Logan Hight
CS 5393 – Advanced Python
April 20, 2025

Final Exam Report
Stock Price Prediction

1. Introduction

The goal of this project was to use Recurrent Neural Networks (RNNs) for time series forecasting in order to build a system that can effectively predict stock prices. Deep learning models like RNNs are of large importance in the context of financial time series forecasting due to their ability to learn complex patterns and relationships in data. This allows well-designed models to make accurate predictions when handling multi-dimensional time series data that has non-linear dependencies.

I compared two main RNN architecture types: Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). LSTMs are specifically designed to learn long-term dependencies from sequential data, making them highly suited for this project. GRUs are similar to LSTMs and are also designed to process sequential data but do so with a simplified architecture. In addition to these two baseline models, I also explored a variety of different activation functions in my GRU model architecture (ReLU, Leaky ReLU, ELU, and SELU), which will be discussed in more detail in the methodology section of this report. Each implemented model was evaluated quantitatively and statistically, with one model demonstrating clearly superior performance and being recommended for use in deployment scenarios.

2. Dataset & Preprocessing

The dataset I selected consisted of daily historical stock prices for AAPL (Apple Inc.) from January 1st, 2010, to November 13th, 2023. I used Python's Yahoo Finance library to download this stock data. By default, this data included features such as Open, High, Low, Close, and

Volume. For my initial baseline LSTM and GRU models, I only used the "Close" attribute. Later in the modeling process, I expanded the input data to also include the "Volume" attribute in an attempt to improve model performance.

Before I began modeling, I performed the necessary preprocessing techniques on the downloaded data. This first involved extracting the "Close" price, resetting the index so that "Date" became a column, and then plotting the close price over time as seen below.

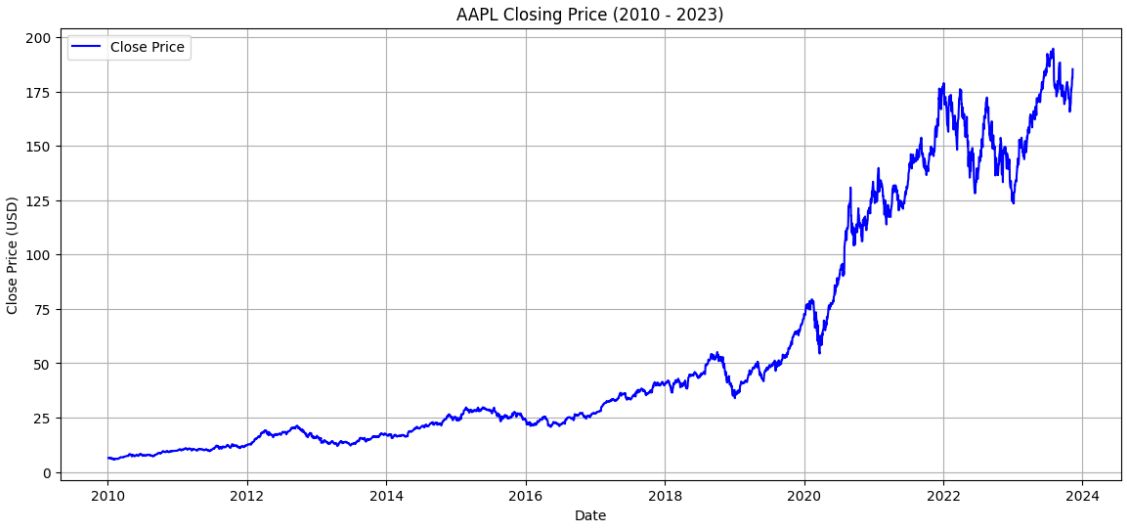

Figure 1: AAPL Closing Price (2010-2023)

The above visualization allowed me to gain a visual understanding of long-term close price trends and volatility in Apple Inc.'s stock before I began modeling. Next, I normalized my data by using MinMaxScaler to scale the "Close" prices to a range between 0 and 1. This step is important because neural networks are highly sensitive to the scale of data, and normalization aids in model convergence.

The final main preprocessing step was to create input sequences for my models. The RNNs I implemented required fixed-length sequences of input data in order to predict the next value in

the sequence. I chose to create overlapping sequences of 60 days to predict the closing price of the 61st day. After this, I split the data using a standard 80%/20% train-test split, which ensures that the models can learn effectively while still providing the ability for meaningful evaluation on the test data. Finally, I reshaped the input data so that it was compatible with my RNN model architectures.

Later in the modeling process, when I expanded the input data to include both "Close" and "Volume" attributes, my preprocessing steps remained largely the same. I normalized the data using MinMaxScaler to scale both the "Close" prices and daily "Volume" to a range between 0 and 1. Next, I created input sequences again of length 60 but including both attributes and split the data using a standard 80%/20% train-test split.

3. Methodology

I followed the same basic steps to implement, train, and evaluate each model architecture. I began by defining the architecture for each model. All models were compiled using the Mean Squared Error (MSE) loss function, Mean Absolute Error (MAE) as a secondary metric, and the Adam optimizer for adaptive learning with a learning rate of 0.001.

Next, I trained each model using early stopping in order to prevent overfitting. Each model was trained for a maximum of 100 epochs, but training stopped early if the validation loss did not improve after 10 epochs. Next, I plotted the model training history and then used the model to make predictions on the test dataset. As part of the prediction process, I inverse-transformed the predictions and real test values back to the original dollar scale using the MinMaxScaler that I fitted as part of my preprocessing. This allowed me to effectively gauge model performance using all of my chosen evaluation metrics, which are discussed in section 4 of this report.

3.1 LSTM (Baseline)

The first RNN architecture I implemented was a standard LSTM model. LSTMs are designed to effectively handle sequential data and are not plagued by the vanishing gradient problem that traditional RNNs face, which helps them learn complex temporal patterns and long-term dependencies. I designed my LSTM model to serve as a foundation against which I was able to compare the subsequent models. The LSTM model architecture used only the closing price as input and included two stacked LSTM layers (each with 50 units), a dense layer with 25 units, and a final dense layer with a single output unit that was used to predict the next closing price. A visual depiction of the LSTM model architecture can be seen in Appendix A.1.

3.2 GRU (Baseline)

The next RNN architecture I implemented was a standard GRU model. GRUs are simplified versions of LSTMs and have faster training time and fewer parameters. They are suited for time-series data and are able to effectively model long-term temporal dependencies through gating mechanisms. I designed my GRU model with a dual purpose in mind. I wanted to compare it to my baseline LSTM model architecture, and also have it serve as a point of comparison for my other GRU models with a larger input space and other architectural adjustments that I implemented later in my modeling process. The GRU model architecture used only the closing price as input and included two stacked GRU layers (each with 50 units), a dense layer with 25 units, and a final dense layer with a single output unit that was used to predict the next closing price. A visual depiction of the GRU architecture can be seen in Appendix A.2.

3.3 GRU with Additional Features (Volume)

Next, I sought to further my analysis of the GRU model architecture by including an additional input feature ("Volume"). My initial thoughts were that this could lead to an improvement in prediction accuracy since the model would be provided with more information about market behaviors. The Multi-Feature GRU model architecture used both closing price and daily trading volume as input features, but the model architecture remained the same as the previous models in terms of the GRU and dense layers. A visual depiction of the Multi-Feature GRU model architecture can be seen in Appendix A.3.

3.4 GRU Activation Function Variants

As a final exploration, I experimented with modifying my Multi-Feature GRU model by implementing various activation function variants. The goal of this was to determine how activation functions impact model performance. All of the activation function variant models used both closing price and daily trading volume as input features. The first activation function variant I implemented was the Rectified Linear Unit (**ReLU**).

I chose to explore ReLU as an activation function because it can improve gradient flow and model convergence. The model architecture remained the same as the previous models in terms of the GRU and dense layers, except the default "tanh" activation function was replaced by "ReLU" in all layers. A visual depiction of the GRU ReLU model architecture can be seen in Appendix A.4.

Next, I explored the **Leaky ReLU** activation function. I chose to explore Leaky ReLU as an activation function because it allows for a small gradient for negative inputs which can help

avoid the dying ReLU problem which is when neurons become inactive. The model architecture remained the same as the previous models in terms of the GRU and dense layers, except the default “tanh” activation function was replaced by “Leaky ReLU” in the dense layers. A visual depiction of the GRU Leaky ReLU model architecture can be seen in Appendix A.5.

Next, I explored the Exponential Linear Unit (**ELU**) activation function. I chose to explore ELU as an activation function because it can enable better learning dynamics for negative values when compared to other activation functions. The model architecture remained the same as the previous models in terms of the GRU and dense layers, except the default “tanh” activation function was replaced by “ELU” in all layers. A visual depiction of the GRU ELU model architecture can be seen in Appendix A.6.

Finally, I explored the Scaled ELU (**SELU**) activation function. I chose to explore SELU as an activation function because it allows the neural network to self-normalize, which can help improve training stability and convergence speed. The model architecture remained the same as the previous models in terms of the GRU and dense layers, except the default “tanh” activation function was replaced by “SELU” in all layers. A visual depiction of the GRU Leaky ReLU model architecture can be seen in Appendix A.7.

4. Evaluation Criteria and Justification

Throughout the project, I used two primary evaluation metrics to gauge model performance. The first was Root Mean Square Error (**RMSE**). RMSE penalizes large errors in prediction heavily. This is very important in the context of stock price forecasting since larger errors present significant financial risk. The second was Mean Absolute Error (**MAE)**. MAE represents the

average absolute dollar error in predictions. MAE is also important in the context of stock price forecasting since it is easily interpreted by decision-makers and other stakeholders.

In addition to these two evaluation metrics, I recorded model training time and prediction time in order to obtain a more complete view of model performance. I also performed statistical testing, which allowed me to confirm that the performance differences I observed were statistically significant. All of these evaluation strategies enabled me to validate model performance and make an effective model deployment recommendation.

5. Experimental Results & Analysis

The exploration conducted in my project yielded many interesting results. After completing the training process for all models, I compiled the performance results to effectively compare each implemented model. Below is a figure depicting the RMSE and MAE for each model.

