Developing Machine Learning Algorithms to Price American-Style Stock Options

by

Nathaniel James Pacey

A thesis submitted to the

Department of Physics, Engineering Physics, and Astronomy

in conformity with the requirements for

the degree of Bachelor of Applied Science

Queen's University

Kingston, Ontario, Canada

April 1st, 2023

Abstract

This thesis examines the effectiveness of Recurrent Neural Networks (RNNs), specifically Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models, and a hybrid GRU variant, in predicting American-style stock option prices. The research encompasses the development and evaluation of these models using historical options data from large technology companies from February to December 2022, aiming to refine prediction accuracy through systematic parameter tuning and architecture comparison.

Throughout this study, lower Mean Squared Error (MSE) and Mean Absolute Error (MAE) values were key in indicating a model's accuracy in predicting bid and ask prices. The findings indicate that the GRU model initially outperformed its counterparts, achieving an MSE of 46.59 and an MAE of 4.48. In contrast, the LSTM model recorded an MSE of 97.39 and an MAE of 6.82. The Hybrid GRU and Fully Connected model, designed to differentiate temporal from non-temporal data elements, had a MSE of 73.47 and an MAE of 6.00.

Expanding the dataset to include more tech companies, the GRU model continued to outperform the other models with a low MAE of 3.22 and bid and ask errors of 2.80% and 6.15%, respectively, on the testing dataset. A comparison with a more complex Hybrid GRU and Convolutional network revealed that increasing the complexity of model architecture to discern features of the data does not necessarily enhance prediction accuracy, further suggesting that simpler GRU models might be more effective in extracting the relevance of each input parameter.

Further research is needed to enhance the normalization of option contracts to manage diverse price volatilities. Improving this aspect could allow models to cover a wider array of options, potentially revealing broader market patterns. Overall, this research evaluates the performance of various machine learning architectures on options contracts, aiming to assess their potential as predictive tools for options traders. By identifying patterns and forecasting price movements, these models could aid traders in spotting arbitrage opportunities within large tech options, contributing valuable insights to the field of financial analytics.

Table of Contents

List of Tables

List of Figures

Acknowledgements

Reflecting on the famous line from *The Wolf of Wall Street*, "Nobody knows if a stock is going to go up, down, sideways, or in circles," perfectly captures the unpredictable essence of financial markets [1]. In this uncertain world, I'd like to thank Professor Ryan Martin for his clear guidance and unwavering support. He helped me navigate through the 'fugazi' of stock options and machine learning, making challenging concepts accessible and manageable.

1 Introduction and Motivation

Stock options are contracts allowing investors to buy or sell company shares at predetermined prices within a predetermined time period (term). In today's competitive trading market computer algorithms including machine learning algorithms account for 50 to 60% of stock and options trading on a typical day [2]. This project focuses on call and put options as leverage tools, enabling traders to control significant amounts of capital with minimal investment. Call options, suitable for expecting share price rises, allow buying a predefined number of shares (usually 100 per option) at a strike price before the contract expires. Conversely, put options, used when expecting share price falls, grant the right to sell a specific number of shares during the contract's term [3]. In addition to option types, two styles exist: American and European. American options allow exercise at any time during the contract, while European options only permit exercise on the expiration date [3]. Despite various attempts, accurately predicting stock option prices remains a challenge, particularly for American options where traditional models like the Black-Scholes formula falter due to their fixed expiration reliance [4]. The goal of this research is to create and evaluate machine learning models that can accurately predict prices of American-style options to address these challenges.

2 Problem Definition

This project utilizes TensorFlow to develop Recurrent Neural Networks (RNNs) in a many-to-many configuration to predict ask and bid prices for short-term tech stock call options with strike prices near the current market value (at the money). The study aims to improve price prediction accuracy by carefully adjusting parameters and exploring different network architectures, supported by continuous testing and validation. The goal is to develop a versatile model capable of handling various equities, achieved by applying normalization techniques and examining a range of network structures. An essential aspect of this project is the comparative evaluation of different machine learning frameworks, including Long Short-Term Memory (LSTM) networks, Gated Recurrent Unit (GRU) networks, and an innovative hybrid GRU model featuring a Fully Connected layer. This analysis aims to pinpoint the most suitable model for analyzing financial time series data.

3 Project Goals and Constraints

The goal is to develop a model capable of recognizing relationships among an option's features, historical outcomes, and market perceptions to accurately forecast ask and bid prices. Where accuracy is assessed through the mean squared error and the spread of prediction discrepancies. A decrease in prediction error percentage would indicate improved correlation and could help traders spot misvalued options and arbitrage chances. Spotting such arbitrage opportunities in highly liquid stocks can be extremely lucrative because of the leveraging effective inherent when trading in such options. However, reaching such accuracy is difficult because of market efficiency and the emphasis financial institutions place on precise pricing [5]. This project aims to refine pricing prediction accuracy through the use of Recurrent Neural Networks (RNNs), focusing on the systematic tuning of parameters and the comparison of different architectures [6]. This research provides insights that could extend time series data analysis and improve understanding of machine learning in financial analytics.

4 Background Information

4.1 Artificial Neurons the Basis of Machine Learning

The fundamental element of a machine learning model is the artificial neuron, which resembles a biological neuron in function. Arranged in layers, these neurons execute mathematical operations, processing input values weighted accordingly. They undergo linear transformations and are then subject to nonlinear activation functions, resulting in the final output [7].

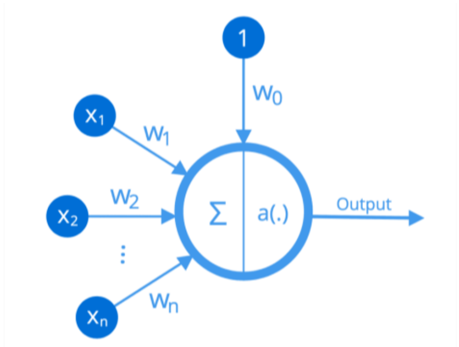


Figure 1. Single Neuron with weights Visualization taken from Machine Learning Mastery [7].

The transformation involves multiplying each input value x_i, by its corresponding weight w_i, summing the results, and adding a bias term b [7].

$$z = \left(\sum_{i=1}^{n} x_i \times w_i \right) + b \tag{1}$$

This process involves summing the weighted inputs to produce a value, z, which is then passed through a nonlinear activation function, represented as a, to produce the neuron's output, $\mathrm{a(z)}$. The use of a nonlinear activation function allows the model to handle complex patterns in the data by introducing nonlinearity [7].

$$a(z) = a\left(\sum_{i=0}^{n} x_i \times w_i \right) \tag{2}$$

A single neuron model is too simple for time series forecasting. To handle such complexity, the model needs a network of interconnected neurons. It also requires the use of techniques like backpropagation and gradient descent, along with cycling of inputs and outputs, all of which are features of a Recurrent Neural Network (RNN).

4.2 Recurrent Neural Networks and Time Series Forecasting

Recurrent Neural Networks (RNNs) are a type of artificial neural network tailored for processing sequential data, featuring the ability to remember past information through directed memory cycles, a characteristic not found in standard neural networks [8]. Typically, an RNN has three layers: input, hidden, and output. The input layer receives the sequential data, which is then processed in one or more hidden layers where activations are applied. Each hidden layer has its unique weights and biases, allowing the network to recognize different patterns in the data [8].

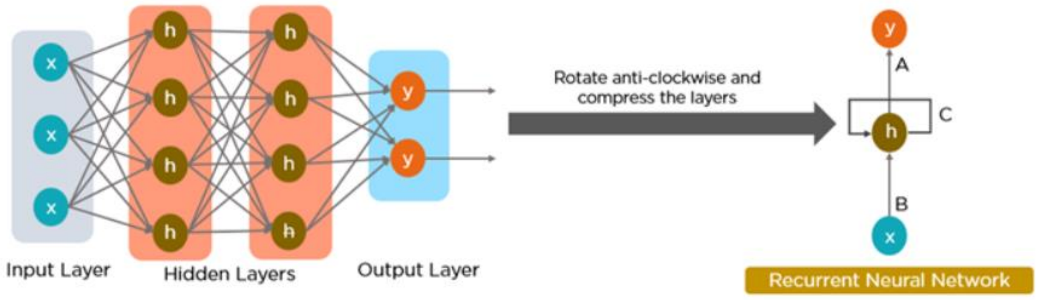


Figure 2. Visualization of a many to many RNN including the essential layers as taken from Analytics Vidhya [6].

RNNs are particularly effective in time series forecasting because they maintain the sequence's temporal connections. This is achieved through recurrent neurons that combine previous and current input states, thus revealing underlying patterns in data such as stock option prices [9].

4.3 Types of Recurrent Neural Networks

Recurrent Neural Networks (RNNs) face a challenge known as short-term memory due to the vanishing gradient problem, which affects their ability to carry information across long sequences [10]. This issue is particularly acute in lengthy sequences, where early information might be lost in later processing stages. The vanishing gradient issue arises during backpropagation, as gradients used for updating network weights decrease exponentially over time, leading to minimal learning in initial layers [10]. To overcome these limitations, advancements have been made, notably Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRU), which are designed to retain information over longer sequences of data [11].

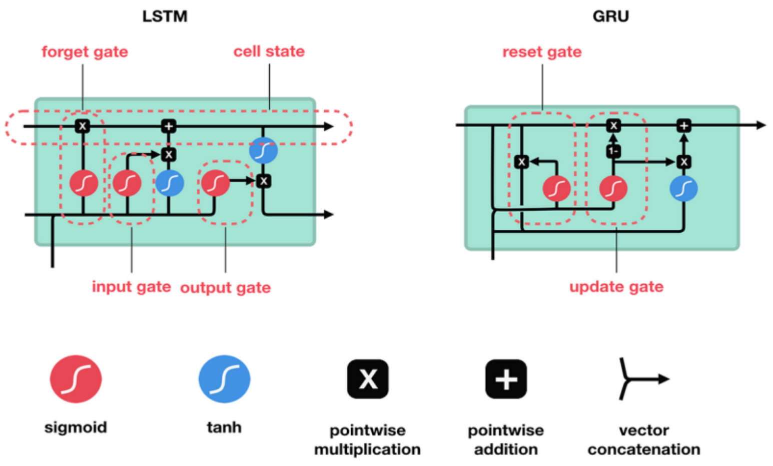


Figure 3. LSTM and GRU internal mechanisms as taken from an Illustrated Guide to LSTM's and GRU's [10].

