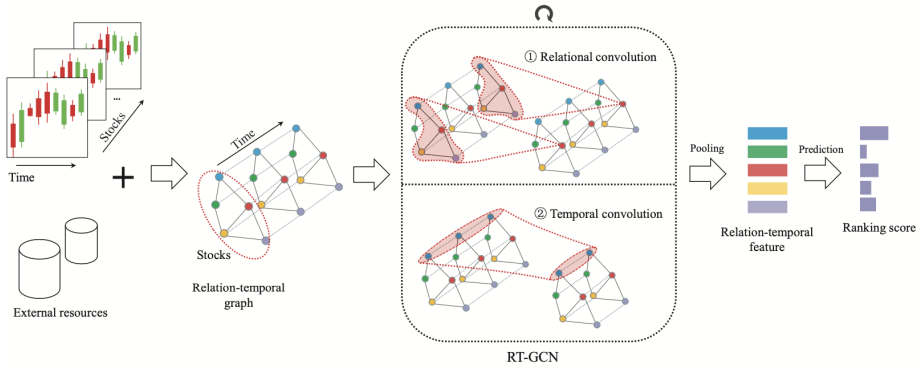

Figure 8: Structure of RT-GCN

3.9 FTS-Diffusion

Figure 9 shows the structure of FTS-Diffusion, a generative framework designed for synthesizing financial time series with irregular and scale-invariant characteristics. The architecture consists of two main modules: the *Pattern Recognition Module* and the *Pattern Generation Module*.

In the Pattern Recognition Module, the input time series $\mathbf{X}$ is first segmented using the Scale-Invariant Shapelet Clustering (SISC) algorithm. The resulting segments are grouped into a library of patterns $\mathbf{P}$, which captures recurring temporal motifs in the financial data.

The Pattern Generation Module then generates synthetic sequences through two sub-networks. The pattern evolution network ϕ predicts the next pattern $\mathbf{p}$ along with its associated duration α and magnitude β. The pattern generator network θ then produces a synthetic subsequence $\hat{\mathbf{x}}$ from these components. A sampling step combines these to generate the final synthetic time series $\hat{\mathbf{X}}$.

FTS-Diffusion effectively preserves the irregularity and scale variability of real financial series. However, the official implementation of the model is not publicly available.

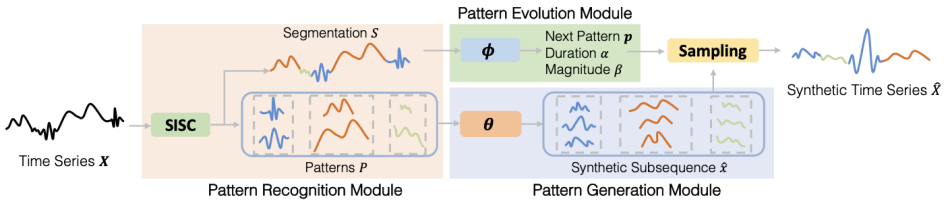

Figure 9: Structure of FTS-Diffusion

4 Conclusion

This project explored a comprehensive spectrum of modeling techniques for financial time series analysis, ranging from traditional statistical filters to advanced deep learning and generative frameworks. Initially, we benchmarked classical models including the Kalman filter, particle filter, and ARIMA. These models, while interpretable and computationally efficient, demonstrated limitations in capturing nonlinear dynamics and sudden structural shifts in financial markets.

To overcome these constraints, we implemented recurrent neural networks such as vanilla RNNs, LSTMs, and GRUs. Among them, the GRU model offered the best trade-off between accuracy and training efficiency, effectively modeling temporal dependencies in stock price sequences. Building on this foundation, we further enhanced the GRU by incorporating sentiment features from FinBERT, which significantly improved predictive performance and showcased the potential of integrating textual signals into quantitative models.

Beyond sequence models, we explored relational and structural learning with RT-GCN, which leverages both temporal and inter-stock relations through graph convolutions. This approach aligned closely with real-world investment scenarios, where asset movements are rarely independent. Finally, we addressed data sparsity and pattern irregularity with the FTS-Diffusion model, which synthesized realistic financial series using a combination of pattern recognition, diffusion generation, and evolution modeling. Although its implementation is not yet public, its design offers a compelling direction for financial data augmentation and simulation.

In summary, this study highlights a key trend in modern financial modeling: the shift from linear, independent assumptions to integrated, structure-aware, and context-sensitive methods. Future work could focus on combining these paradigms into unified forecasting pipelines, exploring multi-modal data sources, and applying reinforcement learning to optimize investment strategies based on these rich representations.

References

[1] M. G. Sousa, K. Sakiyama, L. de Souza Rodrigues, et al., "BERT for stock market sentiment analysis," in 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), IEEE, 2019, pp. 1597–1601

[2] Z. Zheng, J. Shao, J. Zhu, and H. T. Shen, "Relational temporal graph convolutional networks for ranking-based stock prediction," in Proc. IEEE 39th Int. Conf. Data Eng., 2023, pp. 123–136.

[3] Huang, H., Chen, M., & Qiao, X. (2024). Generative Learning for Financial Time Series with Irregular and Scale-Invariant Patterns. In Proceedings of the International Conference on Learning Representations (ICLR).