Comparison Among GRU, LSTM, and Arima/Garch Hybrid Models to Predict Stock Price

Hongzhen Xu, Jingming Cheng, Che-Yi Wu

Khoury College of Computer and Information Science

Northeastern University

Boston, MA

xu.hongzh, cheng.jingm,wu.chey@northeastern.edu

Dec 4th, 2024

I. Objective and Significance

The primary objective of this project was to predict stock prices using advanced machine learning and hybrid models, including GRU, LSTM, GRU-GARCH Hybrid, LSTM-ARIMA Hybrid, and LSTM-GARCH Hybrid models. Stock price prediction is a challenging task due to the inherently volatile and dynamic nature of financial markets, which are influenced by a wide range of factors, from macroeconomic indicators to psychological market behaviors. The goal was to analyze the effectiveness of these models under different scenarios, identify their unique use cases, and provide insights into their strengths and limitations in stock price prediction tasks.

The motivation for this project stemmed from the increasing reliance on algorithmic trading and the demand for robust predictive models in the financial sector.

Traditional statistical methods often struggle to capture the nonlinear patterns and temporal dependencies present in financial data. Machine learning and hybrid approaches, on the other hand, offer promising solutions by leveraging historical data and integrating statistical and deep learning techniques. By exploring a diverse set of models, this project aimed to contribute to the understanding of how these methods perform in real-world financial scenarios.

Our findings revealed that each model has specific scenarios where it excels. For instance, the GRU and LSTM models were effective in capturing temporal dependencies, while the hybrid models demonstrated strengths in incorporating statistical trends and volatility patterns. The GRU-GARCH Hybrid model was particularly useful in scenarios requiring an understanding of volatility dynamics, while the LSTM-ARIMA and LSTM-GARCH models were better suited for datasets with strong seasonality or heteroskedasticity. These results highlight the importance of selecting the right predictive model based on the characteristics of the stock data and the specific requirements of the application.

II. Background

(a)Introduction to Key Concepts and Background Information

GRU (Gated Recurrent Unit):
GRU is a type of recurrent neural network (RNN) architecture designed to handle sequential data by addressing the vanishing gradient problem commonly found in standard RNNs. It achieves this using gating mechanisms—reset gates and update gates—to control the flow of information. GRUs are computationally efficient and often preferred for time-series tasks, including stock price prediction, due to their ability to capture temporal dependencies in data without requiring excessive computational resources.

LSTM (Long Short-Term Memory):
LSTM networks are another RNN variant that effectively models long-term dependencies in sequential data. They achieve this through a more complex gating mechanism consisting of forget gates, input gates, and output gates. Unlike GRUs, LSTMs maintain separate memory cells, which allow them to learn when to "forget"

or "remember" information over long sequences. This makes LSTMs highly effective for tasks requiring intricate temporal relationships, such as stock price forecasting.

ARIMA (Autoregressive Integrated Moving Average):
ARIMA is a widely used statistical model for analyzing and forecasting time-series data. It combines three components:

1. **Autoregressive (AR):** This component uses past values of the time series to predict future values, assuming a linear relationship between current and past observations.
2. **Integrated (I):** This involves differencing the data to make it stationary (i.e., to remove trends or seasonality), a prerequisite for effective modeling.
3. **Moving Average (MA):** This component models the dependency between an observation and a residual error from previous time steps.

ARIMA is highly effective for capturing linear patterns and trends in time-series data, making it a classic choice for financial applications like stock price prediction. However, ARIMA struggles with nonlinear dependencies and cannot effectively model heteroskedasticity (variance that changes over time).

GARCH (Generalized Autoregressive Conditional Heteroskedasticity):
GARCH is a statistical model designed to address time-series data with changing variance over time, a property known as heteroskedasticity. It extends the Autoregressive Conditional Heteroskedasticity (ARCH) model by adding a lagged conditional variance term. The GARCH model predicts not only the mean of a time series but also its variance, which is particularly important in financial contexts where volatility is a critical factor.

The GARCH model consists of two components:

1. **Conditional Variance Equation:** Captures the time-varying volatility by considering past variance (GARCH terms) and past squared residuals (ARCH terms).
2. **Mean Equation:** Often modeled with an ARIMA process to capture trends in the data.

GARCH models are especially popular in finance for forecasting volatility, as they can adapt to the "clustered volatility" often observed in stock markets, where high-volatility periods are followed by high-volatility periods and low-volatility periods follow low-volatility periods.

Application in Financial Forecasting

Stock price prediction is inherently challenging due to the stochastic and non-stationary nature of financial data. Both GRU and LSTM models are well-suited for this domain because they can learn patterns in the sequential data, such as trends and seasonality, without the need for extensive feature engineering.

ARIMA is often used for predicting stock price levels and trends, while GARCH is employed to model and predict volatility, a key measure of risk. The combination of ARIMA for the mean process and GARCH for the variance process has been effective in modeling financial time series. However, both models assume stationarity in their respective components and have limitations when handling complex nonlinear patterns or long-term dependencies, areas where machine learning models like GRU and LSTM excel.

By integrating ARIMA and GARCH with machine learning methods (e.g., in LSTM-ARIMA or GRU-GARCH hybrids), we can combine the strengths of statistical models for capturing trends and volatility with the flexibility of neural networks to handle nonlinear and long-term relationships in financial data.

(b)Previous Works on this Problem

Several studies have explored the application of GRU, LSTM, and their hybrid models with statistical approaches like ARIMA and GARCH for stock price prediction. These works provide valuable insights into the effectiveness of combining deep learning architectures with traditional time-series models.

In the paper "Stock Price Correlation Coefficient Prediction with ARIMA-LSTM Hybrid Model" by Hyeong Kyu Choi (2018) and the paper "Stock Price Prediction using LSTM-ARIMA Hybrid Neural Network Model" (2023), they made predictions based on Hybrid Models Combining LSTM and ARIMA.

In the paper "A Combined Model of ARIMA-GRU to Forecast Stock Price" by Sangeeta Saha et al. (2021), it created and tested Hybrid Models Combining GRU and ARIMA.

In the paper "A Stock Index Futures Price Prediction Approach Based on the MULTI-GARCH-LSTM Mixed Model" by Haojun Pan et al. (2024), it created and tested Hybrid Models Combining LSTM and GARCH.

In the paper "Comparative Analysis of LSTM, GRU, and Transformer Models for Stock Price Prediction" by Jue Xiao et al. (2024), it did a Comparative Analysis of LSTM, GRU and Transformer Models for predicting stock price.

These studies collectively demonstrate the potential of hybrid models that integrate deep learning architectures like LSTM and GRU with traditional statistical methods such as ARIMA and GARCH. By capturing both linear and nonlinear patterns, as well as addressing volatility clustering in financial time-series data, these models offer enhanced predictive capabilities for stock price forecasting.

(c) What Makes This Work Particularly Interesting?

Our work stands out by directly comparing GRU and LSTM models in the same prediction task and extending the analysis to hybrid models. While previous research often focuses on either model in isolation or lacks a comprehensive evaluation of their use cases, our project seeks to:

1. Highlight the specific scenarios where each model excels, providing actionable insights for practitioners.
2. Incorporate hybrid architectures like GRU-GARCH and LSTM-ARIMA, exploring how they handle data characteristics such as volatility, seasonality, and heteroskedasticity.
3. Address the gap in understanding the trade-offs between computational efficiency and predictive accuracy for GRU and LSTM in stock price prediction.

By focusing on these aspects, our work contributes to the growing body of knowledge in financial forecasting, offering a practical guide for choosing the right model based on data and application requirements.

III. Methods

(a)Data Source and Preprocessing

We get daily data of Open Price, Close Price, High Price, Low Price, Volume for stocks from https://www.alphavantage.co/.