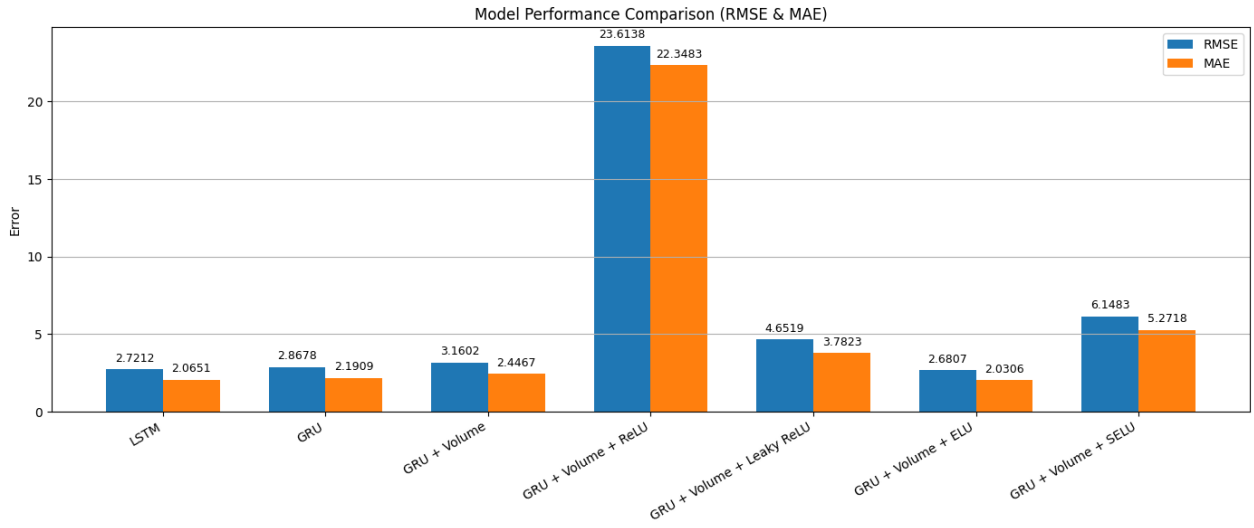

Figure 2: Model Performance Comparison (RMSE & MAE)

As we can see from the bar chart, the best-performing model was the GRU ELU model with an RMSE of 2.68 and an MAE of 2.03. This model was followed by both the baseline LSTM and

GRU models. The initial addition of volume as an input feature yielded slightly decreased performance in the Multi-Feature GRU model. Also, we can see that activation function choice had a large impact on performance. The GRU ReLU and SELU models were by far the worst performing, and the GRU ELU model had the lowest RMSE and MAE out of all implemented models. Next is a figure depicting the training and prediction time for each model.

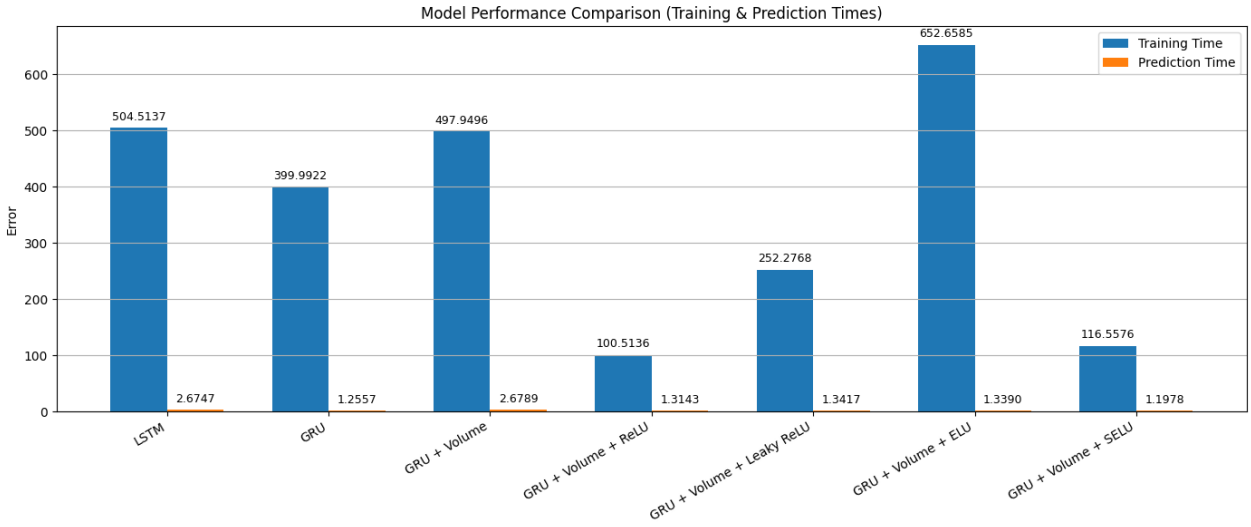

Figure 3: Model Performance Comparison (Training & Prediction Times)

As we can see from the bar chart, the majority of the GRU models trained faster than the LSTM model, which confirms the computational efficiency of GRUs. The initial addition of volume as an input feature resulted in increased training time, likely due to increased dimensionality in the input space of the Multi-Feature GRU model. Again, activation function choice had a large impact on performance. The GRU ReLU and SELU models had the lowest training times, but as we saw previously, this came at the cost of prediction accuracy. The GRU ELU model took by far the longest to train, despite its superior prediction performance. All models demonstrated reasonable prediction times, with the LSTM and Multi-Feature GRU models performing the worst out of all implemented models.

Next, I performed statistical testing to confirm model performance differences. First, I performed paired t-tests between all model combinations on their MAEs. All model pairs were shown to have statistically significant performance differences in MAE, except for the LSTM and GRU ELU models. This pairing had a resulting t-test p-value of 0.145, which indicates the two models are statistically similar in terms of MAE.

The next statistical test I performed was a Wilcoxon signed-rank test between all model combinations on their prediction distributions. For this test, all model pairs were shown to have statistically significant differences in their prediction distributions, even those that had similar performance in terms of RMSE and MAE. This shows that all models make systematically different forecast decisions even if they have similar average errors. Finally, I created a boxplot to display the absolute prediction error distribution for each model, as seen below.

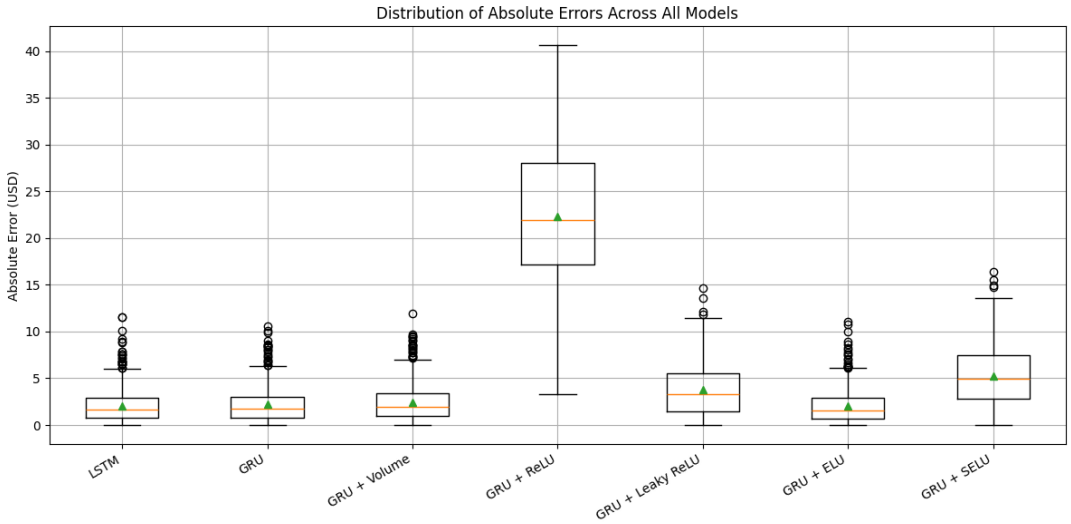

Figure 4: Distribution of Absolute Errors Across All Models

As we can see from the box plot, the GRU, Multi-Feature GRU, and GRU ELU models all demonstrated low medians and compact interquartile ranges, which show stable performance. The LSTM model also had a low median and compact interquartile range, but it had more upper

outliers, which could indicate inconsistent generalization. Once again, activation function choice had a large impact on model performance, with the GRU ReLU, Leaky ReLU, and SELU models showing the worst performance.

6. Final Model Recommendation

After evaluating all implemented models and comparing their performance results, the **GRU ELU** model emerged as the clear best-performing model. The GRU ELU model delivered the best prediction accuracy and had consistent performance across the entire price and time range. It also demonstrated superiority to models with similar average error performance in terms of runtime and stability (specifically, it had ~50% faster prediction time than the LSTM model). ELU clearly improved the gradient flow and effectively handled negative activations, and the GRU ELU model was statistically validated to have significant performance differences from all other implemented models.

7. Conclusion

The results of this work have significant implications within the business context of stock price forecasting. After exploring multiple RNNs for stock price prediction, the GRU ELU model was shown to be the highest performer, with the lowest RMSE and MAE as well as fast prediction time and consistent performance. The GRU ELU model is highly suitable for daily stock trend forecasting, portfolio adjustment and trading strategy systems, and retail finance tools that support investor decision-making. Future work could potentially improve prediction accuracy by incorporating additional input features, which could improve model learning due to a richer representation of complete market conditions. Additionally, different architectures, such as transformer-based or hybrid, could be explored in pursuit of a similar goal. Finally, expanding

the input to multiple stocks or even complete portfolios would result in models that are highly applicable to real-world trading scenarios. However, the experimentation in this project, and the GRU ELU model specifically, provide a strong foundation for future research and demonstrate the ability of well-designed RNN models to deliver high levels of performance in the context of stock price forecasting.