Developed by Hochreiter & Schmidhuber in 1997, Long Short-Term Memory (LSTM) networks comprise crucial components that enhance data retention over long sequences [11]. The Input Gate incorporates new data into the memory state, filtering relevant information. The Forget Gate assesses this data to eliminate outdated parts from the memory [12]. The Cell State in LSTMs acts as a critical reservoir, preserving key information throughout the sequence. Subsequently, the Output Gate decides which information to pass on, shaping the next hidden state. This setup allows LSTMs to retain crucial data over long sequences, making them adept at handling complex temporal data.

Introduced by Kyunghyun Cho among other researchers in 2014, Gated Recurrent Units (GRUs) streamline the LSTM architecture, offering similar functionality with fewer parameters [13]. GRUs merge the LSTM's input and forget gates into a single Update Gate, which balances old and new information retention. The Reset Gate within the GRU cell fine-tunes how much past data is overlooked, enhancing memory management [13]. By integrating the cell state into the Hidden State, GRUs simplify the model, making them suitable for sequence modeling due to their computational efficiency and quicker training capabilities [10].

In both LSTM and GRU architectures, activation functions like sigmoid and tanh introduce necessary non-linearity for complex data pattern recognition. The sigmoid function, ranging from 0 to 1, is crucial in gating mechanisms, aiding in data retention decisions. Conversely, the tanh function, with outputs between -1 and 1, stabilizes network activations, preventing extreme value shifts [12]. This is pivotal in modulating the cell state in LSTMs and crafting new hidden states in both models. These activation functions equip LSTMs and GRUs to adeptly manage and conserve significant information over long data sequences, making them invaluable for analyzing historical context in fields such as natural language processing and time series forecasting [10].

5 Methodology

5.1 Data Extraction

Historical options data for 56 companies spanning from February to December 2022 was obtained using the yfinance Python library and stored on the Neutrino server as zip files. The `PrepareOptionsData.py` script processes this data by extracting it from compressed CSV files using `pandas`, which also handles date parsing. Options with strike price deviations exceeding ±50 from the stock price are filtered out to exclude far out-of-the-money options, focusing on those more likely to expire in the money. Additional features like "delta days" and weekdays are computed to provide temporal context, capturing trading activity patterns and weekly market cycles. For model training, input features such as historical stock prices, strike price, open interest, days to expiry, delta days, and weekdays are compiled from complete records within a specified look-back period. These features are aggregated into the input variable x, while the bid and ask prices are designated as the output variable y.

5.2 Data Encoding and Parsing

The `PrepareOptionsData.py` script utilizes the `produceXYDataSets` function to extract relevant features and target variables for each specified ticker. Extracted data is normalized using min-max techniques tailored for stock options to ensure uniform scaling while preserving critical relationships essential for effective model training. Following normalization, the data is organized and split into training and testing sets was executed via the `train_test_split` function from `sklearn`, adopting an 80/20 partition for training and testing, respectively, to facilitate model assessment. Subsequently, null values are checked to ensure data completeness and consistency before formatting it to meet the input specifications of the recurrent neural network model, adjusting the shapes of input parameters accordingly.

5.3 Model Development and Evaluation

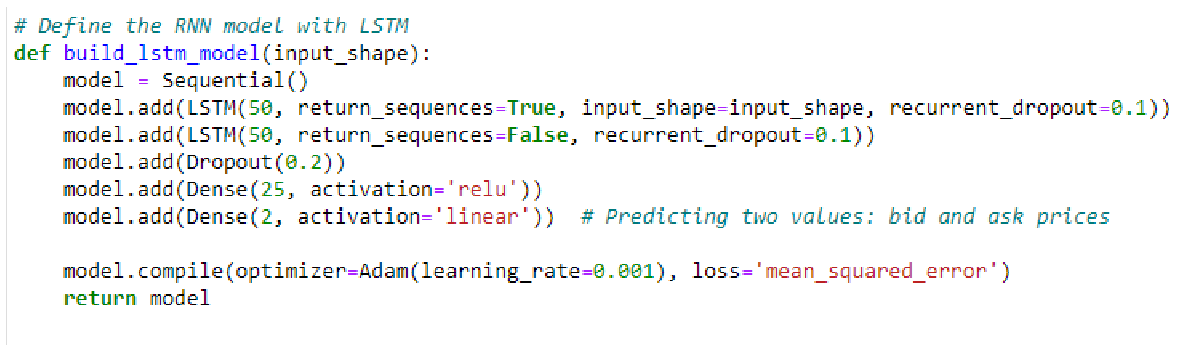
A variety of models, including LSTM, GRU, and a Hybrid GRU with Fully Connected layers, was constructed with uniform hyperparameters to ensure consistent evaluation in training and testing. The initial LSTM design, influenced by Israt Jahan's research, standardized the framework for subsequent model development using TensorFlow and Keras, capitalizing on established techniques for detecting temporal patterns in financial data [14].

Model effectiveness was gauged using mean squared error (MSE) and mean absolute error (MAE) metrics from the `sklearn.metrics` suite, noting that a lower MSE signifies more accurate predictions. Cross-validation was applied to prevent overfitting and ensure reliability on new data. Additionally, error percentage histograms and visual comparisons of predicted versus actual ask-bid spreads were generated to analyze model performance, distinguishing between outliers and overall effectiveness.

6 Models and Simulations

6.1 LSTM Model Development

The initial RNN model is comprised of two LSTM layers with 50 units each, along with a dropout layer to prevent overfitting. This architecture also includes two fully connected (dense) layers: one with 25 units using Rectified Linear Activation (ReLU) and a final layer with 2 units for predicting bid and ask prices with linear activation. The LSTM model is compiled using the Adam optimizer with a learning rate of 0.001, employing Mean Squared Error as the loss function [14].

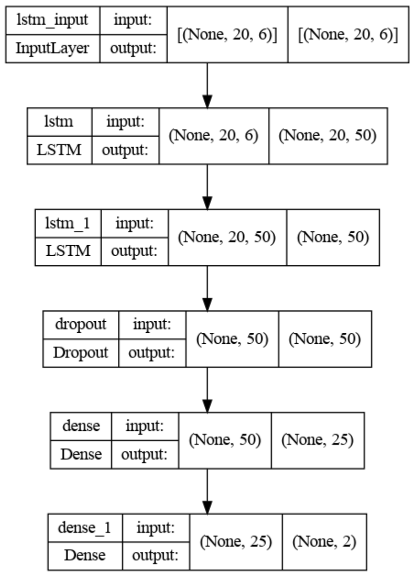


```python
# Define the RNN model with LSTM
def build_lstm_model(input_shape):
    model = Sequential()
    model.add(LSTM(50, return_sequences=True, input_shape=input_shape, recurrent_dropout=0.1))
    model.add(LSTM(50, return_sequences=False, recurrent_dropout=0.1))
    model.add(Dropout(0.2))
    model.add(Dense(25, activation='relu'))
    model.add(Dense(2, activation='linear'))  # Predicting two values: bid and ask prices

    model.compile(optimizer=Adam(learning_rate=0.001), loss='mean_squared_error')
    return model
```

Figure 4. LSTM ML model developed in python for this project.

This model structure can be visualized in the functional diagram as produced by TensorFlow's plot model function.



lstm_input | input: | [(None, 20, 6)] | [(None, 20, 6)]
InputLayer | output:
lstm | input: | (None, 20, 6) | (None, 20, 50)
LSTM | output:
lstm_1 | input: | (None, 20, 50) | (None, 50)
LSTM | output:
dropout | input: | (None, 50) | (None, 50)
Dropout | output:
dense | input: | (None, 50) | (None, 25)
Dense | output:
dense_1 | input: | (None, 25) | (None, 2)
Dense | output:

Figure 5. LSTM model architecture visualized using TensorFlow's plot_model function.

6.2 LSTM Initial Training and Validation

The initial model was trained and validated using options data from Apple and AMD, with the process spanning over 10 epochs. The initial model yielded an MSE of 21.04 and an MAE of 2.30. While the full dataset contains several outliers, an examination of the initial 100 samples shows that the model accurately tracks the actual bid and ask prices.

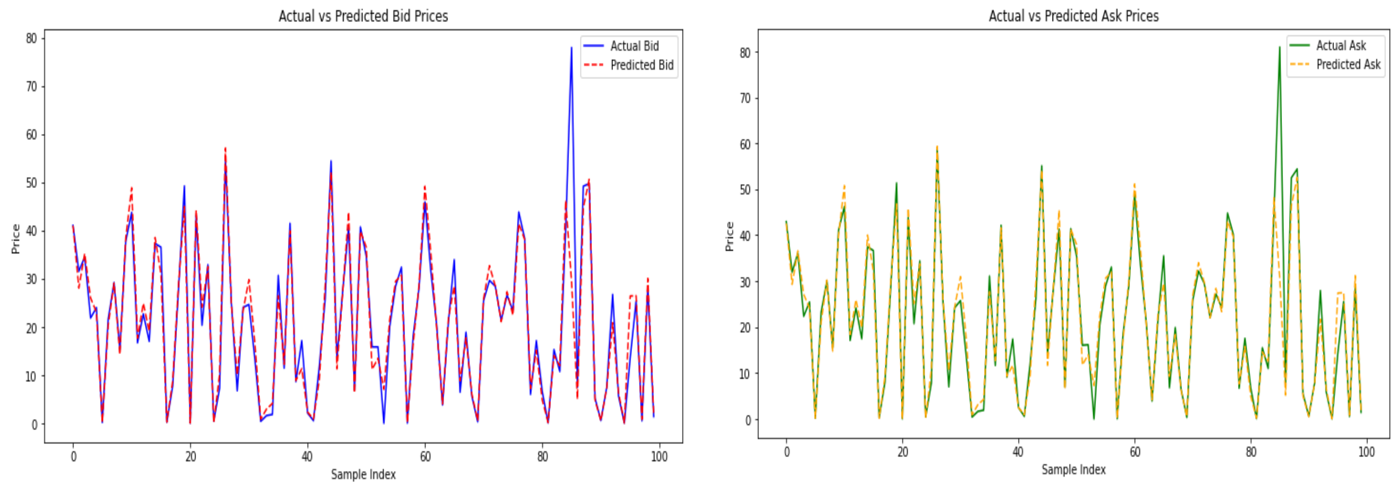


Figure 6. First 100 option contracts bid and ask prices and the respective LSTM model predictions for randomized testing data.

To gain deeper insights into the distribution of ask/bid predictions and their corresponding errors across the dataset, histogram plots were generated.