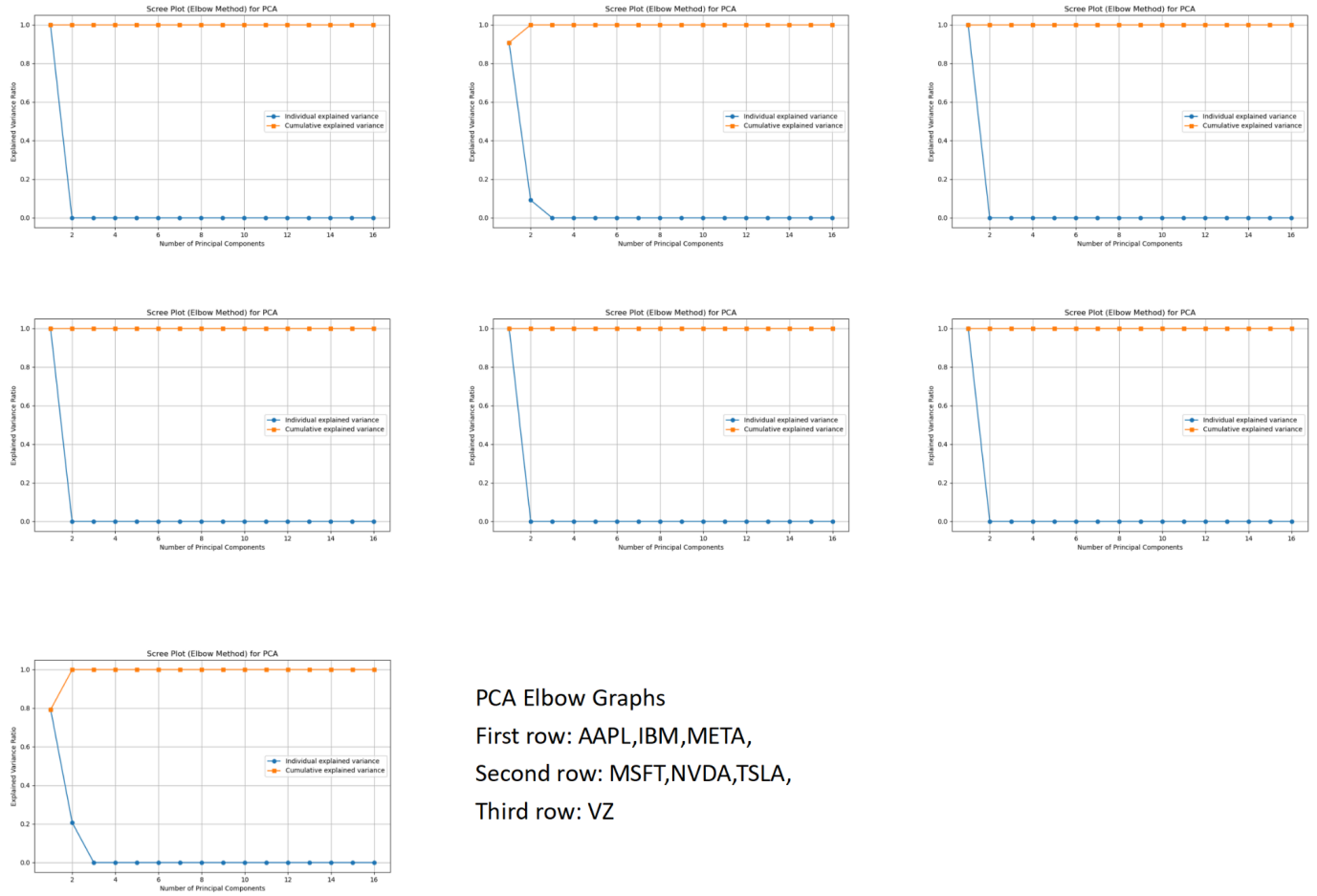
We downloaded data from https://stockanalysis.com , a website with every stock data. The website does not provide daily financial ratios and metrics in history, but we get the quarterly metrics and ratios for ten years for 7 stocks: Apple(AAPL), IBM(IBM), Nvidia(NVDA), Verizon(VZ), Meta(META), Microsoft(MSFT), Tesla(TSLA).

We choose the following metrics and ratios:

1. Market Capitalization,
2. Market Cap Growth,
3. Enterprise Value,
4. PE Ratio,
5. PS Ratio,
6. PB Ratio,
7. P/FCF Ratio,
8. P/OCF Ratio,
9. Debt/Equity,
10. Quick Ratio,
11. Current Ratio,
12. Return on Invested Capital (ROIC),
13. Dividend Yield,
14. Payout Ratio,
15. Buyback Yield,
16. Total Return.

We draw PCA elbow graphs and determine to reduce the above 16 metrics and ratios into 2 Principal Components for each stock. By using the PCA elbow graph, we found that most stocks need only two principal components. Further, we

extracted these principal components using PCA implementation. Then, combine basic daily data with PCA results and reverse the data based on date.(so that the further past data comes first in the input.)



PCA Elbow Graphs
First row: AAPL,IBM,META,
Second row: MSFT,NVDA,TSLA,
Third row: VZ

The finalized data input is in the file named “Reversed_{XXXX}_Data_with_PCA.csv” where {XXXX} is the stock ticker symbol.

The columns represent:

date,open,high,low,close,volume,Principal_Component_1, Principal_Component_2

(b)Methodology

1. **GRU Rate Based Stock Predict Model**

   For purposes of using GPU computing power, we wrote and ran codes on Google Colab.

   We splitted the 10 years data into a training set and a testing set by 8:2 ratio.

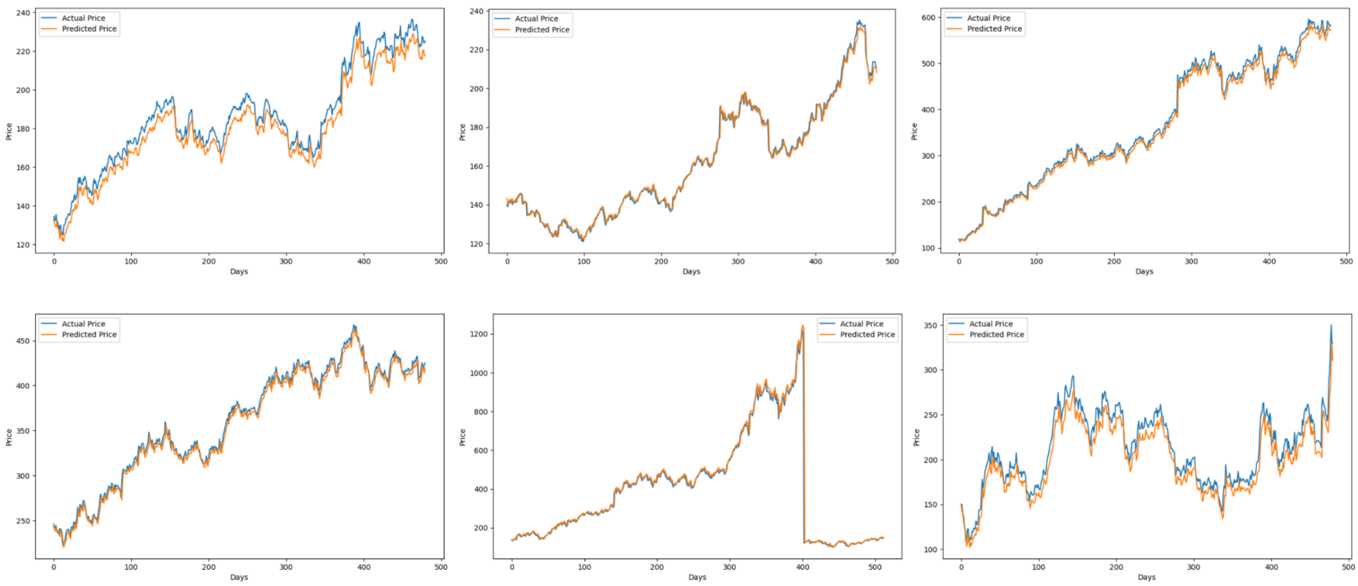
   We normalized the data by MinMaxScaler.

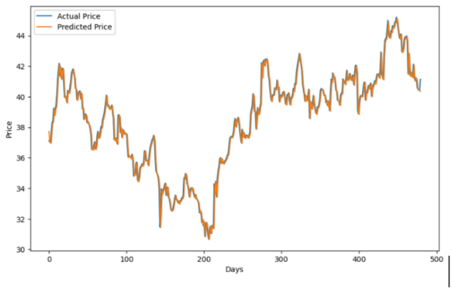
We set the sequence length to be 30 days, and use a sliding window to predict the next day's stock price iteratively. The batch size is 32.
The GRU model parameters are: input_size=7, hidden_size=500, num_layers=2, output_size=1.
The input size is the number of features for each sequence. The hidden size is the hidden state size for GRU memory. The num layers is the number of layers of hidden state. The output size is 1 because we only need the ratio.
The procedure is as follows:
Go through the GRU module, then go through the fully connected layers with Relu activation and 0.3 dropout at each hidden layer.
In early attempts, we found that the "close" stock price itself is not a good target for training because recent years technology industry booming cannot be learned in the historical data. Typically, some stocks reached their highest peak in the recent two years. The prediction on the testing data set shows a horizontal trend after reaching the maximum price in the training data set. Thus, instead of directly predicting the next day's stock price, we choose to use the ratio between tomorrow and today's stock price as the target for training. After each prediction, we multiply the predicted ratio by today's stock price and get a prediction for tomorrow's stock price. Then apply the MSE loss function on the absolute gap between tomorrow's actual price and our model's predicted price. Then back propagate through the model in 50 training epochs. The optimizer is Adam with learning rate 0.001 and weight_decay 1e-4.
After testing on the testing data set, we saved files with dates, actual prices, predicted prices for comparison and evaluation. We also draw plots comparing actual prices and predicted prices over the testing period. The trained model is not saved because of the immense size.