Appendix A

Model Architectures

A.1 LSTM Model Architecture

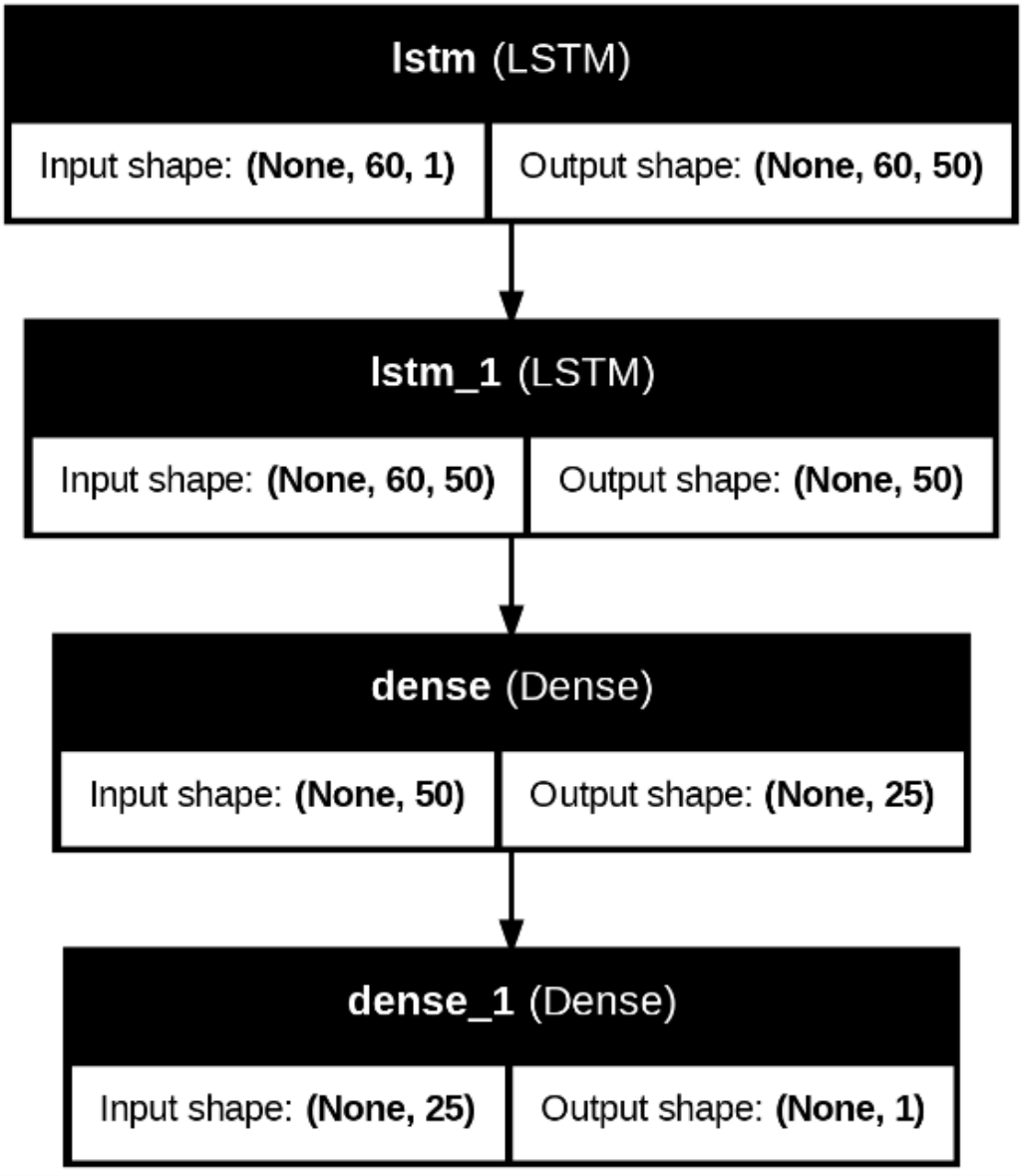

Figure A.1: LSTM Model Architecture

A.2 GRU Model Architecture

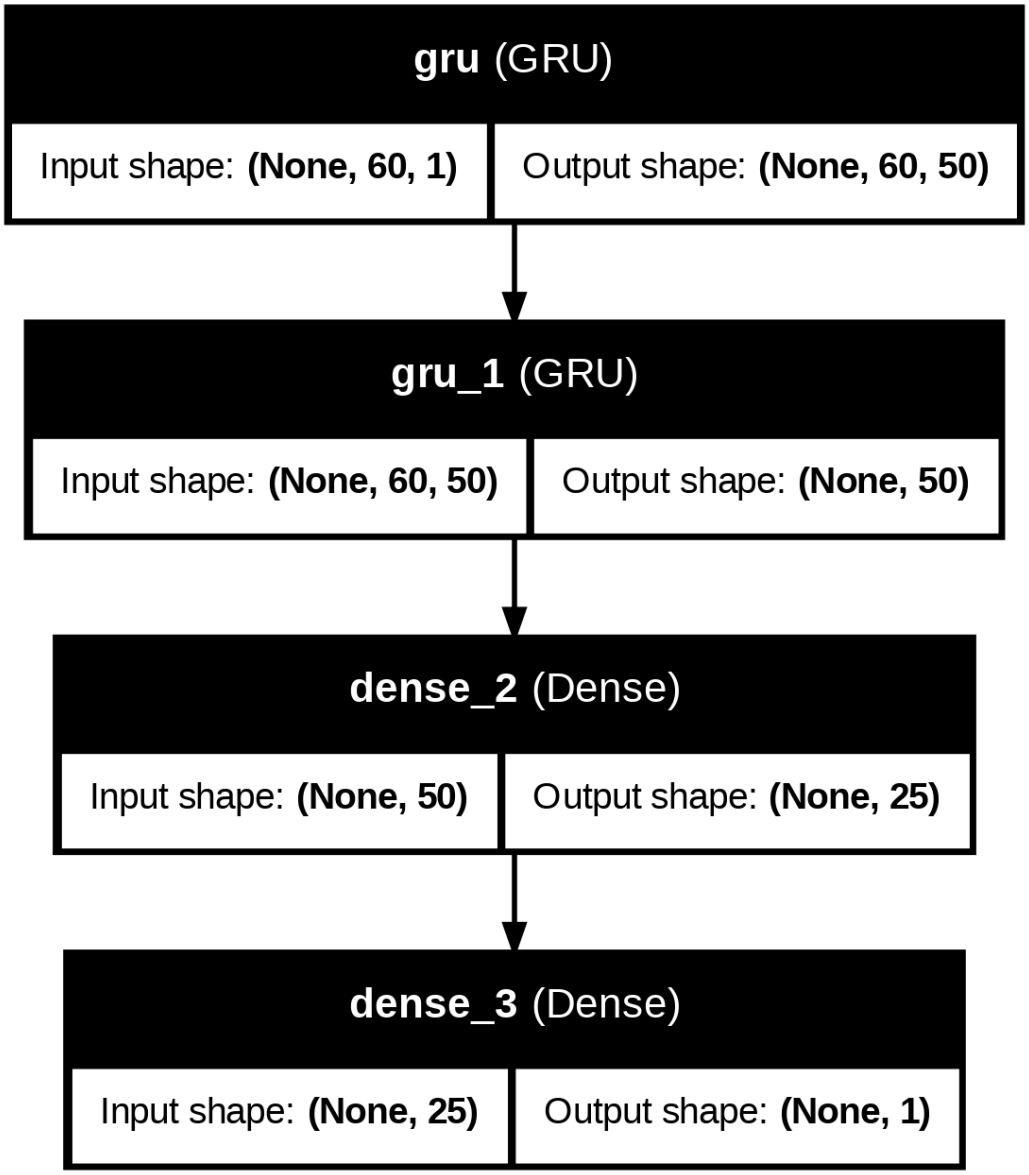

Figure A.2: GRU Model Architecture

A.3 Multi-Feature GRU Model Architecture

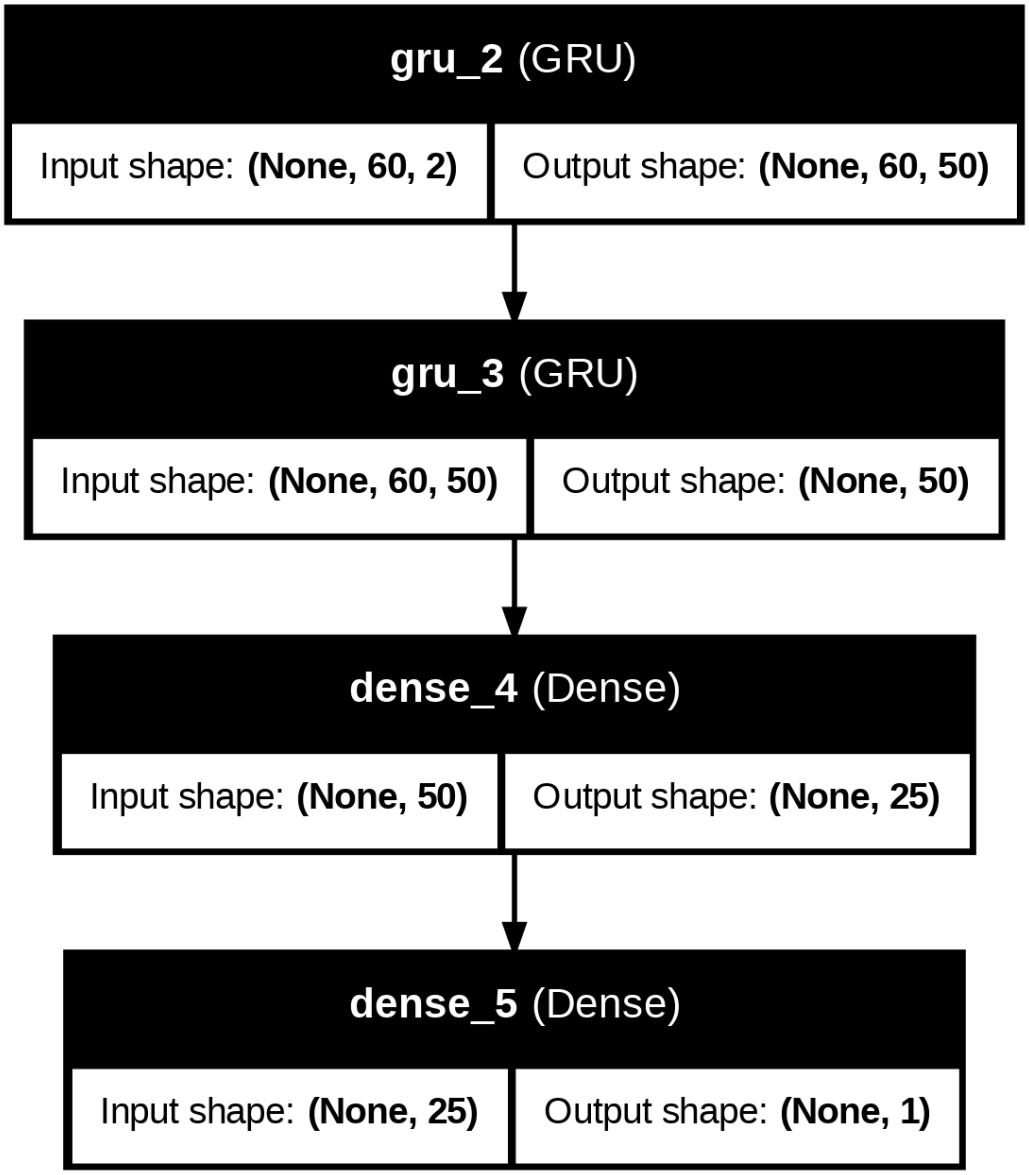

Figure A.3: Multi-Feature GRU Model Architecture

A.4 GRU ReLU Model Architecture

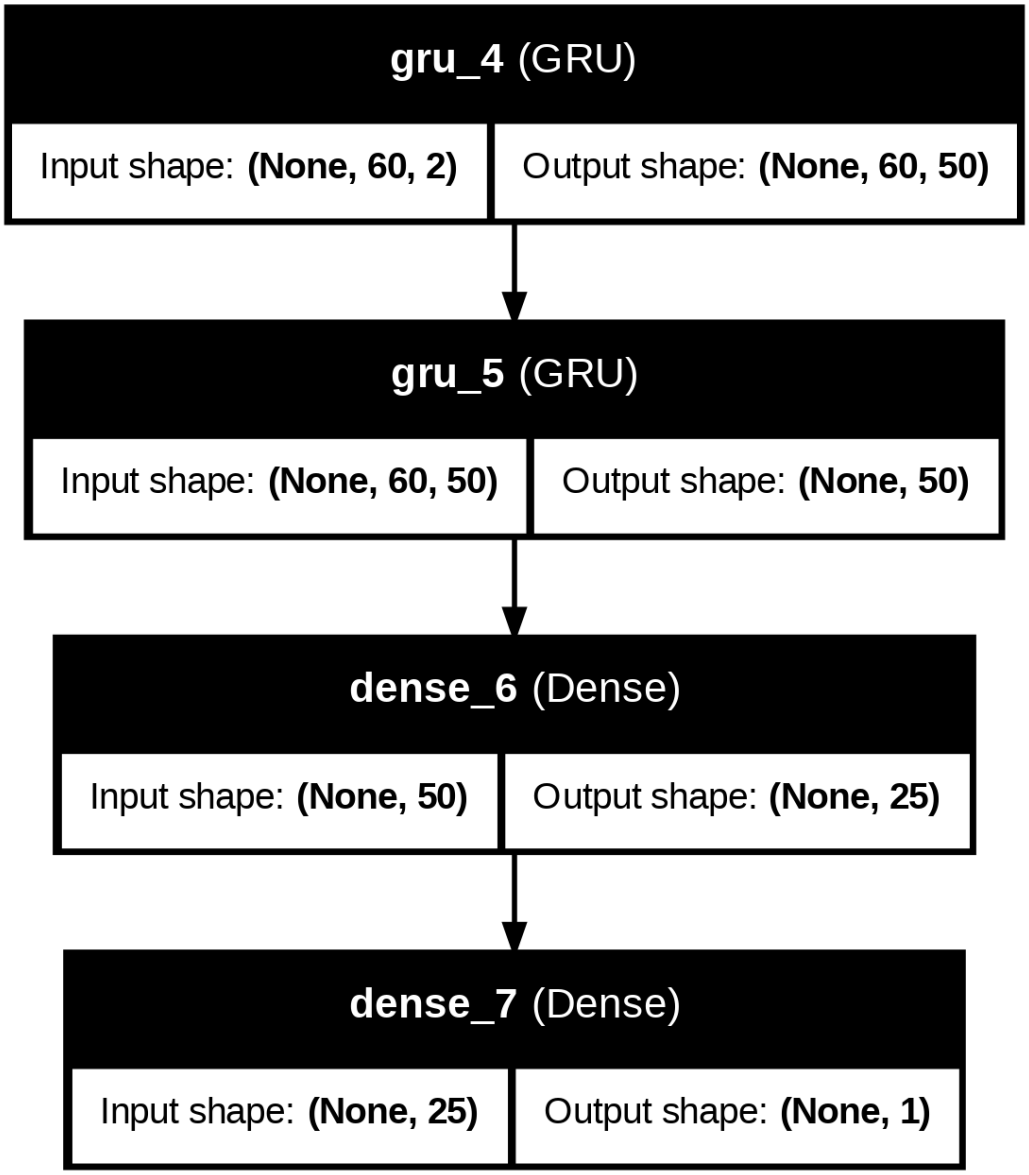

Figure A.4: GRU ReLU Model Architecture