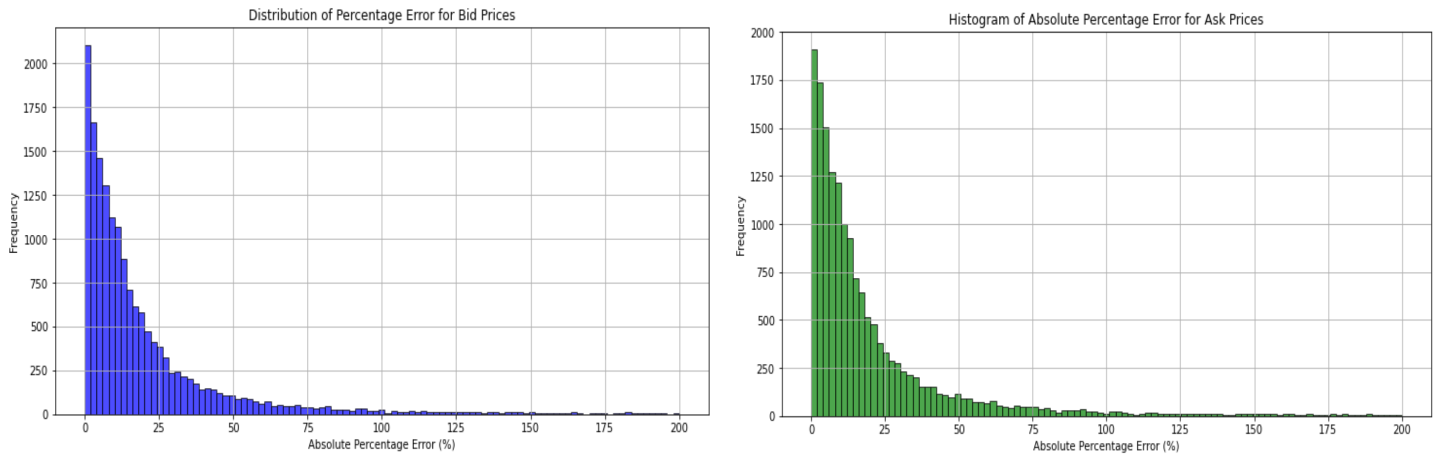
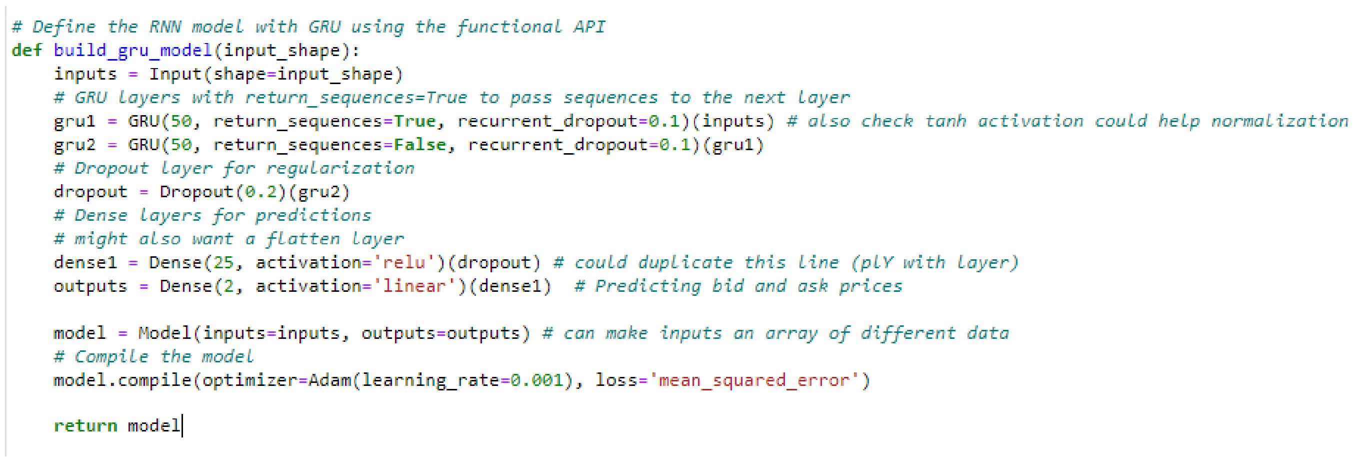


Figure 7. Histogram of the percent distribution of error for bid and ask predictions from the LSTM model on the randomized testing data.

Most errors in bid and ask predictions fall within a 20% margin, affirming the suitability of the chosen RNN model architecture and hyperparameters for this dataset. However, it's important to acknowledge that these results may not perfectly reflect real trading conditions. To realistically evaluate LSTM and other models, testing data was extracted from the final 20% of the dataset, avoiding random selection to enhance model performance assessment.

6.3 GRU Model Development

Following the development and analysis of the LSTM model, a comparable Gated Recurrent Unit (GRU) RNN model was constructed. Both models consist of two gated layers with 50 units and a recurrent dropout rate of 0.1, culminating in a dense output layer optimized with Adam optimizer set at a learning rate of 0.001.



```python
# Define the RNN model with GRU using the functional API
def build_gru_model(input_shape):
    inputs = Input(shape=input_shape)
    # GRU layers with return_sequences=True to pass sequences to the next layer
    gru1 = GRU(50, return_sequences=True, recurrent_dropout=0.1)(inputs) # also check tanh activation could help normalization
    gru2 = GRU(50, return_sequences=False, recurrent_dropout=0.1)(gru1)
    # Dropout layer for regularization
    dropout = Dropout(0.2)(gru2)
    # Dense layers for predictions
    # might also want a flatten layer
    dense1 = Dense(25, activation='relu')(dropout) # could duplicate this line (plY with layer)
    outputs = Dense(2, activation='linear')(dense1)  # Predicting bid and ask prices

    model = Model(inputs=inputs, outputs=outputs) # can make inputs an array of different data
    # Compile the model
    model.compile(optimizer=Adam(learning_rate=0.001), loss='mean_squared_error')

    return model
```

Figure 8.GRU model developed in Python to compare to the LSTM model utilizing a similar structure of layers and hyper parameters.

In the GRU model, the first gated layer preserves temporal information by returning sequences, enabling the second gated layer to generate a singular context vector as its output.

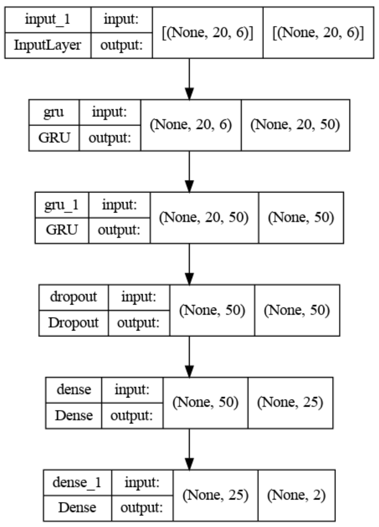

input_1 | InputLayer | input: [(None, 20, 6)] | output: [(None, 20, 6)]
gru | GRU | input: (None, 20, 6) | output: (None, 20, 50)
gru_1 | GRU | input: (None, 20, 50) | output: (None, 50)
dropout | Dropout | input: (None, 50) | output: (None, 50)
dense | Dense | input: (None, 50) | output: (None, 25)
dense_1 | Dense | input: (None, 25) | output: (None, 2)

Figure 9. TensorFlow functional visualization of the GRU based RNN model.

A dropout layer with a rate of 0.2 is introduced after the GRU layers to mitigate overfitting. Following this, the network transitions to a dense layer with 25 neurons employing the ReLU activation function. ReLU introduces non-linearity by outputting zero for negative inputs and retaining positive values, aiding the model in capturing complex data patterns [15]. Finally, the model concludes with a final output layer comprising two units using a linear activation function, tasked with predicting bid and ask prices.

6.4 LSTM and GRU Model Evaluations

Although machine learning models inherently exhibit variability, meaningful comparisons between the LSTM and the initial GRU model's performance can be drawn. Both models underwent training and testing for 10 epochs on Apple and AMD contracts over identical time frames. To ensure an accurate evaluation of predictive accuracy under market-like conditions, the final 20% of the dataset was reserved for testing, analyzing the percentage error distribution for predicted bid and ask prices.

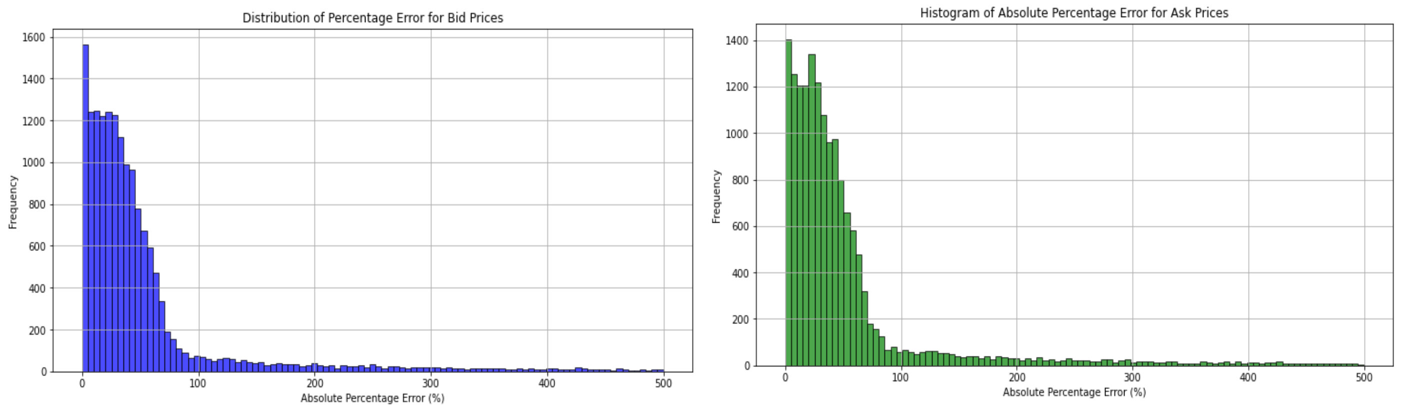


Figure 10. LSTM model performance of ask and bid predictions on realistic data split.