2. **LSTM Rate Based Stock Predict Model**

   The setup and procedures are the same as GRU Rate Based Stock Predict Model. The only difference is that LSTM has two variables for hidden state, h0 and c0, representing short-term and long-term memory.

   Actual Price(blue line) vs Predicted Price(orange line) during almost 500 days testing period are plotted below:



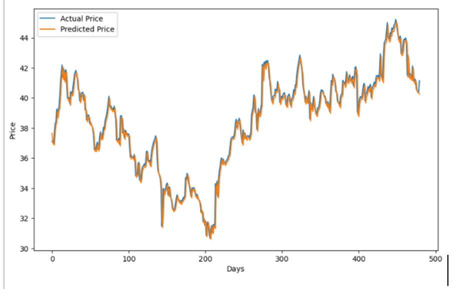


GRU rate Graphs

First row: AAPL,IBM,META,

Second row: MSFT,NVDA,TSLA,

Third row: VZ



LSTM rate Graphs

First row: AAPL,IBM,META,

Second row: MSFT,NVDA,TSLA,

Third row: VZ

3. **Arima-LSTM Hybrid Stock Predict Model**

3.1 Arima Fitting:

Data splitting technique is similar to those used in the GRU model.

The process of fitting an ARIMA model involves selecting the appropriate order (p, d, q), estimating the model parameters, and examining the model residuals.

We firstly applied the Ljung-Box test on data to check if autocorrelation exists in a time series by calculating Augmented Dickey-Fuller (ADF), p-value, critical values.

Take stock VZ as an example,

Before Differencing:

- ADF Statistic: **-1.3673,** indicates the test statistic for the null hypothesis that the series is non-stationary. Higher values closer to 0 imply a stronger trend or seasonality.
- p-value: **0.5979,** greater than 0.05 suggests that we fail to reject the null hypothesis, meaning the series is non-stationary.
- Critical Values:
    - 1%: **-3.4329**
    - 5%: **-2.8627**
    - 10%: **-2.5674**

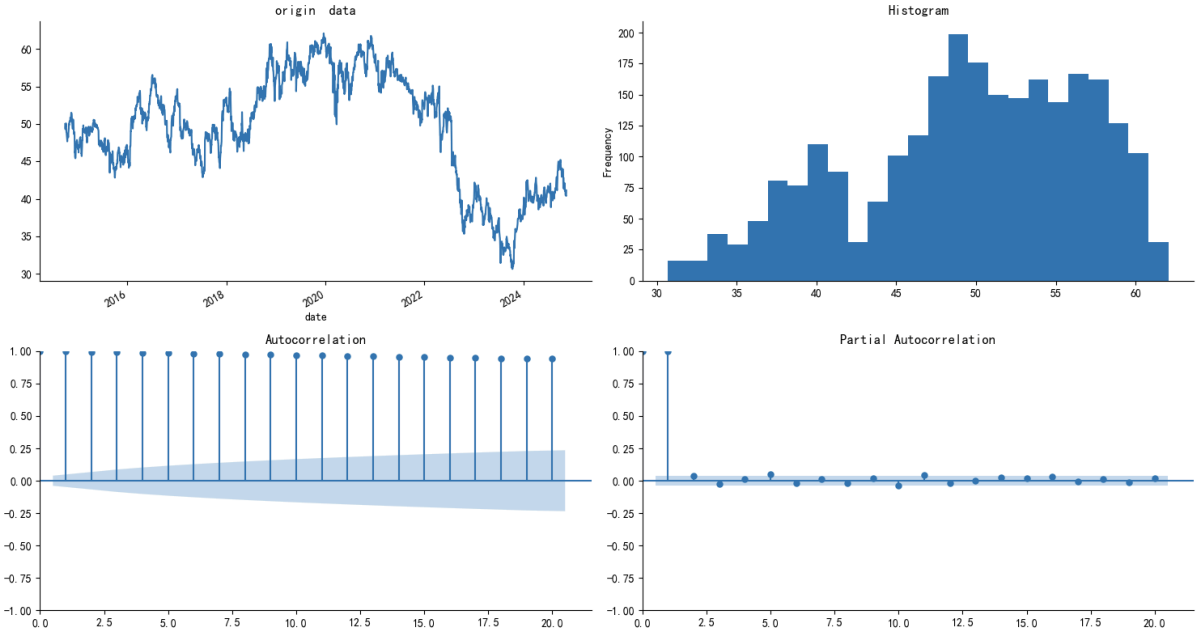
These thresholds are used to compare the test statistic (ADF) for rejecting the null hypothesis. Since -1.3673 is greater than all critical values, the time series is confirmed non-stationary.

After First Differencing:

- ADF Statistic: **-10.9723,** much smaller value indicates that differencing removed the non-stationarity.
- p-value: **7.8241e-20,** effectively 0, meaning the null hypothesis of non-stationarity is rejected.
- Critical Values:
    - 1%: **-3.4329**
    - 5%: **-2.8627**
    - 10%: **-2.5674**

The test statistic is smaller than the 1% threshold, confirming stationarity after differencing.

Then we calculate Akaike Information Criterion (AIC) to determine p、q、d of ARIMA. We can also use the auto_arima function, or determine from figures of ACF and PACF, but it's subjective.



3.2 Get Residuals:

We calculate residuals for the Arima model, then examine whether residuals follow a normal distribution. When training the LSTM model, standardization or normalization is required to ensure that the input data is within a reasonable range.

3.3 LSTM model training:

The LSTM model is used to train the residuals of the ARIMA model. Here, the residuals are used as the input sequence, and the LSTM model learns the nonlinear pattern of the residuals.The LSTM layer has 64 hidden units, randomly drops 20% of the units during training to prevent overfitting, adds a fully connected layer with one output node, representing the predicted residual, trained for 100 iterations over the dataset, processed in mini-batches of 32 samples.

Adaptive Moment Estimation (Adam) optimizer is used for efficient training.

After formatting the data (Example on stock VZ):

- X_train.shape = (508, 2, 1):
    - 508 training samples
    - Each sample contains 2 time steps (seq_len=2)
    - 1 feature per time step (residual values)
- y_train.shape = (508, 1):
    - 508 target values (residual at t)

3.4 Prediction:

Add the predicted value of the Arima model to the predicted value of the LSTM model to obtain the final prediction result of the hybrid model.

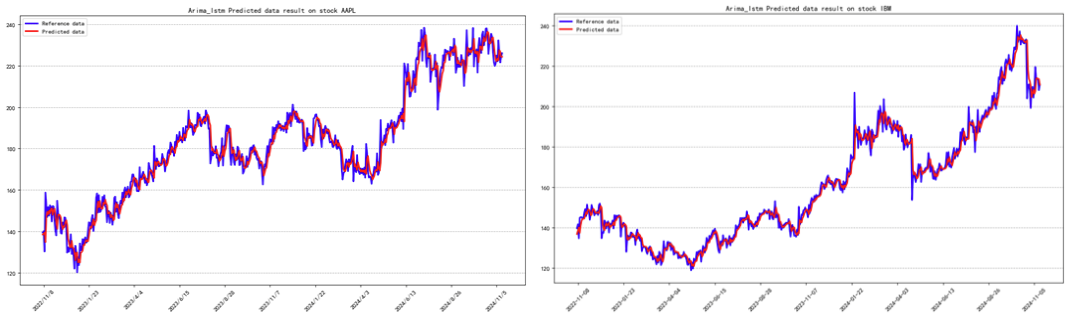
3.5 Visualization:

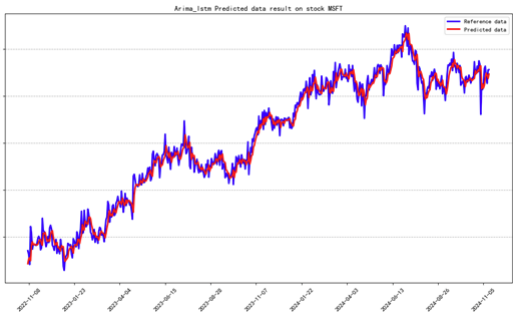
Plot the original time series, the forecasted values, and the confidence intervals. This visual representation helps communicate the model's performance and predictions effectively.