A.5 GRU Leaky ReLU Model Architecture

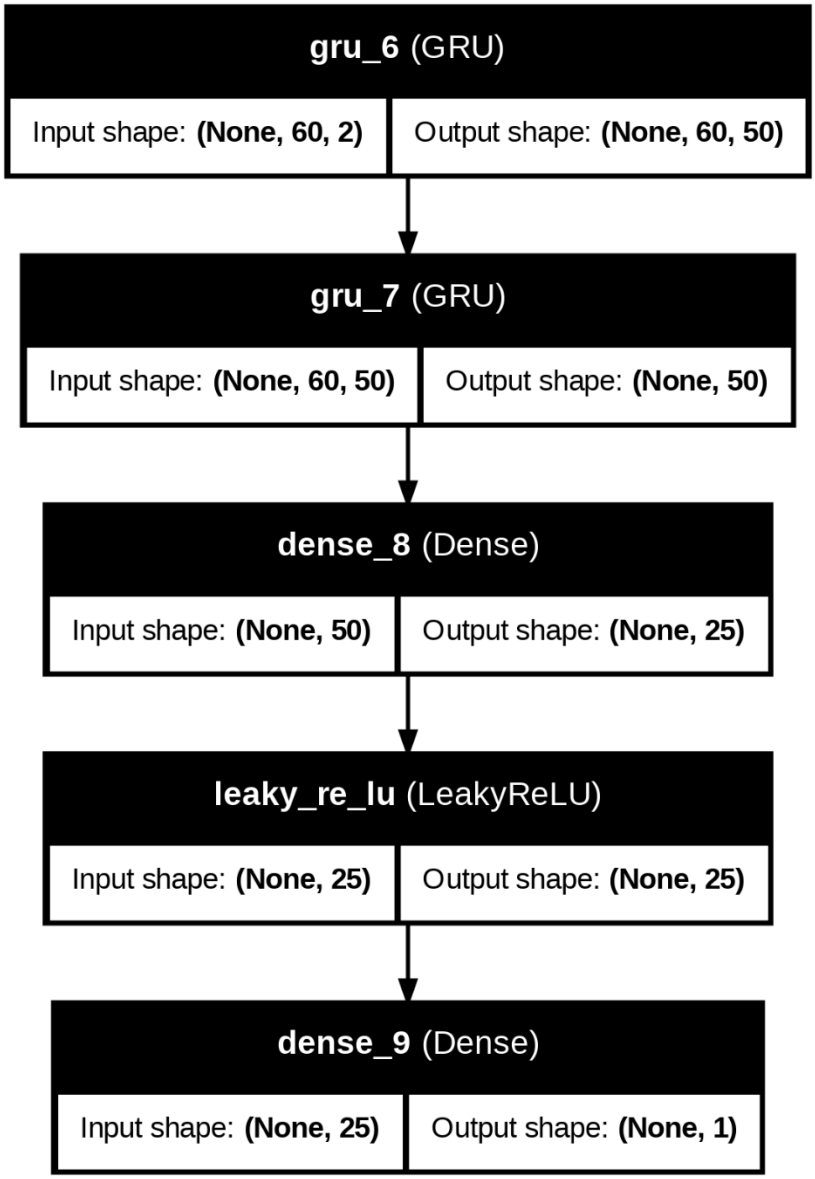

Figure A.5: GRU Leaky ReLU Model Architecture

A.6 GRU ELU Model Architecture

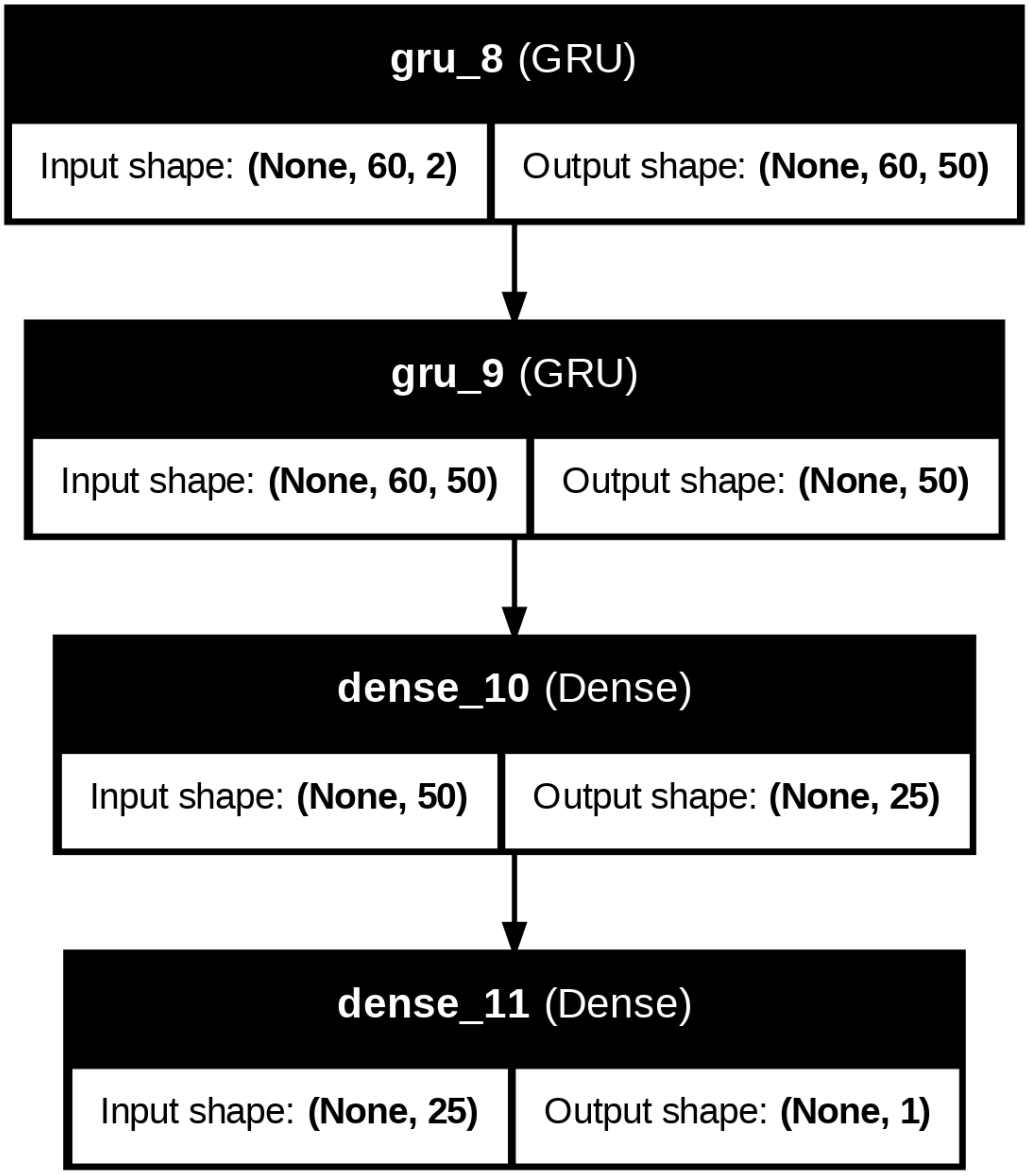

Figure A.6: GRU ELU Model Architecture

A.7 GRU SELU Model Architecture

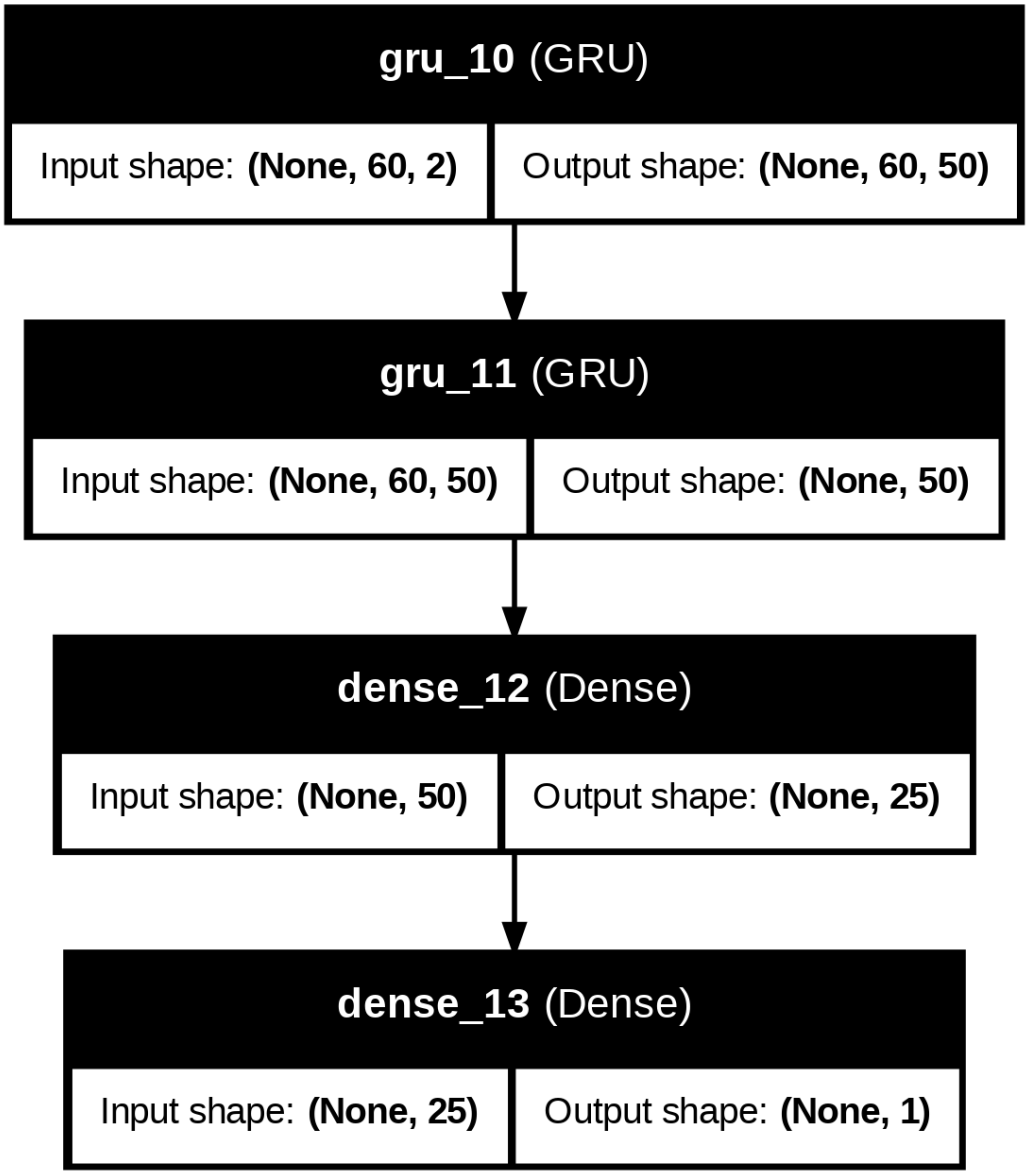

Figure A.7: GRU SELU Model Architecture

Logan Hight
CS 5393 – Advanced Python
April 20, 2025

Appendix B

Model Training Curves

B.1 LSTM Model Training Curves