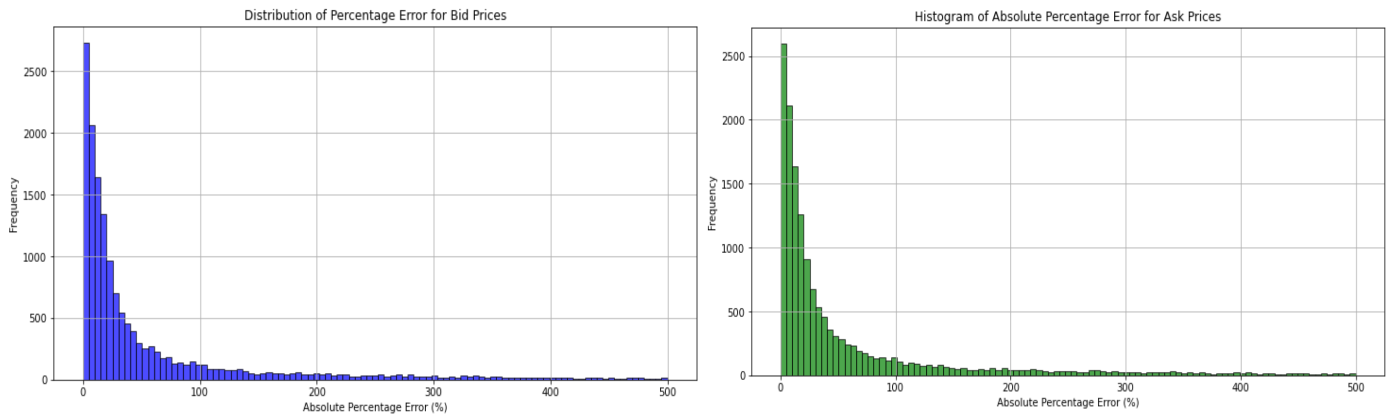


Figure 11. GRU model bid and ask error performance on realistic data split.

The LSTM model achieved an MSE of 97.39 and MAE of 6.82, with average bid and ask errors of 46.68% and 26.50%, respectively. The GRU model, however, showed improved performance with an MSE of 46.59 and MAE of 4.48, and lower average bid and ask errors of 7.86% and 12.95%, indicating its superior forecasting accuracy for the testing dataset. To further evaluate the models' effectiveness, assessments were conducted using a new dataset, specifically Amazon options over the same duration.

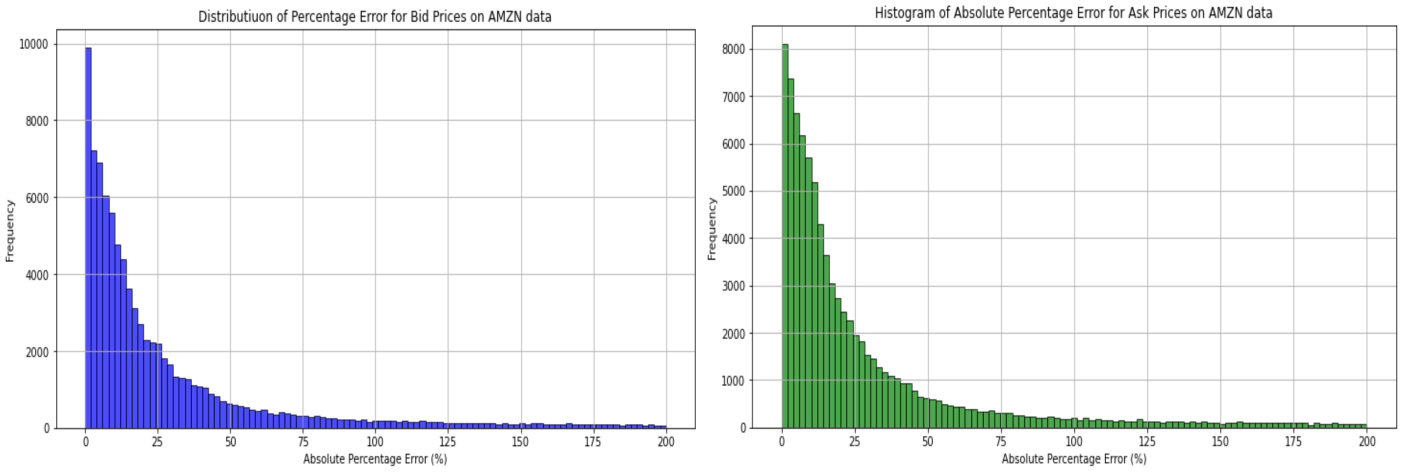


Figure 13. LSTM bid and ask error frequency over Amazon data.

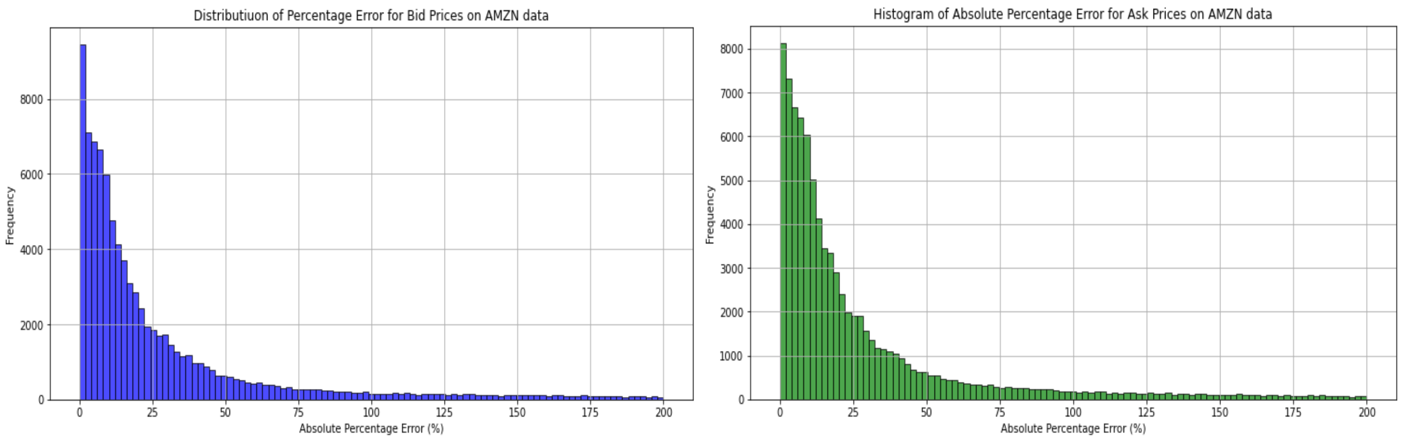


Figure 12. GRU bid and ask error frequency over Amazon data.

For the Amazon options, both models generally displayed prediction errors below 25%, with a significant portion under 10%. The LSTM model yielded a MSE of 39.12 and MAE of 3.56, alongside bid and ask errors at 31.48% and 25.16%, respectively. Conversely, the GRU model demonstrated a MSE of 27.06 and MAE of 3.17, with bid and ask errors reaching 84.39% and 114.15%, respectively. Despite variations, both models exhibit low MSE and MAE values, indicating strong alignment with actual bid and ask prices, evident in the analysis of the initial 100 Amazon options contracts.

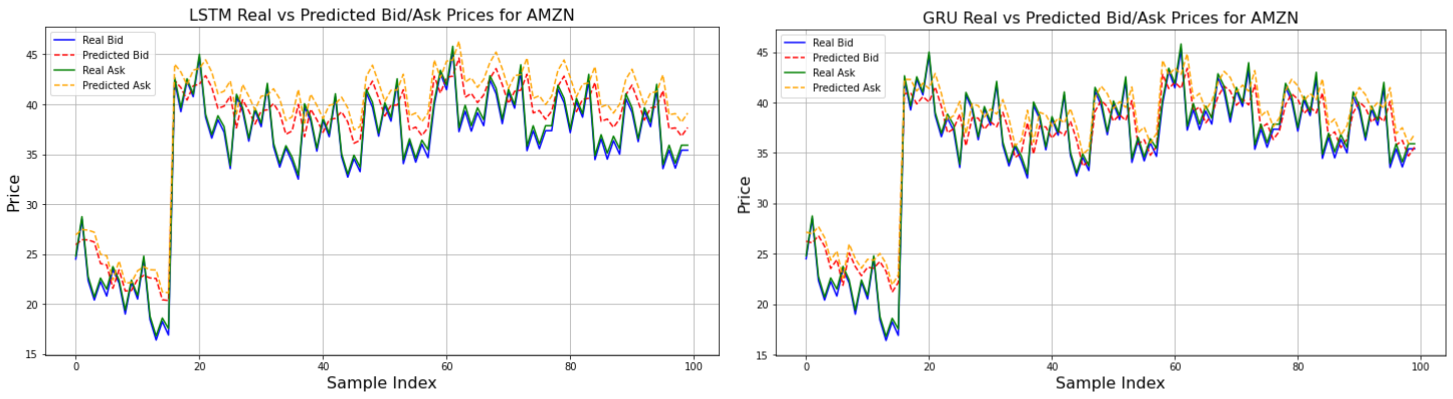


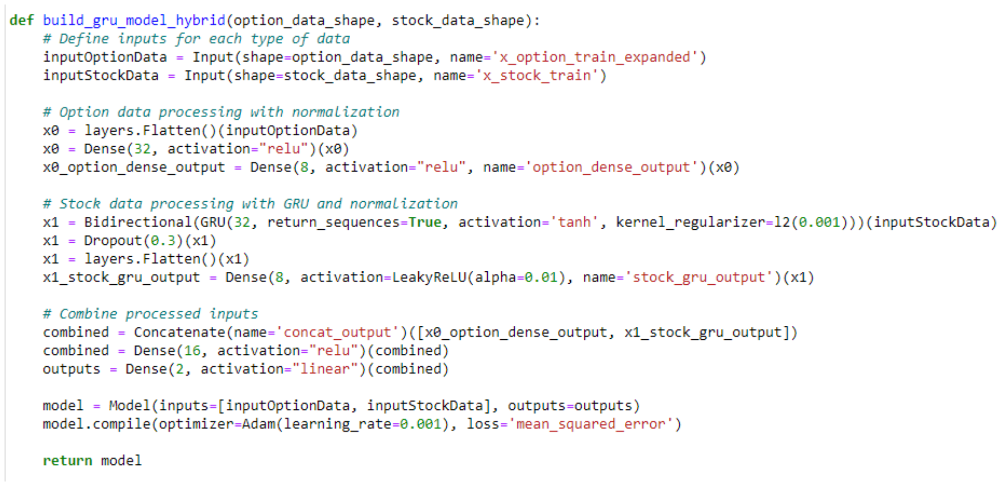
Figure 14. LSTM and GRU predicted compared to actual bid and ask prices for the first 100 AMZN option contracts.

The LSTM model's 52.91% and 88.99% reduction in bid and ask errors compared to the GRU model on the Amazon dataset is surprising given previous results but aligns with the expected variability inherent in machine learning trials. These results suggest that the GRU model, while demonstrating a better overall fit to the data, may have significant outliers in its bid and ask price predictions, accounting for the observed discrepancy.