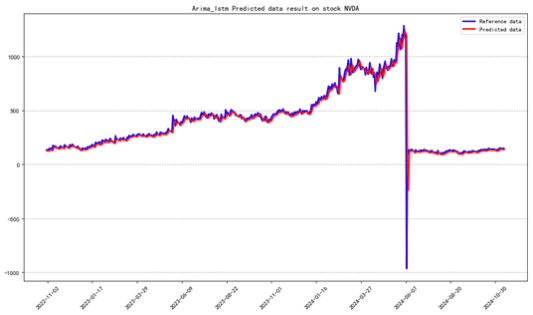
Example: result on stock VZ

- mse: 1.155
- mae: 0.754
- r2: 0.889

Plots: AAPL, IBM, META, MSFT, NVDA, TSLA, VZ







4. **Garch-GRU Hybrid Stock Predict Model**

   Similar to the GRU model, we utilized an 8:2 ratio to split the data into training and testing sets. However, our approach differs in several key aspects:

   - Applied StandardScaler instead of MinMaxScaler for better handling of volatility
   - Implemented a sequence length of 20 days (versus 30 in pure GRU) to optimize the balance between historical information and computational efficiency
   - Maintained the same batch size of 32 for consistency

   The GRU-Garched hybrid model combines two components:

   GRU:

   - Input dimension: Features + 1

- Hidden dimension: 64 (128 for NVDA)
- Number of layers: 2 (3 for NVDA)
- Output Dimension: 1

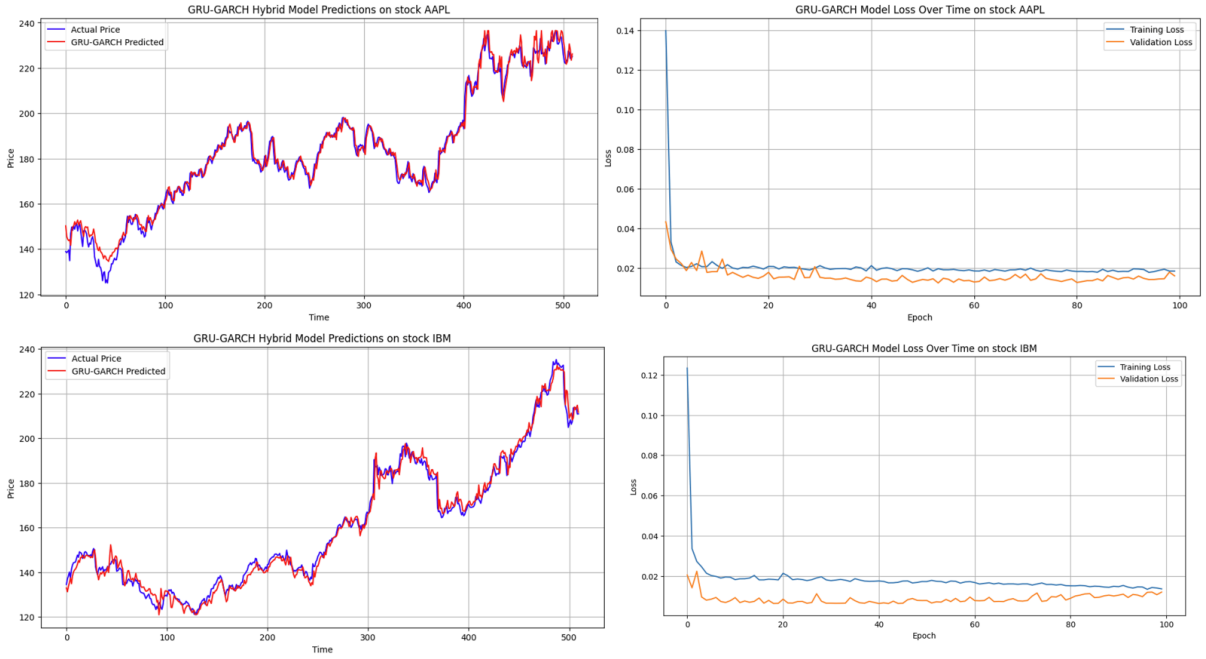
Garch(1, 1):

- omega (ω): Constant term
- alpha (α): ARCH parameter
- beta (β): GARCH parameter

Additionally, we gave a separate implementation that was necessary for NVIDIA(Garch-NVDA), due to its unique characteristics of high volatility, would cause errors when implementing GRU-Garch. We increased the hidden dimension from 64 to 128, added an extra 3 GRU layers, and reduced the learning rate from 0.001 to 0.0005 to handle the high volatility of NVDA. Additionally, we also implement gradient clipping (max_norm=1.0) and error handling to prevent training instability caused by large price movements.

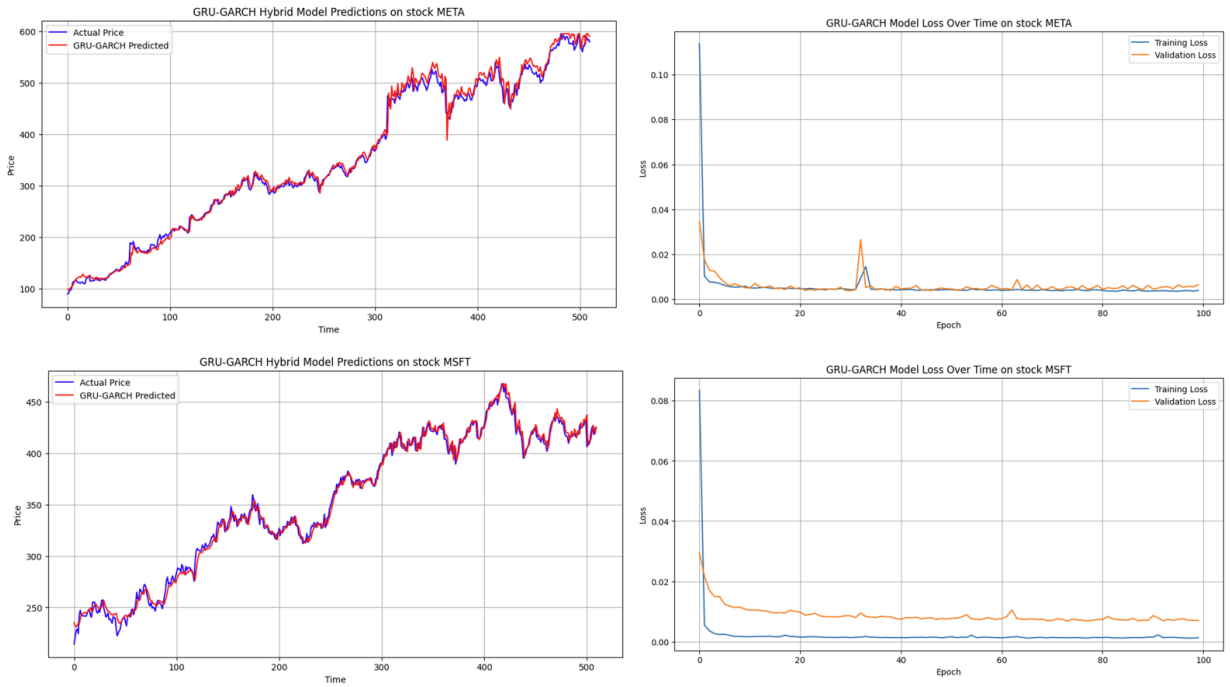
The training procedure of GRU-Garch for each epoch in each batch first process sequence through GRU, then calculate Garch volatility and combine GRU output with Garch volatility, generate final prediction, calculate MSE loss, and finally Backpropagate and update parameters.
For optimization, we used Adaptive Moment Estimation (Adam) as optimizer and Mean Squared Error(MSE) as loss function, learning rate of 0.001(0.0005 for NVDA), and training epochs of 100.

Plots:
Left: Price Prediction, Right: Training and Validation Loss
First Row: AAPL, Second Row: IBM



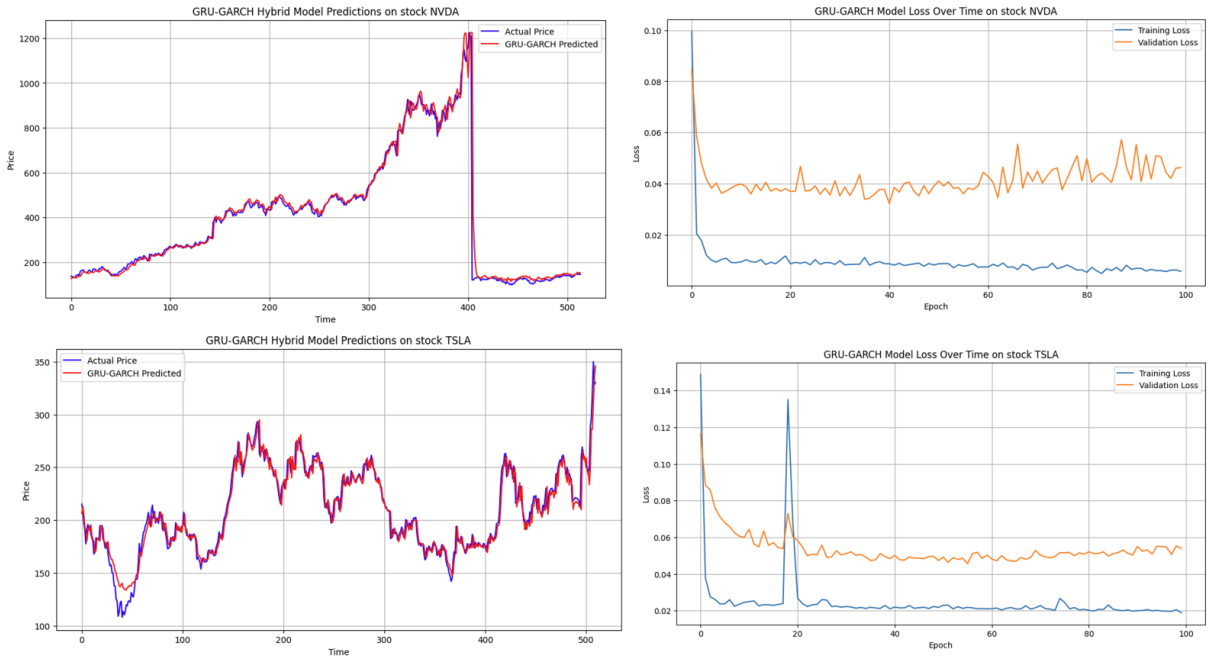
Left: Price Prediction, Right: Training and Validation Loss