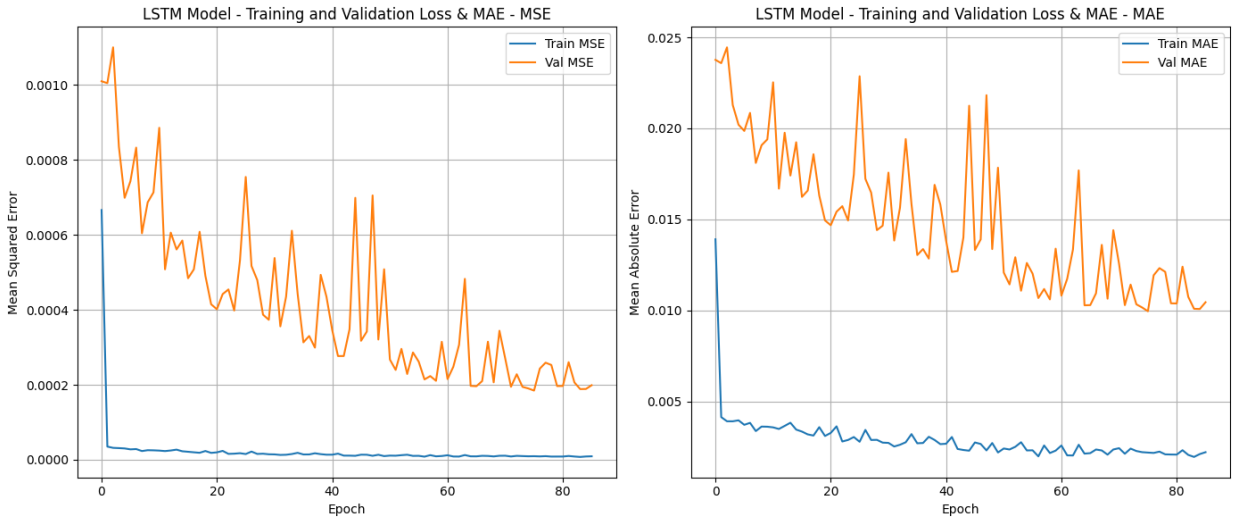

Figure B.1: LSTM Model Training Curves

B.2 GRU Model Training Curves

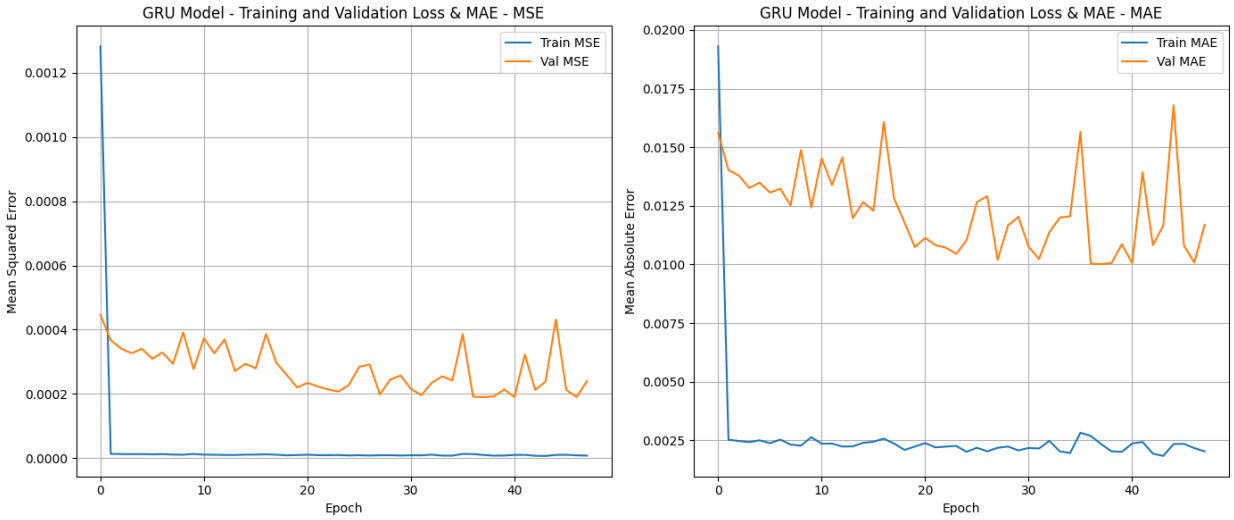

Figure B.2: GRU Model Training Curves

B.3 Multi-Feature GRU Model Training Curves

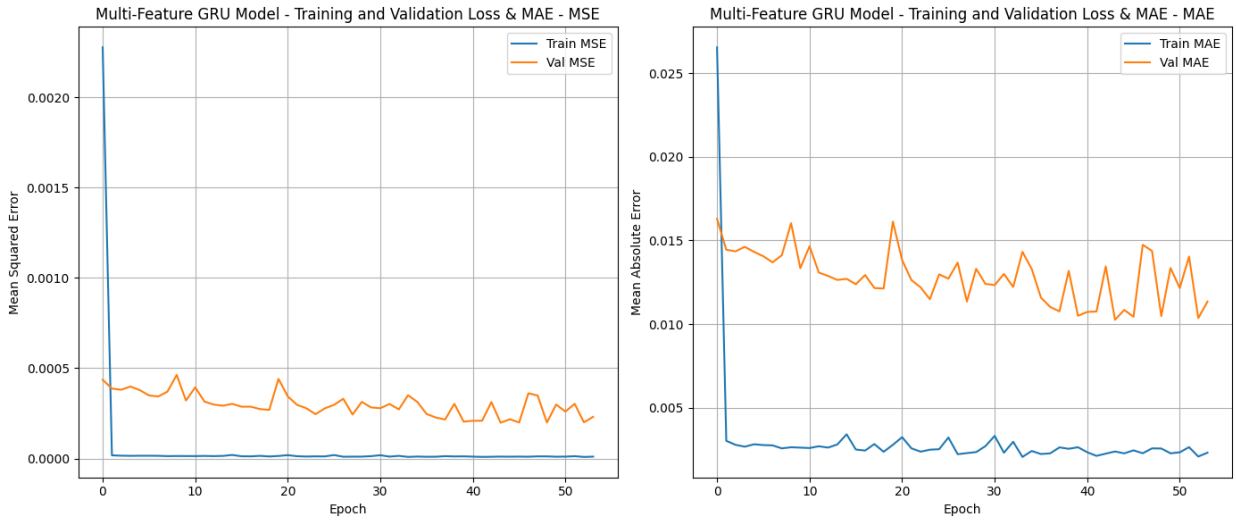

Figure B.3: Multi-Feature GRU Model Training Curves

B.4 GRU ReLU Model Training Curves

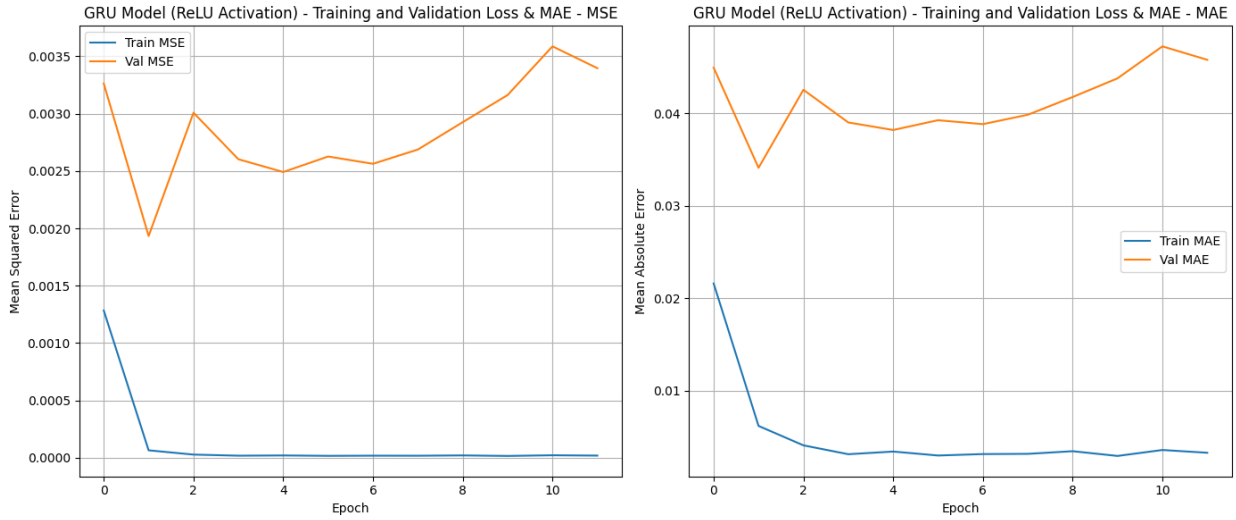

Figure B.4: GRU ReLU Model Training Curves

B.5 GRU Leaky ReLU Model Training Curves