7 Model Iteration

7.1 Hybrid GRU and Fully Connected Network Development

In an effort to improve the GRU model's effectiveness, a hybrid model was designed to distinguish between temporal and static elements. This hybrid model aligns with previous models, employing a learning rate of 0.001, utilizing MSE as the loss function, and leveraging the Adam optimizer. Input data is processed through two distinct pathways: one for option-related features and another for stock-related features. Within the option data pathway, a dense layer manages time-independent inputs such as contract names, extracting patterns not reliant on temporal sequences.



```
def build_gru_model_hybrid(option_data_shape, stock_data_shape):
    # Define inputs for each type of data
    inputOptionData = Input(shape=option_data_shape, name='x_option_train_expanded')
    inputStockData = Input(shape=stock_data_shape, name='x_stock_train')

    # Option data processing with normalization
    x0 = layers.Flatten()(inputOptionData)
    x0 = Dense(32, activation="relu")(x0)
    x0_option_dense_output = Dense(8, activation="relu", name='option_dense_output')(x0)

    # Stock data processing with GRU and normalization
    x1 = Bidirectional(GRU(32, return_sequences=True, activation='tanh', kernel_regularizer=l2(0.001)))(inputStockData)
    x1 = Dropout(0.3)(x1)
    x1 = layers.Flatten()(x1)
    x1_stock_gru_output = Dense(8, activation=LeakyReLU(alpha=0.01), name='stock_gru_output')(x1)

    # Combine processed inputs
    combined = Concatenate(name='concat_output')([x0_option_dense_output, x1_stock_gru_output])
    combined = Dense(16, activation="relu")(combined)
    outputs = Dense(2, activation="linear")(combined)

    model = Model(inputs=[inputOptionData, inputStockData], outputs=outputs)
    model.compile(optimizer=Adam(learning_rate=0.001), loss='mean_squared_error')

    return model
```

Figure 15. Hybrid GRU & Fully Connected Network implementation code utilizing TensorFlow built in functions.

The stock data pathway employs a bidirectional GRU layer with 32 units, leveraging the ReLU activation function to account for temporal dependencies in stock prices. A recurrent dropout rate of 0.1 is incorporated within the GRU layers to mitigate overfitting. The model also includes a layer normalization step to normalize the input data, ensuring the model remains effective even when the stock and options experience new highs and lows outside the training set's range. After processing through these layers, both pathways converge, and the combined data is passed through additional dense layers, culminating in the final output layer that predicts bid and ask prices.

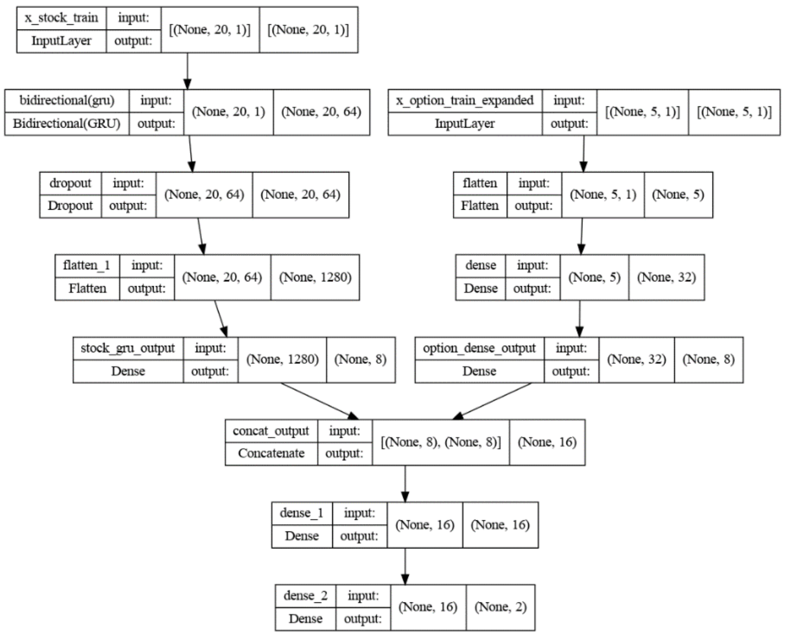

x_stock_train | InputLayer | input: | output: | [(None, 20, 1)] | [(None, 20, 1)]
bidirectional(gru) | Bidirectional(GRU) | input: | output: | (None, 20, 1) | (None, 20, 64)
x_option_train_expanded | InputLayer | input: | output: | [(None, 5, 1)] | [(None, 5, 1)]
dropout | Dropout | input: | output: | (None, 20, 64) | (None, 20, 64)
flatten | Flatten | input: | output: | (None, 5, 1) | (None, 5)
flatten_1 | Flatten | input: | output: | (None, 20, 64) | (None, 1280)
dense | Dense | input: | output: | (None, 5) | (None, 32)
stock_gru_output | Dense | input: | output: | (None, 1280) | (None, 8)
option_dense_output | Dense | input: | output: | (None, 32) | (None, 8)
concat_output | Concatenate | input: | output: | [(None, 8), (None, 8)] | (None, 16)
dense_1 | Dense | input: | output: | (None, 16) | (None, 16)
dense_2 | Dense | input: | output: | (None, 16) | (None, 2)

Figure 16. Hybrid GRU visualization using TensorFlow plot_model.

7.2 Hybrid GRU Results

The hybrid model underwent training using the stock options data for Apple and AMD, extending over 100 epochs. During this training phase, the best-performing model iteration was preserved using checkpoint mechanisms. A training and validation loss plot, presented on a logarithmic scale and covering all 100 epochs, was created to assess the model's learning progression and its ability to reduce loss over time.

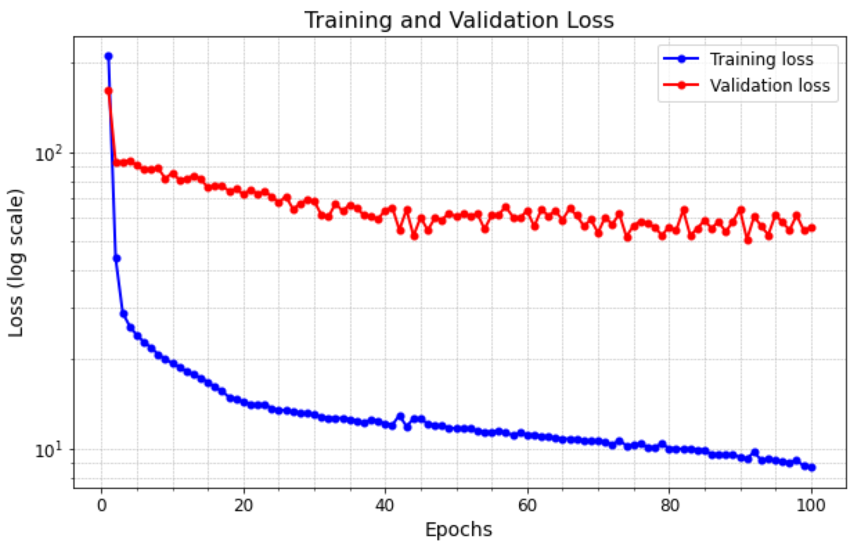


Figure 17. Hybrid GRU Fully Connected Network training and validation loss over the span of 100 training epochs.

The Hybrid GRU model performed worse than the earlier GRU model on the testing dataset, showing a Mean Squared Error of 73.47 and a Mean Absolute Error of 6.00. To understand whether the model consistently overvalued or undervalued option prices, an analysis of the percent error distribution was conducted. This analysis looked at both positive and negative percent errors for bid and ask prices, aiming to identify any systematic bias in the model's predictions.

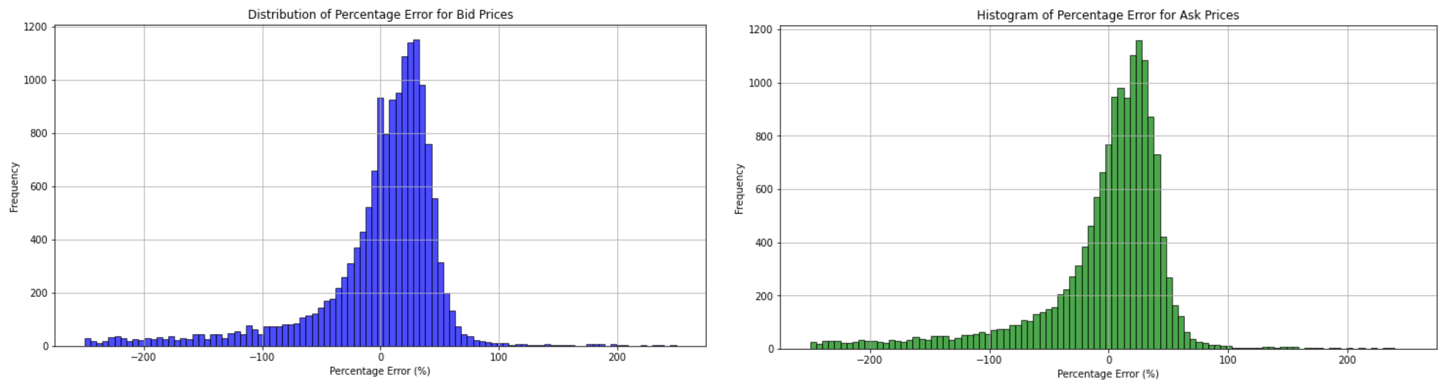


Figure 19. Distribution of predicted to real percentage error bid and ask prices.

The analysis indicates that the model's predictions for both bid and ask prices consistently overestimate, with the distribution peaking around 25 to 30% above the actual option prices. This trend may stem from systematic issues that adjusting the model's parameters could address or could result from the overall larger error associated with the Hybrid Model's predictions. This tendency is further highlighted by examining the absolute distribution of percentage error between the model's predicted bid and ask prices and the actual values.

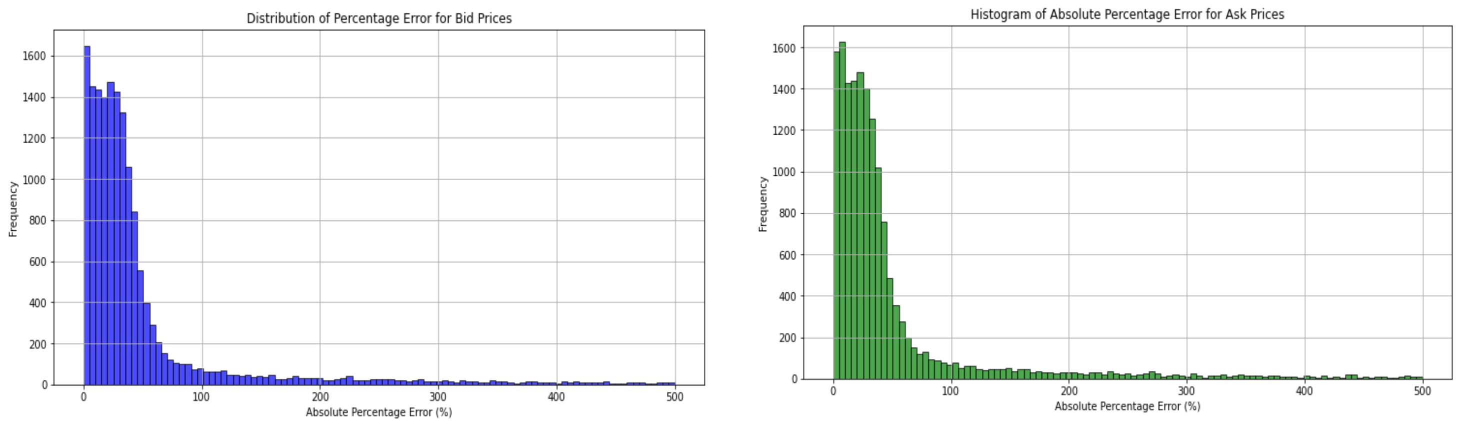


Figure 18. Absolute percent error distribution of predicted to actual bid and ask prices.