First Row: META, Second Row: MSFT



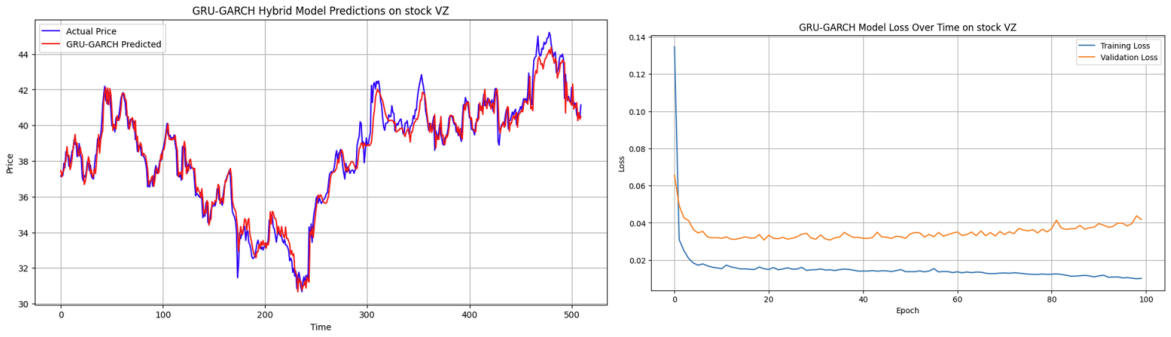
Left: Price Prediction, Right: Training and Validation Loss

First Row: NVDA, Second Row: TSLA



Left: Price Prediction, Right: Training and Validation Loss

VZ



5. **Garch-LSTM Hybrid Stock Predict Model**

Similar to the base LSTM model, we employed:

- 8:2 ratio for training/testing split
- StandardScaler for data normalization (instead of MinMaxScaler)
- Sequence length of 20 days for time series analysis
- Batch size of 32 for training efficiency

The LSTM-Garched hybrid model combines two components:

LSTM:

- Input dimension: Features + 1 (close price)
- Hidden dimension: 64

- Number of layers: 2
- Dropout rate: 0.2
- Output dimension: 1

Hidden States:

h0 = Short-term memory state

c0 = Long-term memory state

GARCH(1,1) Component:

$\sigma^2 t = \omega + \alpha(rt\text{-}1)^2 + \beta\sigma^2 t\text{-}1$

Where:

- $\sigma^2 t$ = Conditional variance at time t
- $rt\text{-}1$ = Previous period's return
- ω = Constant term (omega)
- α = ARCH parameter (alpha)
- β = GARCH parameter (beta)

The training procedure of LSTM-Garch for each batch first processes sequence through LSTM, then calculates Garch Volatility, and finally combines LSTM output with Garch volatility.

For optimization, we used ADAM as optimizer, MSE as loss function, learning rate of 0.001, weight decay 1e-5, and training epochs of 50.

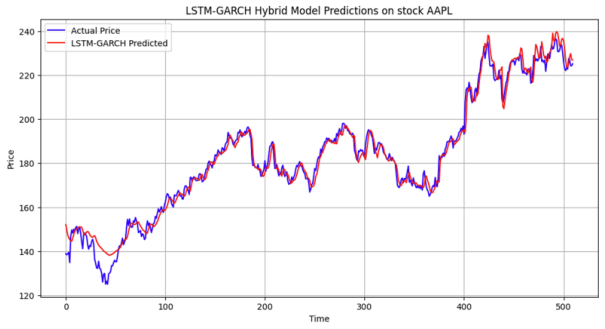
Key Differences from Base LSTM:

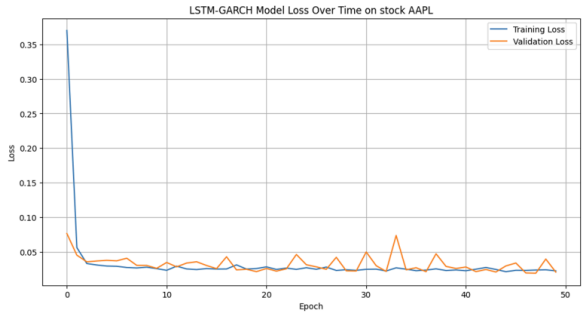
- Memory Management:
  Base LSTM: Dual memory states (h0, c0) for temporal dependencies
  LSTM-Garch: Additional volatility memory through GARCH component
- Prediction Mechanism:
  Base LSTM:
  ```
  prediction = LSTM(input_sequence)
  ```
  LSTM-Garch:
  ```
  price_prediction = LSTM(input_sequence)
  volatility = GARCH(returns)
  final_prediction = combine(price_prediction, volatility)
  ```
- Loss Calculation:
  ```
  loss = MSE(predicted_price, actual_price) + λ *
  MSE(predicted_volatility, actual_volatility)
  ```
  ,where λ is a weighting factor for volatility contribution

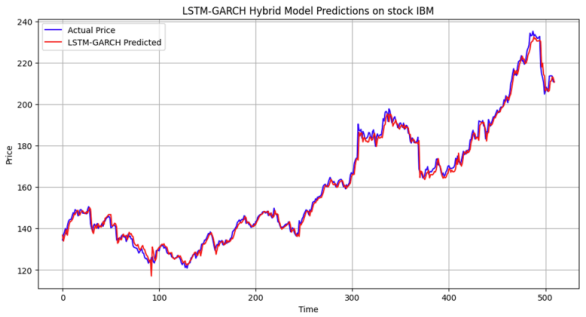
Plots:

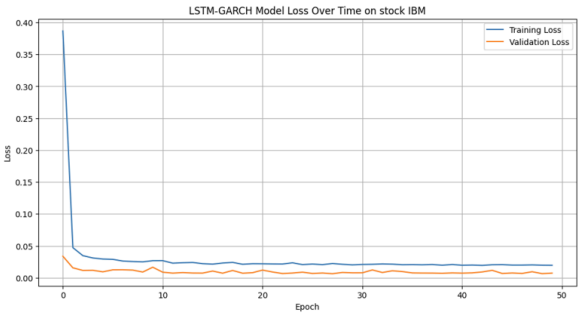
Left: Price Prediction, Right: Training and Validation Loss

First Row: AAPL, Second Row: IBM



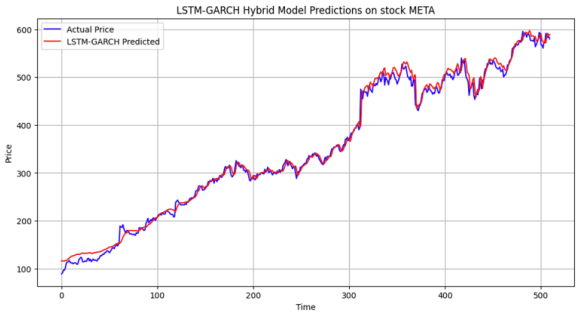


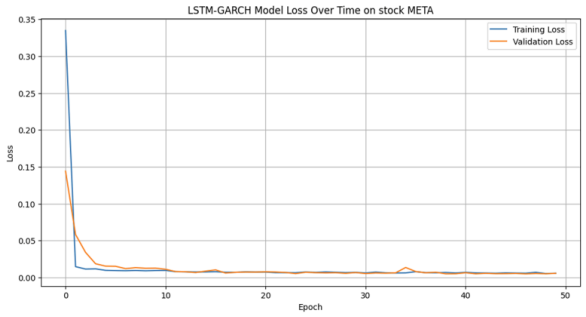


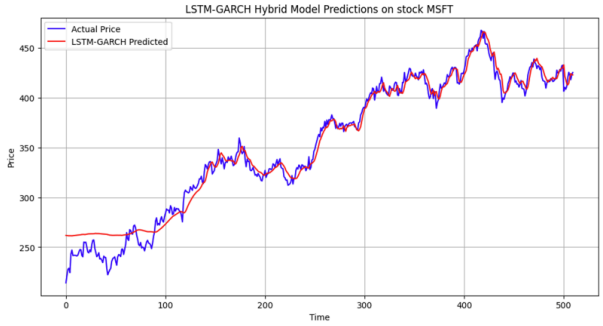


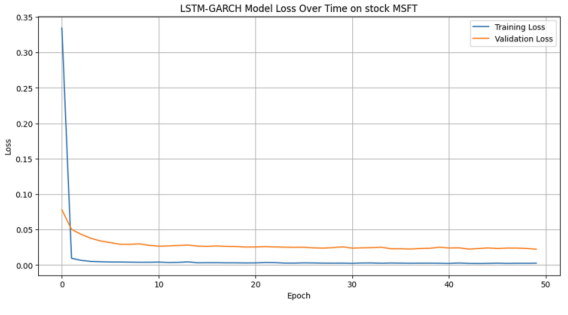
Left: Price Prediction, Right: Training and Validation Loss