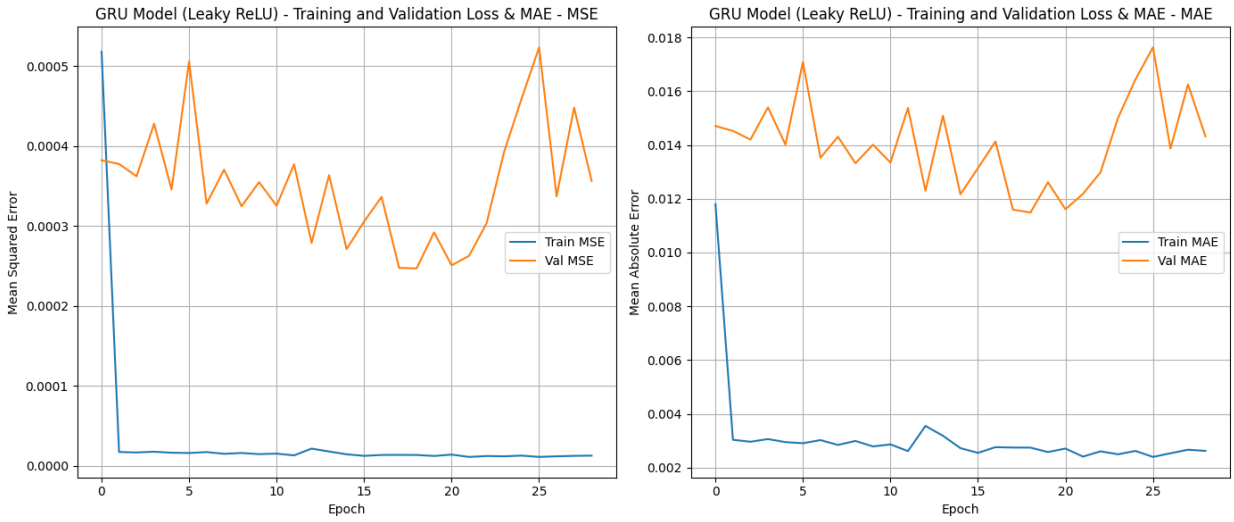

Figure B.5: GRU Leaky ReLU Model Training Curves

B.6 GRU ELU Model Training Curves

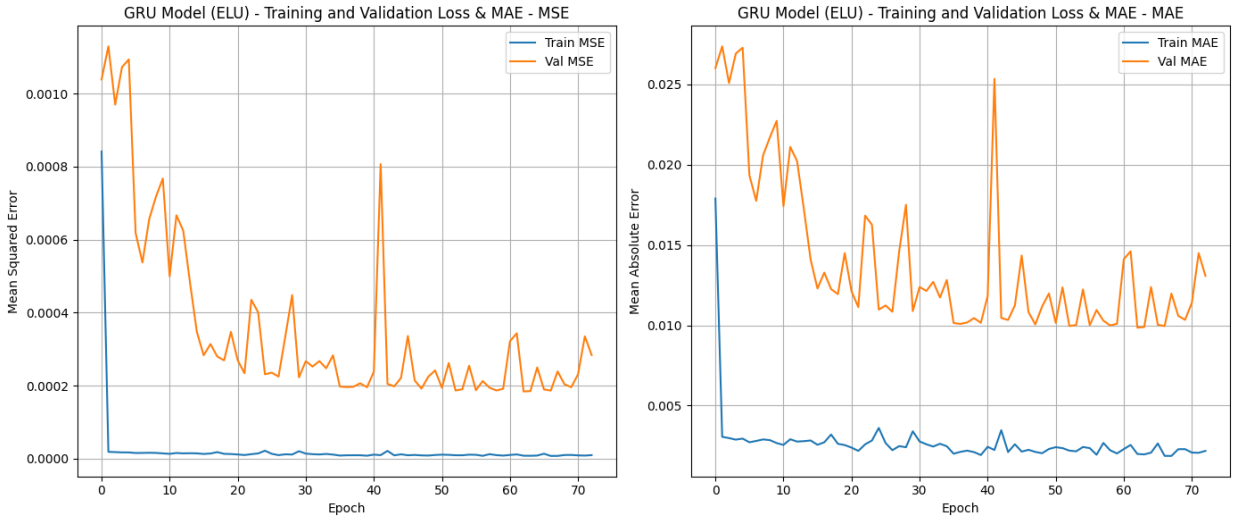

Figure B.6: GRU ELU Model Training Curves

Logan Hight
CS 5393 – Advanced Python
April 20, 2025

B.7 GRU SELU Model Training Curves

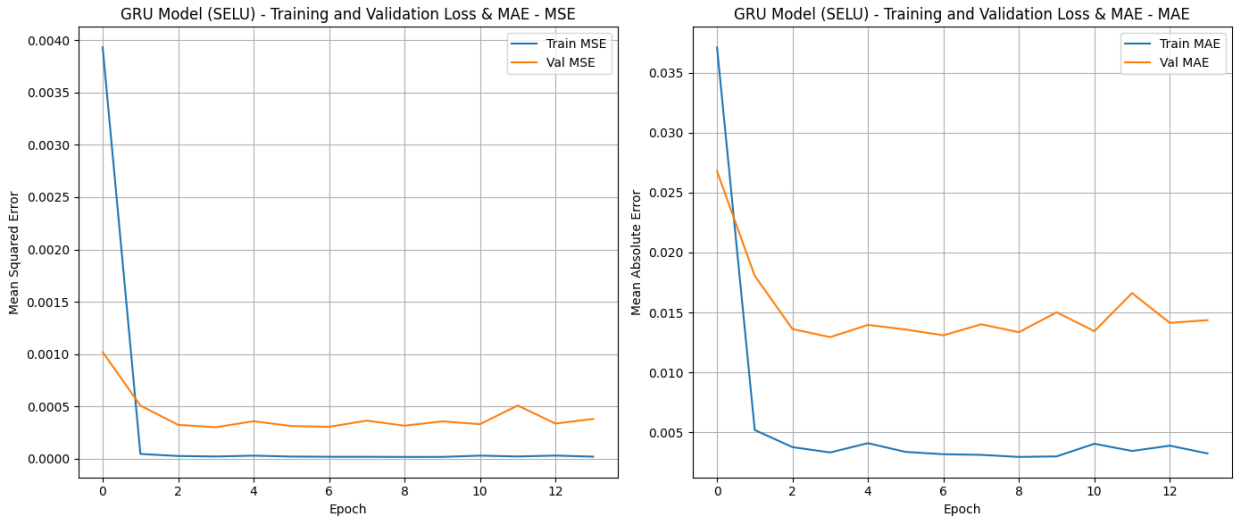

Figure B.7: GRU SELU Model Training Curves

Logan Hight
CS 5393 – Advanced Python
April 20, 2025

Appendix C

Model Prediction Visualizations