The error distribution for the Hybrid GRU model on the testing data reveals a notably higher frequency of errors ranging from 20 to 60%, indicating a weaker correlation and reduced effectiveness compared to the standalone GRU and LSTM models. This decreased performance is particularly evident in the model's predictions on the Amazon dataset, where bid and ask errors are notably elevated, reaching 227.03% and 170.94% respectively, with a Mean Squared Error of 46.79 and a Mean Absolute Error of 4.66.

7.3 Additional GRU Model Evaluation

Continuing from the promising results of the GRU-only model, additional evaluation was undertaken by integrating options contracts from Microsoft and Google, alongside those from AMD and Apple. These selections were based on their similarities, providing a focused dataset representative of major tech companies' options contracts. The objective was to enhance the model's ability to discern sector-specific predictive patterns. Training extended over 100 epochs, with only the iteration exhibiting the lowest loss retained. However, discernible improvements ceased after approximately 15 epochs, prompting a decision to limit further testing to just 20 epochs.

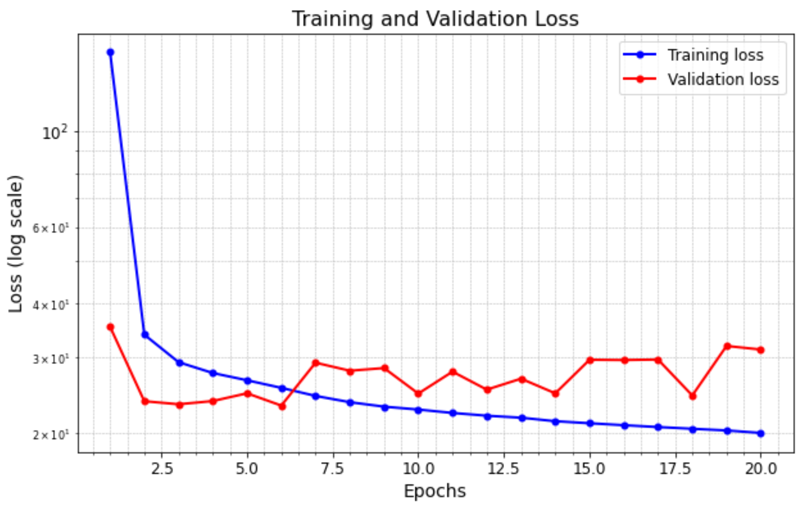


Figure 20. GRU only training losses with additional stock contracts over 20 epochs.

The additional option contracts and adjustments to the testing epochs led to notable enhancements in the model's performance. On the testing data, the Mean Squared Error was reduced to 21.21, and the Mean Absolute Error was brought down to 3.22. Specifically, the bid and ask errors were measured at 2.80% and 6.15%, respectively. Detailed visualizations of the percent error distribution for this testing data are accessible in Appendix B.

When applied to Amazon data, the model continued to show promising results with bid and ask errors recorded at 12.24% and 7.23%, respectively with a Mean Squared Error and Mean Absolute Error for this dataset stood at 22.64 and 2.86, further affirming the model's improved accuracy and efficacy in forecasting option prices. The model was further evaluated against the Amazon option contracts of which the distribution of percentage errors can be seen.

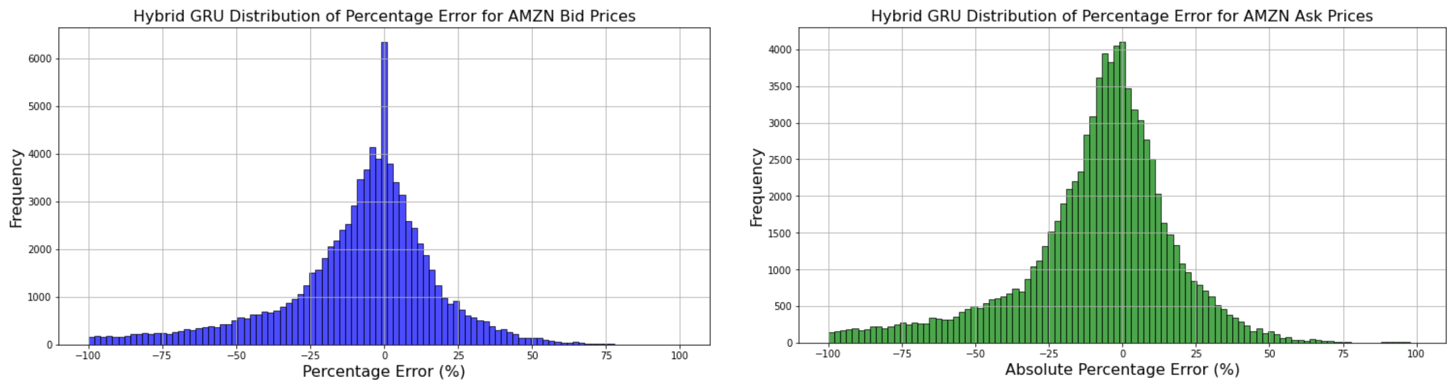


Figure 21. Percent error distribution of the GRU model's predicted bid and ask prices after training on additional options.

The error distribution suggests a uniform spread across bid and ask price predictions, with approximately 75% of predictions falling within 25% of the actual bid or ask price, reflecting a strong correlation in the model's forecasting relative to the actual price spread. The model's accuracy is underscored in its performance on the first 100 Amazon contracts.

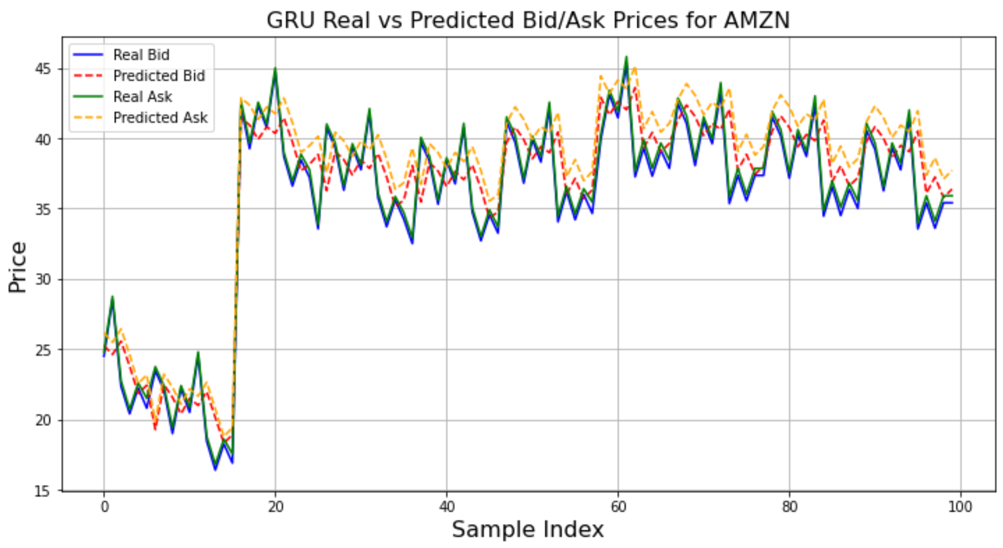


Figure 22. GRU model's predictions on the first 100 Amazon option contracts after being trained on additional options.

The close agreement between predicted and actual bid and ask prices, coupled with the low MAE, indicates a high level of forecasting precision. As well, the model consistently undervalues peaks and overvalues troughs in option prices, suggesting its effectiveness in identifying potential arbitrage opportunities. This consistent pattern in the model's predictions can be valuable for making strategic trading decisions.

8 Results and Discussion

8.1 Results Comparison

A range of neural network architectures, was systematically tested and compared using similar parameters against a Hybrid GRU and Convolutional Network developed by Professor Ryan Martin. The specific performance and structure of this reference model are detailed in Appendix B. The summarized results below reflect each model's performance on critical metrics when trained on AMD and Apple data and subsequently tested on these and additional Amazon data.

Table 1. Key results of each RNN model trained on Apple and AMD options coloured according to performance.

Models Trained on AMD and Apple	**Testing Data**				**Amazon Data**			
Model Type	**Bid Error**	**Ask Error**	**MSE**	**MAE**	**Bid Error**	**Ask Error**	**MSE**	**MAE**
LSTM	46.68%	26.50%	97.39	6.82	31.47%	25.16%	39.12	3.56
GRU	7.86%	12.95%	46.59	4.48	84.39%	114.15%	27.06	3.17
GRU and FC Hybrid	19.15%	1.81%	73.47	6.00	227.03%	170.94%	46.79	4.66
GRU and Convolutional Hybrid	29.54%	38.13%	79.25	6.17	121.19%	124.05%	40.61	4.01

The GRU model outperformed others on the initial test set with the lowest bid, ask errors, and MAE, yet it showed an uptick in prediction errors when applied to Amazon data. The LSTM yielded moderate performance consistently across datasets. While the standalone GRU model registered increased bid and ask errors for Amazon, a closer look at individual option predictions revealed a performance on par with the LSTM model. This observation suggests that outliers are influencing the perceived accuracy. In contrast, the hybrid models combining GRU with FC and convolutional layers incurred significantly higher errors on Amazon data, possibly due to a misalignment with this dataset's unique attributes. Particularly, the GRU and Convolutional Hybrid model's inflated error rates on Amazon data suggest a heightened sensitivity to specific option pricing characteristics. Across multiple evaluations, the Mean

Absolute Error on Amazon data emerged as a reliable indicator of model performance, especially since the models were not trained on Amazon data. The consistency of the MAE supports its use as a gauge for how well models generalize to new, unseen data.

To rigorously assess each model's ability to identify trends and forecast bid and ask prices accurately, the datasets were broadened to cover option contracts from Microsoft and Google, alongside AMD and Apple. This broadening was intended to confirm the GRU-only model's consistent performance across a variety of data, rather than being confined to a single dataset. The development and evaluation process for each model followed the same approach as that detailed for the GRU-only model in section 7.3.