First Row: META, Second Row: MSFT



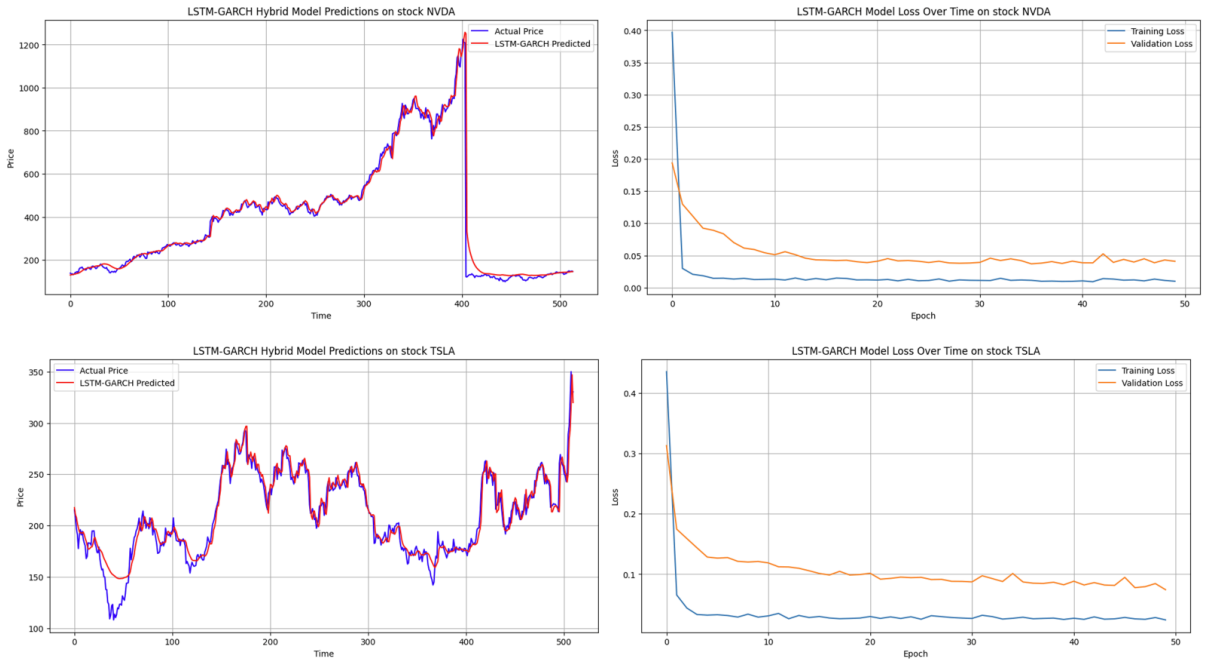






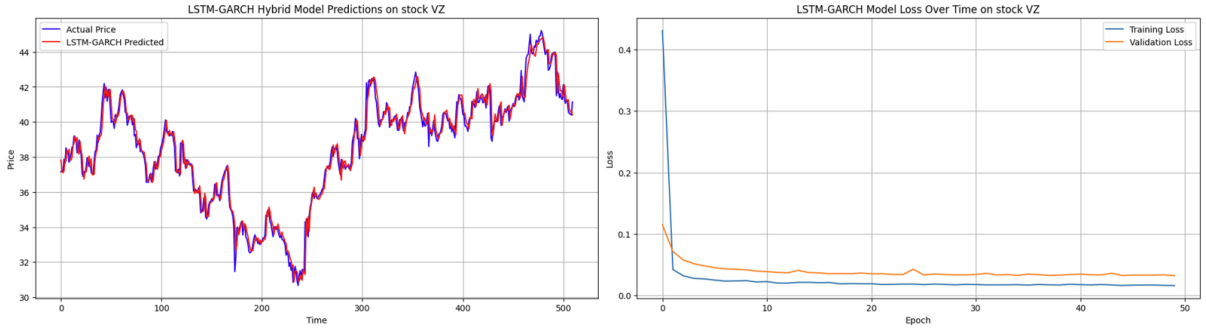
Left: Price Prediction, Right: Training and Validation Loss

First Row: NVDA, Second Row: TSLA



Left: Price Prediction, Right: Training and Validation Loss

VZ



(c) Evaluation Strategy

Basically, we use statistics approaches such as MSE, MAE, RMSE(for GRU-Garched and LSTM-Garched), R squared to evaluate the performance of predictions vs actual values. We also developed a compare group called “default” or “standard”. This compare group was constructed by using the previous day close price as the next day prediction. We have curiosity about the performance of such a simple strategy compared to our models.

1. **Mean Squared Error (MSE)**

   The formula for **Mean Squared Error (MSE)** is:

MSE = (1 / n) * Σ (y_i - y_pred_i)^2

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

Where:

- **n**: The number of observations.
- **y_i**: The actual values of the dependent variable.
- **y_pred_i**: The predicted values from the model.
- **Σ**: Summation across all observations.

Interpretation of MSE:

- The MSE measures the average squared difference between actual and predicted values.
- Smaller MSE values indicate better model performance.

2. **Mean Absolute Error (MAE)**

The formula for **Mean Absolute Error (MAE)** is:

MAE = (1 / n) * Σ |y_i - y_pred_i|

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Where:

- **n**: The number of observations.
- **y_i**: The actual values of the dependent variable.
- **y_pred_i**: The predicted values from the model.
- **Σ**: Summation across all observations.

Interpretation of MAE:

- The MAE measures the average absolute difference between actual and predicted values.
- Like MSE, smaller MAE values indicate better model performance, but MAE is less sensitive to outliers.

3. **Formula for R^2 (Coefficient of Determination)**

   The R^2 value is calculated as:

   R^2 = 1 - [Σ (y_i - y_pred_i)^2] / [Σ (y_i - y_mean)^2]

   $$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

   Where:

   - **y_i**: The actual values of the dependent variable.
   - **y_pred_i**: The predicted values from the model.
   - **y_mean**: The mean of the actual values.
   - **Σ**: Summation across all observations.

   Interpretation of R^2:

   - **R^2 = 1**: Perfect fit, meaning the model explains all the variance.
   - **R^2 = 0**: The model explains none of the variance, like always predicting the average value.
   - **R^2 < 0**: The model is worse than using the mean as the prediction.

4. **Mean Squared Error (RMSE)**

   It is the root of MSE.

IV. Results

(a) Results and Findings

1. **GRU rate stock price predict model**
   GRU shows decent results as shown in the plots above and **Table 1** below. The GRU predictions have a bigger MSE than the standard model, which means we have potential to do it better. However, prediction cannot always be fairly evaluated by such metrics like MSE and MAE, because the more important thing investors care about is the trend and volatility rather than a naive guess of 0 deviation from yesterday. The standard model has the best MSE and MAE from its definition, but it provides no useful information to the investors.
2. **LSTM rate stock price predict model**
   LSTM illustrates much better accuracy than GRU in most cases as shown in the plots and **Table 1** below. But it still loses compared to the standard strategy. The predictions are somehow a lagging index that we exploit the past information as much as we can in order to approach the actual future targets. Things will be more interesting if these models are not predicting the next day price but predicting the next month price. We believe that the standard strategy cannot win in this monthly price prediction match, because 0 deviation is anti-logic.