C.1 LSTM Model Prediction Visualization

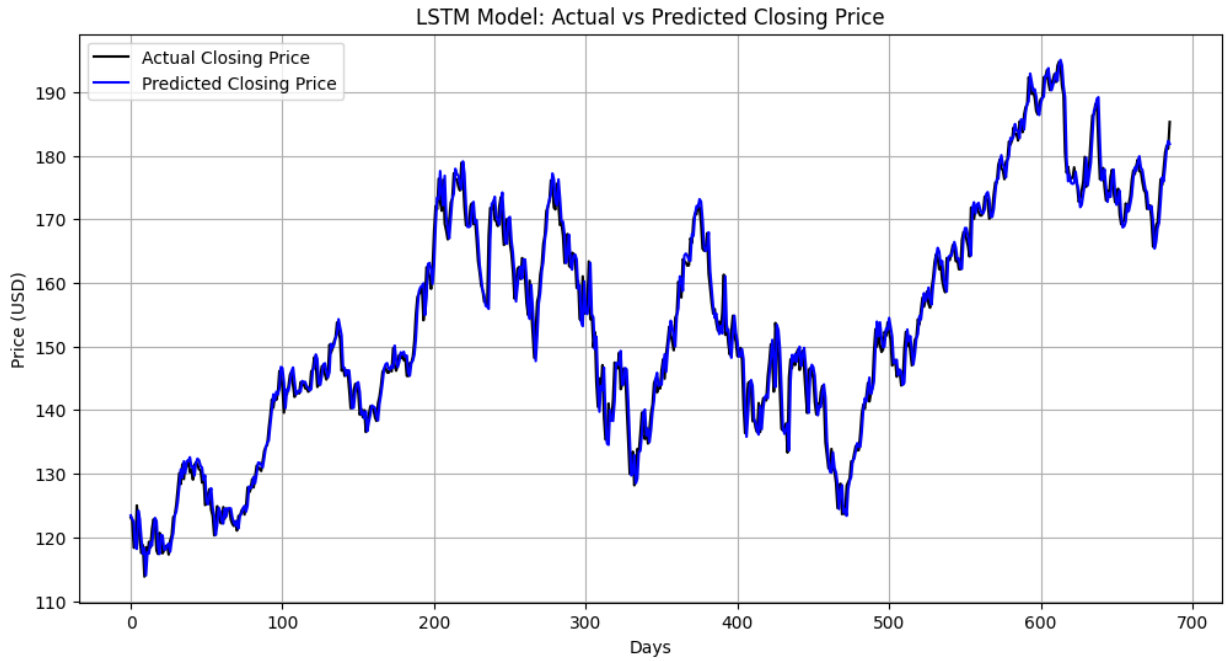

Figure C.1: LSTM Model Prediction Visualization

C.2 GRU Model Prediction Visualization

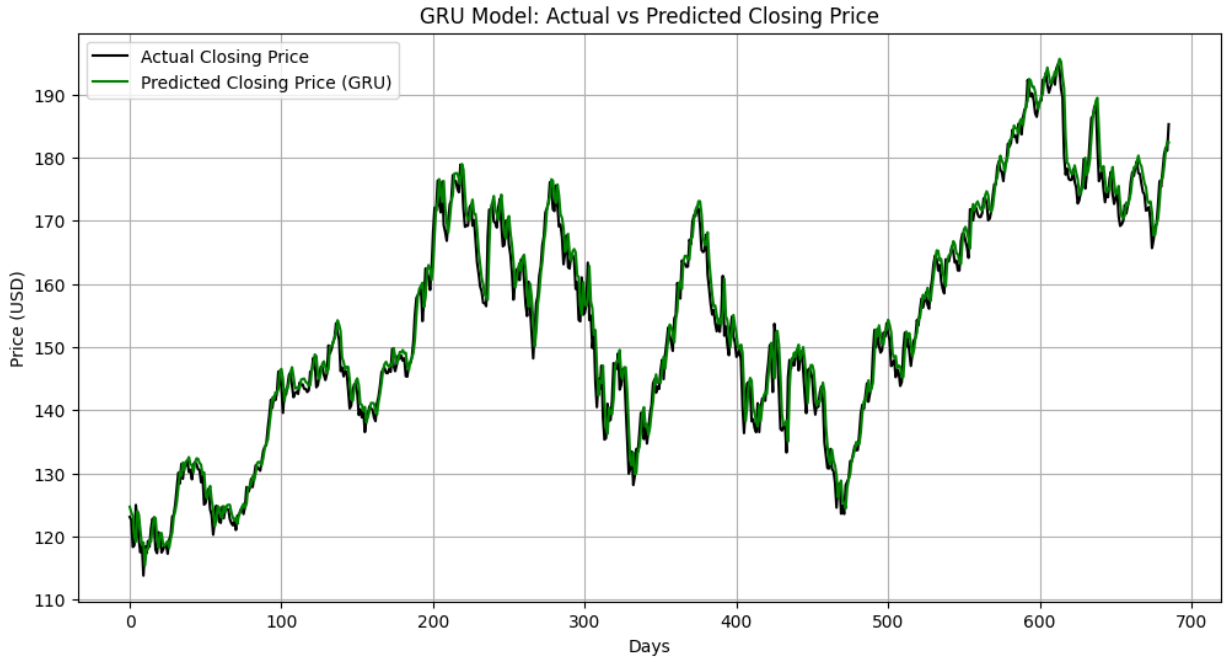

Figure C.2: GRU Model Prediction Visualization

C.3 Multi-Feature GRU Model Prediction Visualization

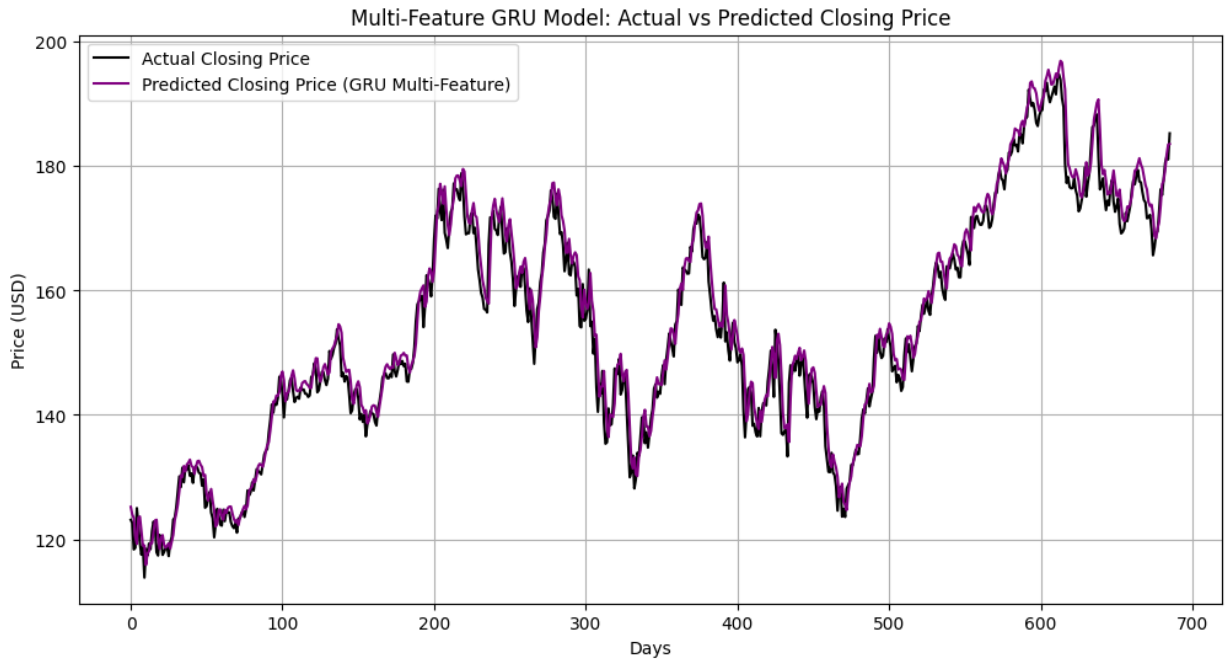

Figure C.3: Multi-Feature GRU Model Prediction Visualization

C.4 GRU ReLU Model Prediction Visualization

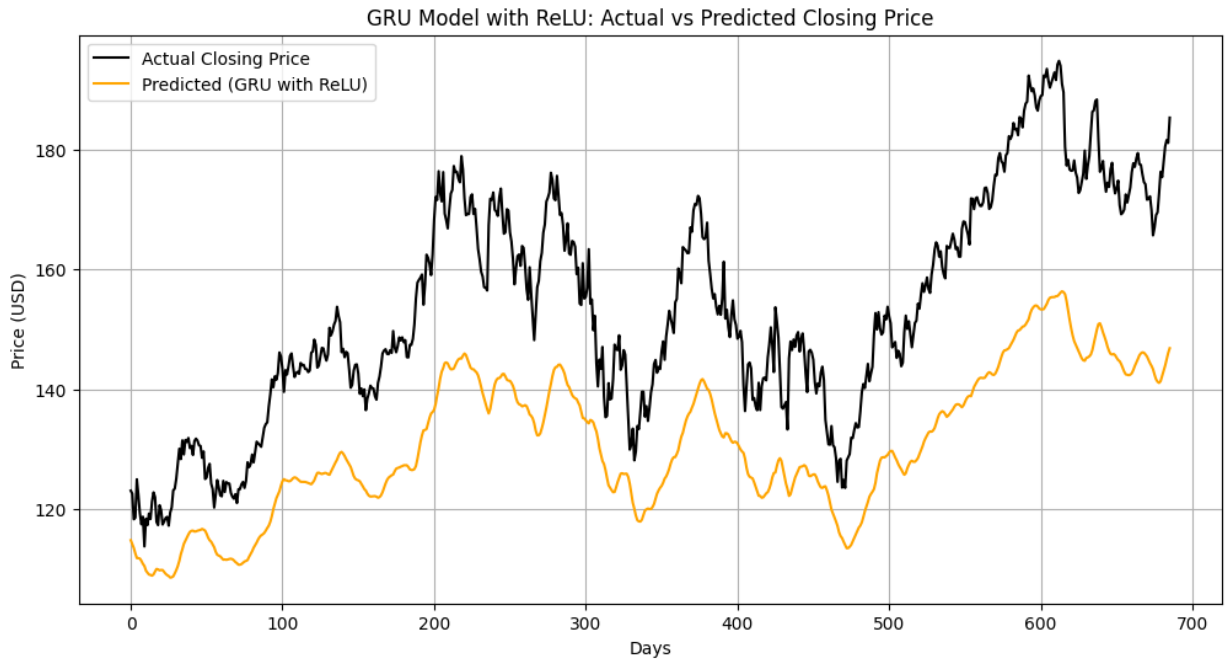