Table 2. Key results of each RNN model trained on Apple, AMD, Microsoft and Google options colored according to performance.

Models Trained on AMD, Apple, Microsoft and Google	**Testing Data**				**Amazon Data**			
Model Type	**Bid Error**	**Ask Error**	**MSE**	**MAE**	**Bid Error**	**Ask Error**	**MSE**	**MAE**
LSTM	6.34%	4.41%	108.75	6.63	1.50%	62.31%	43.39	3.74
GRU	2.80%	1.81%	21.21	3.22	12.23%	7.23%	22.64	2.87
GRU and FC Hybrid	0.53%	1.58%	27.33	3.94	10.21%	3.93%	25.62	3.12
GRU and Convolutional Hybrid	31.39%	30.50%	54.18	6.36	75.73%	77.97%	28.60	3.63

The GRU-only model had the lowest Mean Absolute Error on both testing and Amazon datasets and displayed a balanced error distribution for both bid and ask predictions. The LSTM model, while showing low bid error rates on Amazon data, had higher ask error rates and overall higher MSE and MAE on the test set. The GRU and Fully Connected Hybrid model marked a significant performance improvement, with lower bid and ask errors compared to the GRU-only model, while maintaining a similar MAE on both datasets. However, the GRU and Convolutional Hybrid model lagged behind, turning in the weakest performance of the group. Across the board, all models demonstrated enhancements in nearly every evaluated metric, suggesting that the incorporation of additional data from similar stocks positively influenced the models' forecasting capabilities.

8.2 Discussion

This study's exploration of Recurrent Neural Network (RNN) models revealed the intricate nature of financial data's influence on machine learning performance. Despite the variability, all models showcased predictive capabilities, affirmed by their performance on both testing and Amazon datasets. Interestingly, the straightforward GRU model surpassed both the LSTM and more elaborate hybrid models, suggesting that complexity does not guarantee superior predictions. The GRU model's success underscores the effectiveness of simple architectures in capturing key data patterns, highlighting the value of algorithmic autonomy in identifying relevant features and patterns that might elude human designers. This emphasizes the significance of simplicity and data-driven approaches in the realm of financial time series forecasting.

9 Limitations and Extensions

9.1 Project Limitations

This study faced notable limitations that affected its scope and results. Conducted over eight months amidst other academic commitments, the time constraint limited the depth of analysis and experimentation. Technical difficulties with the Neutrino Server, essential for GPU-based code execution, further impeded progress by consuming valuable time for debugging, slowing down the development and iteration process. Additionally, the complex intersection of artificial intelligence (AI) and stock options prediction presented a significant learning curve. A considerable portion of the initial phase was devoted to studying relevant literature, which restricted hands-on model experimentation and refinement.

9.2 Extensions and Future Work

The evaluation of several models was extended to incorporate additional option contracts from companies such as Tesla, Nvidia, and Netflix, in addition to the previously considered Google, Microsoft, AMD, and Apple. However, this expansion yielded notably inferior results attributed to challenges related to stock splits and the wider variance in option pricing. These undesirable results are illustrated in the training versus validation loss for the Hybrid GRU and Fully Connected model.

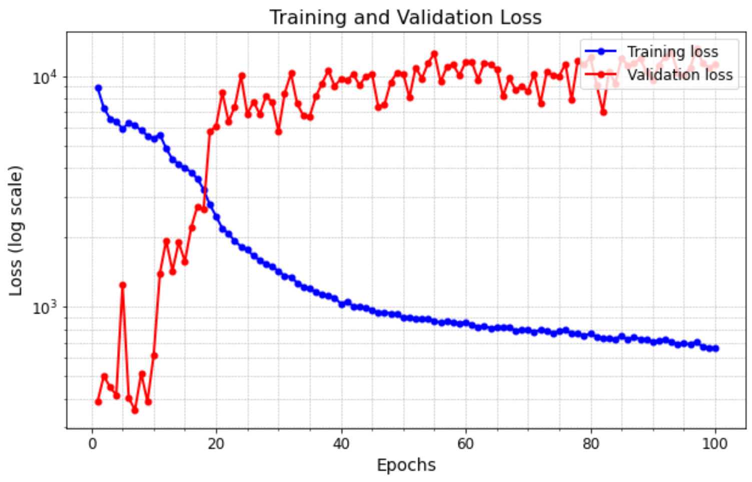


Figure 23. Training and validation loss associated with the Hybrid GRU and Fully Connected model utilizing several new options.

This indicated that the model struggled to identify any discernible pattern or reduce validation loss after each epoch, indicating a lack of correlation from the model's predictions to the underlying dataset. This discrepancy could be attributed to the fact that while options for AMD, Apple, Google, and Microsoft, all priced under $100 and displayed similar behavior, the inclusion of higher-priced options resulted in substantial deviations from real prices. This discrepancy was illustrated in the plot below, depicting bid predictions for the Hybrid GRU FC model.

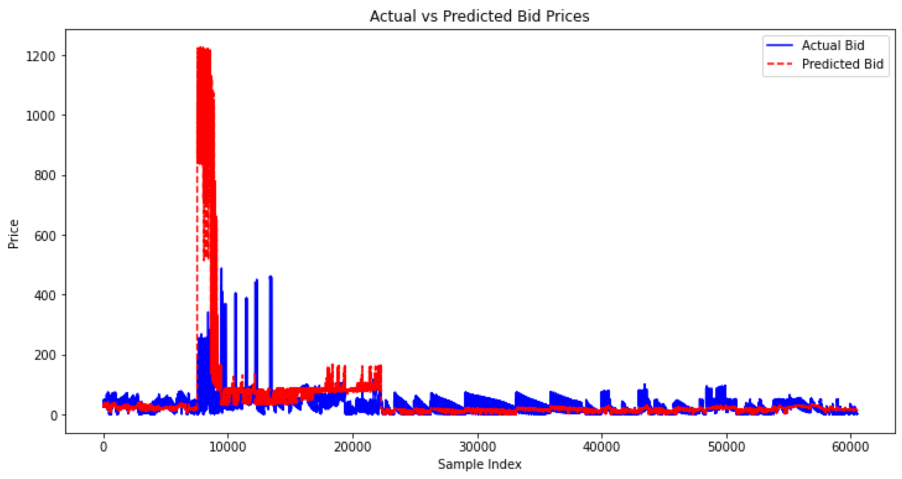


Figure 24. Actual versus predicted bid prices from the Hybrid GRU and FC model with the inclusion of several additional options.

In an effort to mitigate errors stemming from this issue, normalization was performed individually for each contract. This approach aimed to eliminate the model's dependence on absolute price values, instead emphasizing the relative scale of each parameter. However, upon implementation, the predicted values converged toward a single value, as evident from the results.

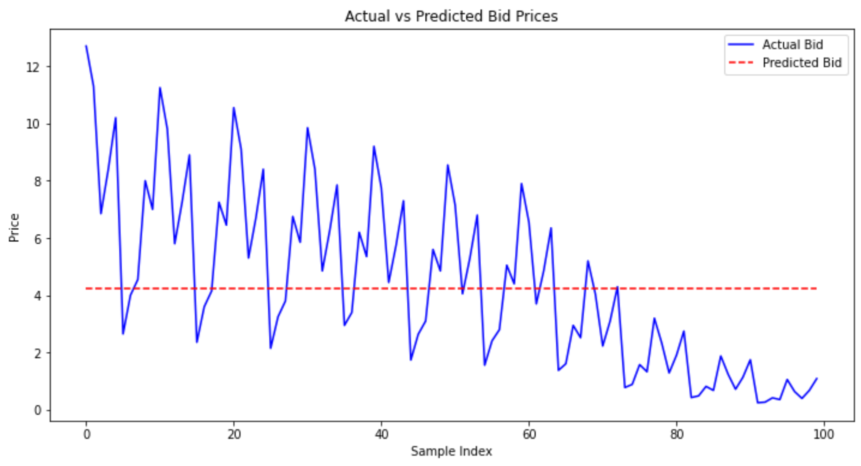


Figure 25. The effect of improper normalization techniques on the predicted bid prices from the Hybrid GRU FC model.

To enhance the developed model, it is crucial to establish a more dynamic and specific method for normalizing the input data to avoid losing essential training information. Furthermore, broadening the training and testing dataset to include a wider range of options would enable the model to adapt to various patterns, potentially resulting in more accurate predictions. This increased accuracy could make the model a more valuable tool for traders as it could enable them to identify potential arbitrage opportunities across a larger range of option contracts.

10 Conclusion

This thesis compared the performance of various Recurrent Neural Network (RNN) architectures in pricing American-style stock options, using consistent hyperparameters and the same datasets for training and testing. The study found that several models demonstrated predictive accuracy, which could be useful for options traders in identifying arbitrage opportunities.

Among the key results, the GRU model stood out, delivering a Mean Squared Error (MSE) of 46.59 and Mean Absolute Error (MAE) of 4.48, outperforming the LSTM model which yielded an MSE of 97.39 and an MAE of 6.82 on the testing dataset. The hybrid GRU model's performance was mixed, with an MSE of 73.47 and an MAE of 6.00, pointing to potential areas for optimization. Notably all models showed improved prediction accuracy after the introduction of additional options (Google and Microsoft) to the training and testing dataset. The GRU-only model consistently outperformed, exhibiting a low MAE of 3.22 and bid and ask errors of 2.80% and 6.15%, respectively. Moreover, comparisons with more complex Hybrid GRU networks indicate that increasing model complexity doesn't always enhance prediction accuracy, underscoring the effectiveness of simpler GRU model in extracting relevant input parameters for option pricing analysis. Despite the inherit variability of machine learning trials, the GRU-only model consistently demonstrated more accuracy in the prediction of bid and ask prices making it a more advantageous tool for traders to identify arbitrage opportunities compared to the other models.

Expanding the dataset to include a wider range of options may improve the model's adaptability to various market patterns, leading to more accurate predictions. This enhanced accuracy can offer traders valuable insights, aiding in the identification of arbitrage opportunities across a broader array of option contracts.

In conclusion, this study adds to the existing research on stock price prediction and derivative options by providing a thorough comparison of various RNN architectures. It emphasizes the effectiveness of RNNs, specifically the GRU-only model, in stock options pricing, offering a basis for further development in the field of financial analytics.

Appendix A. Statement of Work

Before September 1st, Professor Ryan Martin conducted research related to this project, focusing on the analysis of stock options. This work involved acquiring historical data for 56 American and Canadian companies from February 2022 to December 2022 using the yfinance Python library. This data was used throughout the duration of the project, stored in zip files on the Neutrino server. Data processing tasks, such as excluding out-of-the-money options, were executed using the pandas library. Additionally, Professor Martin developed a hybrid GRU and Fully Connected machine learning model, including data parsing, encoding, and decoding algorithms. This preliminary work provided a foundational guideline for this project's development.