3. **GRU-Garch model**
   Stocks Showing Improvement with Hybrid Model(Approximate)
   Instead of using GRU rate stock prediction model
   3.1 Apple (AAPL)
   MSE: 71.2% improvement (39.84 → 11.46)
   MAE: 57.4% improvement (5.75 → 2.45)
   RMSE: 46.3% improvement (6.31 → 3.39)
   3.2 Tesla (TSLA)
   MSE: 59.9% improvement (207.55 → 83.26)
   MAE: 47.9% improvement (12.42 → 6.47)
   RMSE: 36.7% improvement (14.41 → 9.12)

3.3 Microsoft (MSFT)

MSE: 18.4% improvement (41.55 → 33.90)

MAE: 16.4% improvement (5.29 → 4.43)

RMSE: 9.7% improvement (6.45 → 5.82)

3.4 Meta (META)

MSE: 7.5% improvement (148.78 → 137.63)

MAE: 10.7% improvement (9.29 → 8.30)

RMSE: 3.8% improvement (12.20 → 11.73)

Stocks Showing Degradation with Hybrid Model

Instead of using GRU rate stock prediction model

3.5 Verizon (VZ)

MSE: 34.9% degradation (0.30 → 0.40)

MAE: 21.4% degradation (0.38 → 0.47)

RMSE: 16.1% degradation (0.54 → 0.63)

3.6 IBM

MSE: 84.8% degradation (5.32 → 9.83)

MAE: 50.8% degradation (1.57 → 2.36)

RMSE: 36.0% degradation (2.31 → 3.14)

3.7 NVIDIA (NVDA)

MSE: 7.8% degradation (2649.60 → 2856.38)

MAE: 24.8% degradation (12.23 → 15.27)

RMSE: 3.8% degradation (51.47 → 53.45)

4. **LSTM-Garch model**

Stocks Showing Improvement with Hybrid Model(Approximate)

Instead of using LSTM rate stock prediction model

4.1 Tesla (TSLA)

MSE: 14.4% improvement (158.63 → 135.77)

MAE: 25.8% improvement (10.50 → 7.79)

RMSE: 7.5% improvement (12.59 → 11.65)

4.2. Verizon (VZ)

MSE: 2.2% improvement (0.309 → 0.302)

MAE: 3.2% improvement (0.401 → 0.388)

RMSE: 1.1% improvement (0.556 → 0.550)

4.3 IBM (IBM)

MSE: 40.4% improvement (10.28 → 6.13)

MAE: 29.1% improvement (2.44 → 1.73)

RMSE: 22.8% improvement (3.21 → 2.48)

Stocks Showing Degradation with Hybrid Model

Instead of using LSTM rate stock prediction model

4.4 Microsoft (MSFT)

MSE: 339.7% degradation (26.68 → 117.30)

MAE: 97.1% degradation (3.94 → 7.77)

RMSE: 109.7% degradation (5.17 → 10.83)

4.5 Meta (META)

MSE: 58.2% degradation (104.80 → 165.75)

MAE: 35.5% degradation (7.27 → 9.86)

RMSE: 25.8% degradation (10.24 → 12.87)

4.6 NVIDIA (NVDA)

MSE: 22.6% degradation (2555.36 → 3133.57)

MAE: 72.9% degradation (11.00 → 19.00)

RMSE: 10.7% degradation (50.55 → 55.98)

4.7 Apple (AAPL)

MSE: 199.5% degradation (6.77 → 20.28)

MAE: 66.5% degradation (1.94 → 3.23)

RMSE: 73.1% degradation (2.60 → 4.50)

5. **Arima-LSTM model**

Compare to pure Arima, all the stocks showing improvement:

From the evaluation strategy, pure Arima is worse than using the mean as the prediction because R^2 < 0 on all of the 7 stocks, while R^2 > 95% on 4 stocks, and R^2 > 85% on all of the 7 stocks. Shows that the Arima-LSTM model explains most of the variance.

5.1 Apple (AAPL)

MSE: 98.8% improvement (2415.38 → 27.80)

MAE: 90.8% improvement (42.32→ 3.88)

5.2 IBM (IBM)

MSE: 99.4% improvement (3159.64 → 18.63)

MAE: 90.8% improvement (50.27 → 2.82)

5.3 Meta (META)

MSE: 99.6% improvement (3159.64 → 18.63)

MAE: 95.4% improvement (232.08 → 10.56)

5.4 Microsoft (MSFT)

MSE: 98.9% improvement (9175.54 → 101.68)

MAE: 89.5% improvement (72.63 → 7.65)

5.5 NVIDIA (NVDA)

MSE: 85.8% improvement (74028.11 → 10484.09)

MAE: 86.8% improvement (197.59 → 26.05)

5.6 Tesla (TSLA)

MSE: 96.0% improvement (6176.10 → 249.41)

MAE: 82.9% improvement (66.15 → 11.30)

5.7 Verizon (VZ)

MSE: 92.6% improvement (15.60 → 1.16)

MAE: 74.7% improvement (3.00 → 0.76)

(b) More descriptions on GRU-Garch and LSTM-Garch

1. **GRU-Garch:**

   Our analysis reveals a crucial pattern: the effectiveness of adding GARCH to the GRU model appears to be inversely related to the initial GRU performance.

   GRU Low Error Cases

   - When GRU shows a very low error (e.g., VZ with MSE of 0.30)
   - Adding GARCH tends to degrade performance
   - Example: VZ showed 34.9% worse MSE after adding GARCH

   GRU High Error Cases

   - When GRU shows higher error (e.g., AAPL with MSE of 39.84)
   - Adding GARCH significantly improves performance
   - Example: AAPL showed 71.2% better MSE after adding GARCH

Performance Threshold Pattern

Based on our dataset, we can identify approximate thresholds where the hybrid model becomes beneficial:

MSE Threshold

- Beneficial when GRU MSE > 30
- Detrimental when GRU MSE < 10

MAE Threshold

- Beneficial when GRU MAE > 5
- Detrimental when GRU MAE < 2

RMSE Threshold

- Beneficial when GRU RMSE > 6
- Detrimental when GRU RMSE < 3

2. **LSTM-Garch:**

Our analysis and observations in the LSTM-Garch hybrid model shows better performance with mid-cap stocks (TSLA) than large-cap tech giants (AAPL, MSFT, META) and the stable telecommunications stock (VZ) shows slight improvement with the hybrid model. However, High-volatility stocks (NVDA, TSLA) show divergent results. TSLA shows improvement but NVDA shows significant degradation. We observe that the LSTM-Garch model's volatility handling is less consistent than GRU-Garch.

Performance Threshold Pattern

- Similar to GRU-GARCH, initial LSTM performance affects hybrid model success
- The threshold pattern is less clear and more volatile
- Lower initial errors (VZ: MSE 0.309) show minimal improvement
- Higher initial errors show significant degradation in most cases

3. **GRU-Garch V.S. LSTM-Garch**

Based on the output and observation above, we can see that LSTM-GARCH shows less consistent improvements compared to GRU-GARCH. Only 2 out of 7 stocks show improvement (compared to 4 out of 7 for GRU-GARCH). LSTM-GARCH tends to produce larger error increases when it performs poorly. The improvements, when present, are generally smaller than with

GRU-GARCH. For volatility handling, LSTM-GARCH appears less effective at handling high-volatility stocks.

(c)Tables

Table 1. Evaluation Results for each model and stock

Stock	Model	MSE	MAE	R^2
AAPL	GRU	39.84418	5.748402	0.940097
	LSTM	**6.772204**	**1.93896**	**0.989818**
	Standard	6.599229	1.899645	0.990019
	Arima	2415.384	42.32494	-2.36261
	Arima+LSTM	27.797	3.88	0.962
	Garch	16.64307	3.856158	-14.9373
	GRU-Garch	11.46384	2.449823	0.984012
	LSTM-Garch	15.67571	2.839001	0.978137
IBM	**GRU**	**5.315318**	**1.568222**	**0.993789**
	LSTM	10.2844	2.440156	0.987983
	Standard	4.68257	1.405136	0.994534
	Arima	3159.636	50.27181	-2.86113
	Arima+LSTM	18.633	2.816	0.978
	Garch	1.281469	0.926214	-0.51212
	GRU-Garch	9.824694	2.362427	0.988054
	LSTM-Garch	6.128372	1.732848	0.992549

META	GRU	148.7772	9.293132	0.99189
	LSTM	**104.8046**	**7.274903**	**0.994287**
	Standard	72.74642	5.479854	0.996015
	Arima	74655.24	232.0838	-2.55908
	Arima+LSTM	272.865	10.56	0.987
	Garch	27.66499	5.064757	-6.20469
	GRU-Garch	137.852	8.303587	0.993393
	LSTM-Garch	127.5522	8.097174	0.993887
MSFT	GRU	41.54565	5.294612	0.989585
	LSTM	**26.6781**	**3.941074**	**0.993312**
	Standard	25.0093	3.816096	0.9937
	Arima	9175.539	72.63049	-1.0148
	Arima+LSTM	101.683	7.651	0.977
	Garch	1.64395	1.095374	-0.55291
	GRU-Garch	33.88891	4.426056	0.992506
	LSTM-Garch	112.4986	7.670652	0.975122
NVDA	GRU	2649.598	12.23345	0.962893
	LSTM	**2555.365**	**10.99387**	**0.964213**
	Standard	2535.872	11.06063	0.96449
	Arima	74028.11	197.5897	-0.03839
	Arima+LSTM	10484.09	26.045	0.863