Figure C.4: GRU ReLU Model Prediction Visualization

C.5 GRU Leaky ReLU Model Prediction Visualization

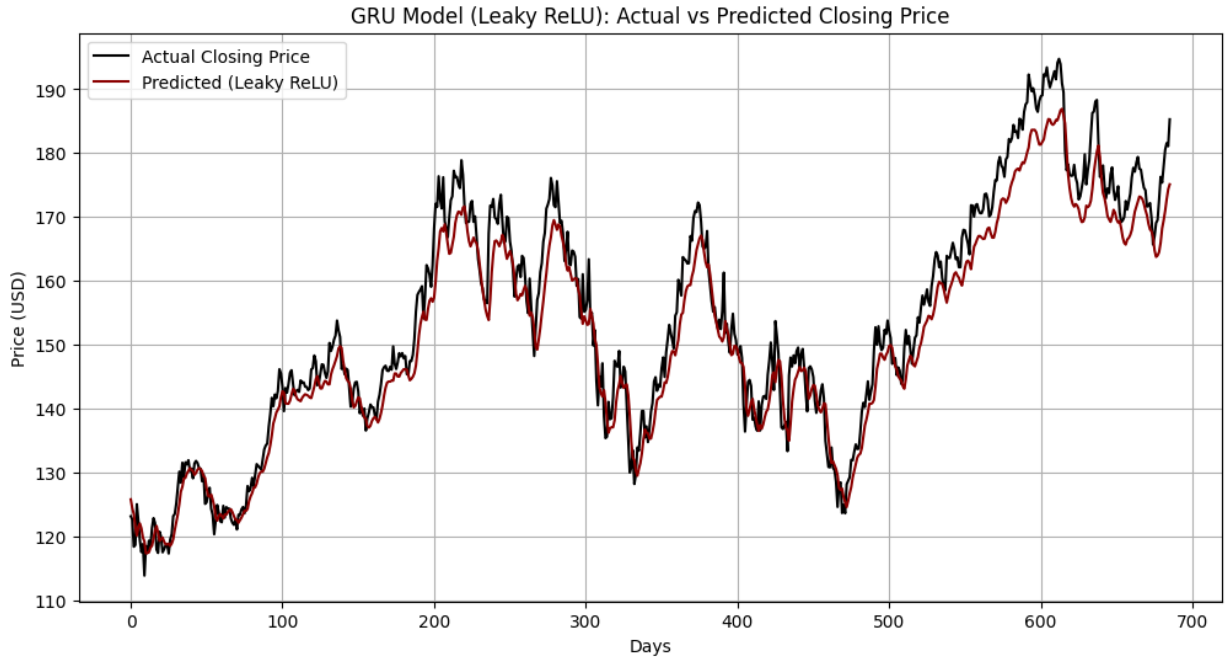

Figure C.5: GRU Leaky ReLU Model Prediction Visualization

C.6 GRU ELU Model Prediction Visualization

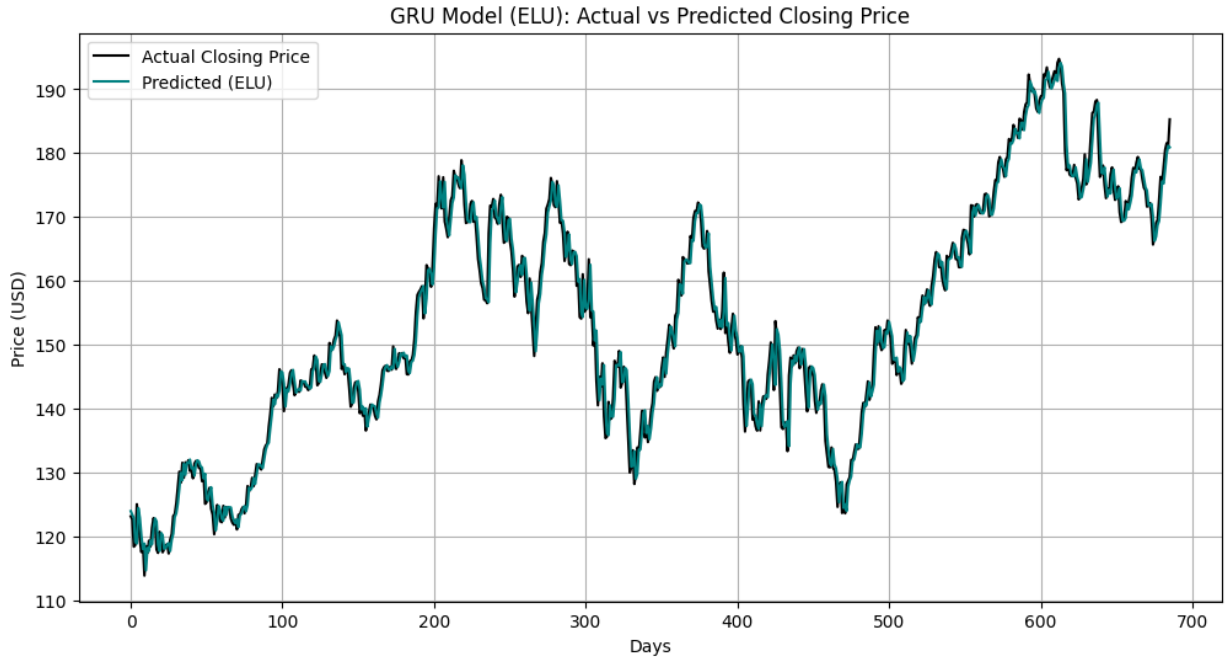

Figure C.6: GRU ELU Model Prediction Visualization

C.7 GRU SELU Model Prediction Visualization

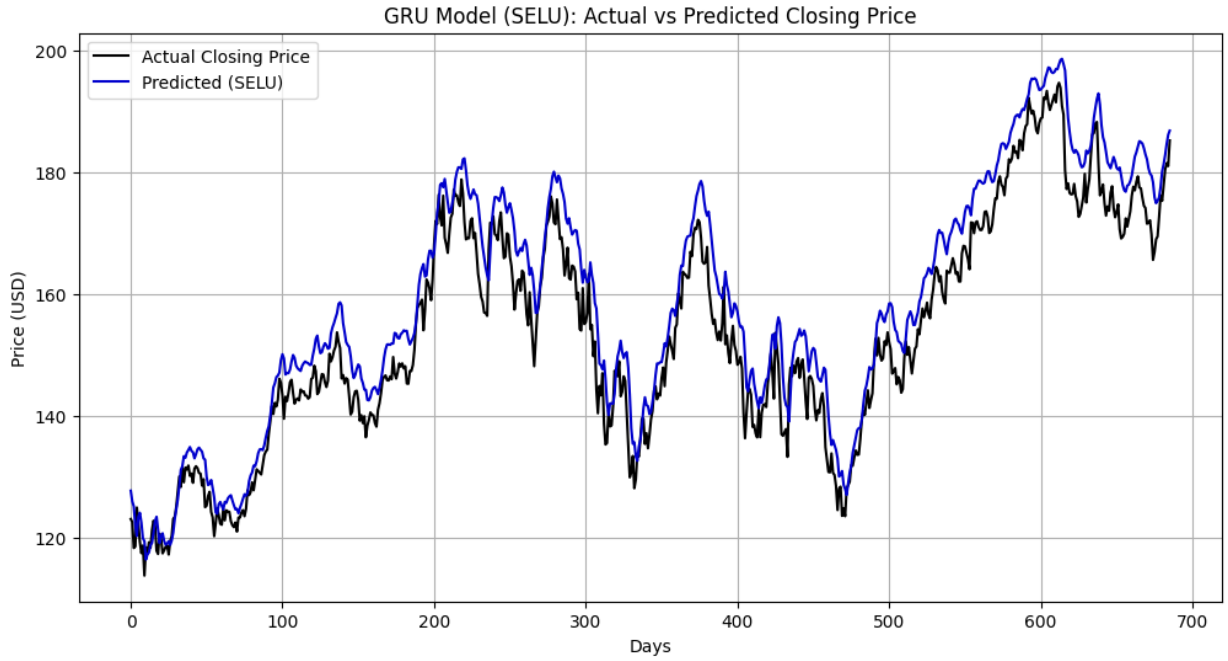

Figure C.7: GRU SELU Model Prediction Visualization