	Garch	390.6409	18.30687	19.76464
	GRU-Garch	3010.255	15.93891	0.95783
	LSTM-Garch	3080.26	16.65185	0.956849
TSLA	GRU	207.5451	12.42449	0.87455
	LSTM	158.6278	10.49566	0.904118
	Standard	62.72147	5.621065	0.961989
	Arima	6176.104	66.15359	-2.84752
	Arima+LSTM	249.408	11.295	0.859
	Garch	29.87027	5.10113	-3.34125
	GRU-Garch	**83.29943**	**6.47311**	**0.948293**
	LSTM-Garch	118.0236	7.441091	0.926739
VZ	**GRU**	**0.296435**	**0.383593**	**0.971408**
	LSTM	0.309226	0.400965	0.970175
	Standard	0.294147	0.377662	0.971676
	Arima	15.59744	3.000769	-0.6151
	Arima+LSTM	1.158	0.758	0.889
	Garch	1.085854	0.761944	-0.03873
	GRU-Garch	0.399783	0.465758	0.959214
	LSTM-Garch	0.302884	0.392033	0.9691

V. Conclusion

(a) Summary and Rationalizations

From Table 1, we see the GRU model and LSTM model approach the standard and show best prediction in most experiments.
But there is a reason behind that.
Compared to GRU rate stock predict model and LSTM rate stock predict model, our hybrid models have fewer hidden layers in the feed forward neural network after GRU and LSTM modules and a much smaller number of neurons in each hidden layer.
By simplifying the neural networks, the hybrid models created predicted values in proximity with limitations of computing power.
On the other hand, the architecture design of our GRU model and LSTM model takes much more detailed refinement than the hybrid models.
However, nobody can deny that hybrid models generally work better than pure GRU models and LSTM models.
In conclusion, model selection and architecture design should always be based on scenarios and datasets. There exists no single universal solution of prediction to every stock on the market.

(b) Possible Improvements and Future Extensions

1. Basically, if we had free access to daily metrics and ratios instead of using quarterly financial data combined with daily basic prices and trade volume, the accuracy would have been much better.

2. If we predict a longer term stock price (eg. the stock price after two weeks from today) instead of the next day stock price, it will be more useful for catching the trend and volatility. And it will be a more challenging and interesting project.

3. The financial report has too many entries unrelated to the stock price. If someone with finance knowledge could help us select features for the models, the improvement would be guaranteed.

4. We used 8 years data for training data and 2 years data for testing data. This is too short to reflect the real stock market. However, it is very difficult to find financial metrics and ratios for stocks beyond the 10 year period. There are a lot of missing cells(empty cells, “-”, “N”, etc) in the data beyond the 10 year period. We cannot simply do data cleaning by filling fictional values with our imagination.

5. The usage of Transformer will be an ideal advancement, especially the Temporal Fusion Transformer (TFT). But we lack time, data and computing power to try Transformers. Despite the pity, we did learn about the TFT during the project.

   The Temporal Fusion Transformer (TFT) is a state-of-the-art deep learning architecture designed for time series forecasting tasks. It combines flexibility, interpretability, and robustness, making it well-suited for applications like stock price prediction, demand forecasting, and more.

   The Key Advantages of TFT include:
   - Multivariate Time Series Handling,
   - Attention Mechanism,
   - Temporal Context Processing,
   - Built-In Interpretability
   - Multi-Task Learning
   - Robustness to Missing Data
   - Modeling Nonlinear Relationships

   We are more than excited to implement Transformer in future projects.

6. Graph Neural Network is a potential way to link many stocks and other factors together and assign weights to them based on attention mechanisms.

7. A Major Flaw in Transformers is that cyclical characteristics are not caught by the learning. Peking University Introduces Fourier Analysis Networks (FAN) to Address Deficiencies in Periodic Feature Modeling.

   Despite the significant successes achieved by foundational models like MLP and Transformer, they have inherent deficiencies in capturing periodic patterns. Even with a simple sine function, current foundational models struggle to grasp its periodic nature and exhibit erratic behaviors when extrapolating, failing to capture the essence of periodic phenomena effectively.

   To address this, Professor Li Ge's team from Peking University has proposed a novel architecture called FAN (Fourier Analysis Networks) in the paper "FAN: Fourier Analysis Networks"(2024) several days ago. By incorporating the concept of Fourier series, FAN can directly embed periodic information into the network structure, allowing the model to capture and understand periodic patterns in data more naturally. By combining FAN and current approaches, the stock prediction can be more accurate with models recognizing cyclical patterns.

8. For model selection between pure GRU versus GRU-Garch, it is better to use pure GRU for stocks with stable prices and good initial GRU performance, and choose to implement the hybrid model for volatile stocks or when GRU shows high error rates. Regularly reassess model selection based on changing market conditions. For LSTM-Garch, it can provide improvements in specific cases, generally less robust and more volatile in its performance compared to both the base LSTM model and the GRU-GARCH hybrid approach. The model appears to be more sensitive to initial conditions and market volatility, making it potentially less suitable for general application across different types of stocks.

VI. Individual Tasks

Group github repository: https://github.com/hx394/stockPredictModelsProject

Hongzhen Xu:

- Propose the objective and approaches of this project, brainstorm ideas for the whole procedure and improvement, distribute tasks to team members.
- Collect data, preprocess data (draw PCA elbow graph, create principal components, combine basic daily data with PCA results, reverse the data based on date).
- Complete the Deep Learning codes, and then ask the team members for potential improvement using Garch and Arima.
- Design, build and train GRU rate stock price predict model and LSTM rate stock price predict model.
- Test models on testing data set and generate comparing group results in csv format and draw plots. The GRU rate and LSTM rate models have been updated and improved gradually by tuning parameters and making small adjustments. The detailed refinement shows its magic on model performance after repeated careful experiments. Provide team members with these results and suggest evaluation strategies.
- Create the github repository and “ReadMe.md”. Ask all team members to upload their work on the same git repository.
- Write this report as the first drafter.

Cheyi Wu:

- Collecting data from stockanalysis.com and providing ideas with the selection of financial metrics and ratios. Additionally, providing the sample data for teammates to do data preprocessing.
- Implement GRU-Garch and LSTM-Garch hybrid models based on Hongzhen’s work on GRU and LSTM models.
- Providing clear output by generating the graph and CSV files from code to do deep analysis and comparisons among GRU and LSTM base models and Garch hybrid models.

- Gives model selection suggestions and analysis among GRU, LSTM, GRU-Garch, and LSTM-Garch.
- Complete this report and Readme files for my part.

Jingming Cheng:

- Synthesized data from different txt and csv files for future analysis.
- Implemented Arima and Arima-LSTM model, tested on testing data set and generated comparing group results in csv format and drew plots.
- Evaluated and arranged all team members' results based on MSE, MAE and R^2.
- Resolved compatibility issues of programs and integrated files from all of our 3 members.
- Optimized Garch.ipynb file.
- Wrote part of the readme file of our GitHub repository and replenished this report.

VII. References

Choi, H.-K. (2018). *Stock Price Correlation Coefficient Prediction with ARIMA-LSTM Hybrid Model*. arXiv preprint arXiv:1808.01560.

Dong, Y., Zhang, J., & Li, G. (2024). *FAN: Fourier Analysis Networks*. arXiv preprint arXiv:2410.02675.

Kim, K.-J., & Won, H.-K. (2018). *Forecasting the Volatility of Stock Price Index: A Hybrid Model Integrating LSTM with GARCH-Type Models*. *Complex & Intelligent Systems*, 4(3), 211-221.

Mutinda, J. K., & Langat, A. K. (2024). Stock price prediction using combined GARCH-AI models. *Scientific African, 26*, e02374.

Nelson, D. M., Pereira, A. C. M., & de Oliveira, R. A. (2017). *Stock Market's Price Movement Prediction with LSTM Neural Networks*. In *2017 International Joint Conference on Neural Networks (IJCNN)* (pp. 1419-1426). IEEE.

Saha, S., Mondal, A., & Saha, S. (2021). *A Combined Model of ARIMA-GRU to Forecast Stock Price*. In *2021 International Conference on Intelligent Engineering and Management (ICIEM)* (pp. 126-130). IEEE.

Xiao, W., & Xiang, Y. (2019). *A Comparative Study of LSTM and GRU Network Structures for Stock Price Forecasting*. In *Proceedings of the 2019 IEEE International Conference on Big Data (Big Data)* (pp. 5493-5495). IEEE.