In the Fall term, the project began with studying stock options and Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) networks for their time-series data handling capabilities. An initial LSTM model was developed, focusing on data encoding, decoding, and realistic data split strategies to ensure practical applicability.

In the Winter term, attention shifted to developing and evaluating a Gated Recurrent Unit (GRU) model against the LSTM model. A Hybrid GRU Fully Connected model was then constructed to potentially improve accuracy. The project also included testing a Hybrid GRU Convolutional model from Professor Ryan Martin. Performance optimization was pursued by adjusting parameters such as stock numbers, training epochs, and learning rates across all models to enhance their real-world utility.

Appendix B. Additional Figures

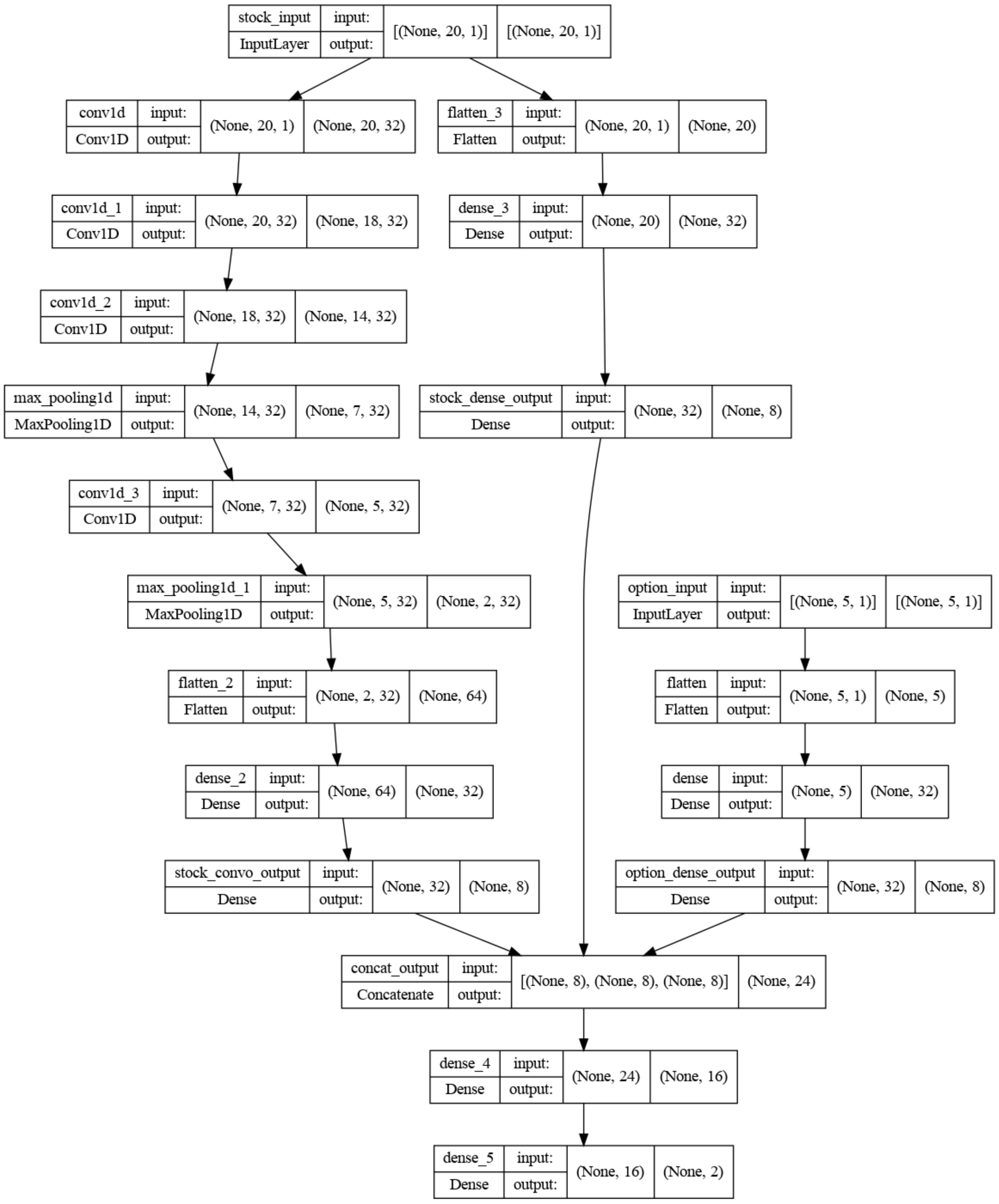
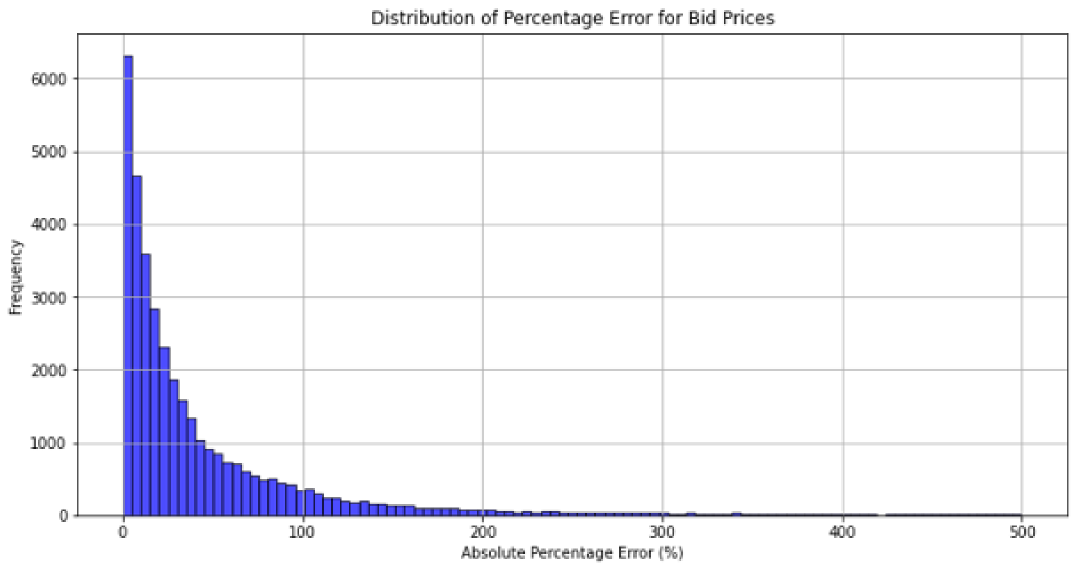

stock_input | InputLayer | input: | output: | [(None, 20, 1)] | [(None, 20, 1)]
conv1d | Conv1D | input: | output: | (None, 20, 1) | (None, 20, 32)
flatten_3 | Flatten | input: | output: | (None, 20, 1) | (None, 20)
conv1d_1 | Conv1D | input: | output: | (None, 20, 32) | (None, 18, 32)
dense_3 | Dense | input: | output: | (None, 20) | (None, 32)
conv1d_2 | Conv1D | input: | output: | (None, 18, 32) | (None, 14, 32)
max_pooling1d | MaxPooling1D | input: | output: | (None, 14, 32) | (None, 7, 32)
stock_dense_output | Dense | input: | output: | (None, 32) | (None, 8)
conv1d_3 | Conv1D | input: | output: | (None, 7, 32) | (None, 5, 32)
max_pooling1d_1 | MaxPooling1D | input: | output: | (None, 5, 32) | (None, 2, 32)
option_input | InputLayer | input: | output: | [(None, 5, 1)] | [(None, 5, 1)]
flatten_2 | Flatten | input: | output: | (None, 2, 32) | (None, 64)
flatten | Flatten | input: | output: | (None, 5, 1) | (None, 5)
dense_2 | Dense | input: | output: | (None, 64) | (None, 32)
dense | Dense | input: | output: | (None, 5) | (None, 32)
stock_convo_output | Dense | input: | output: | (None, 32) | (None, 8)
option_dense_output | Dense | input: | output: | (None, 32) | (None, 8)
concat_output | Concatenate | input: | output: | [(None, 8), (None, 8), (None, 8)] | (None, 24)
dense_4 | Dense | input: | output: | (None, 24) | (None, 16)
dense_5 | Dense | input: | output: | (None, 16) | (None, 2)

Figure 26. Visual diagram of the additional Hybrid GRU and Convolutional Network as developed by Professor Ryan Martin.



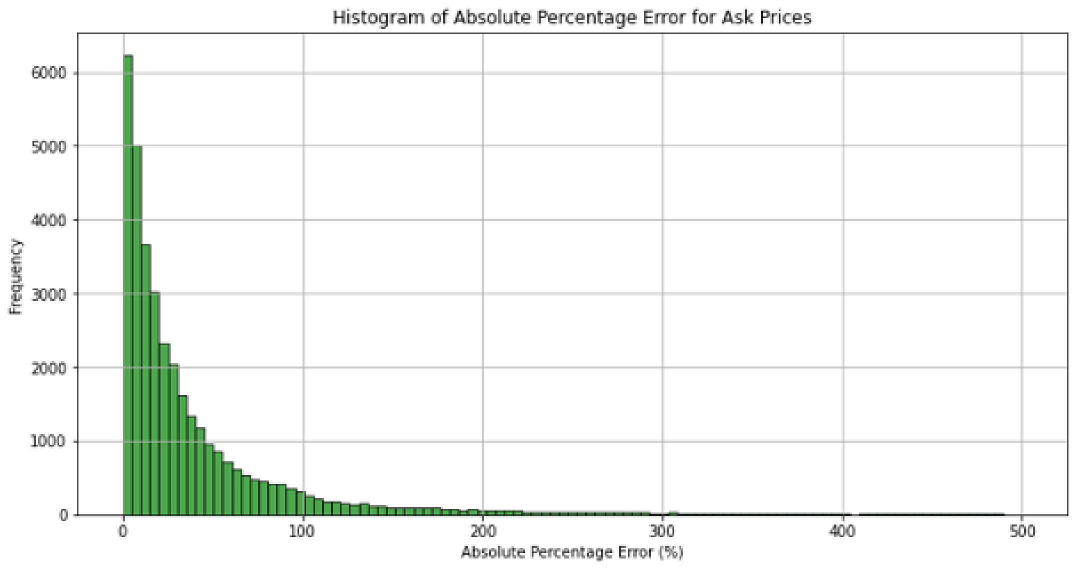


Figure 27. Absolute percent error distribution of the predicted bid and ask prices of the GRU model with additional Google and Microsoft options included in the training/testing dataset.

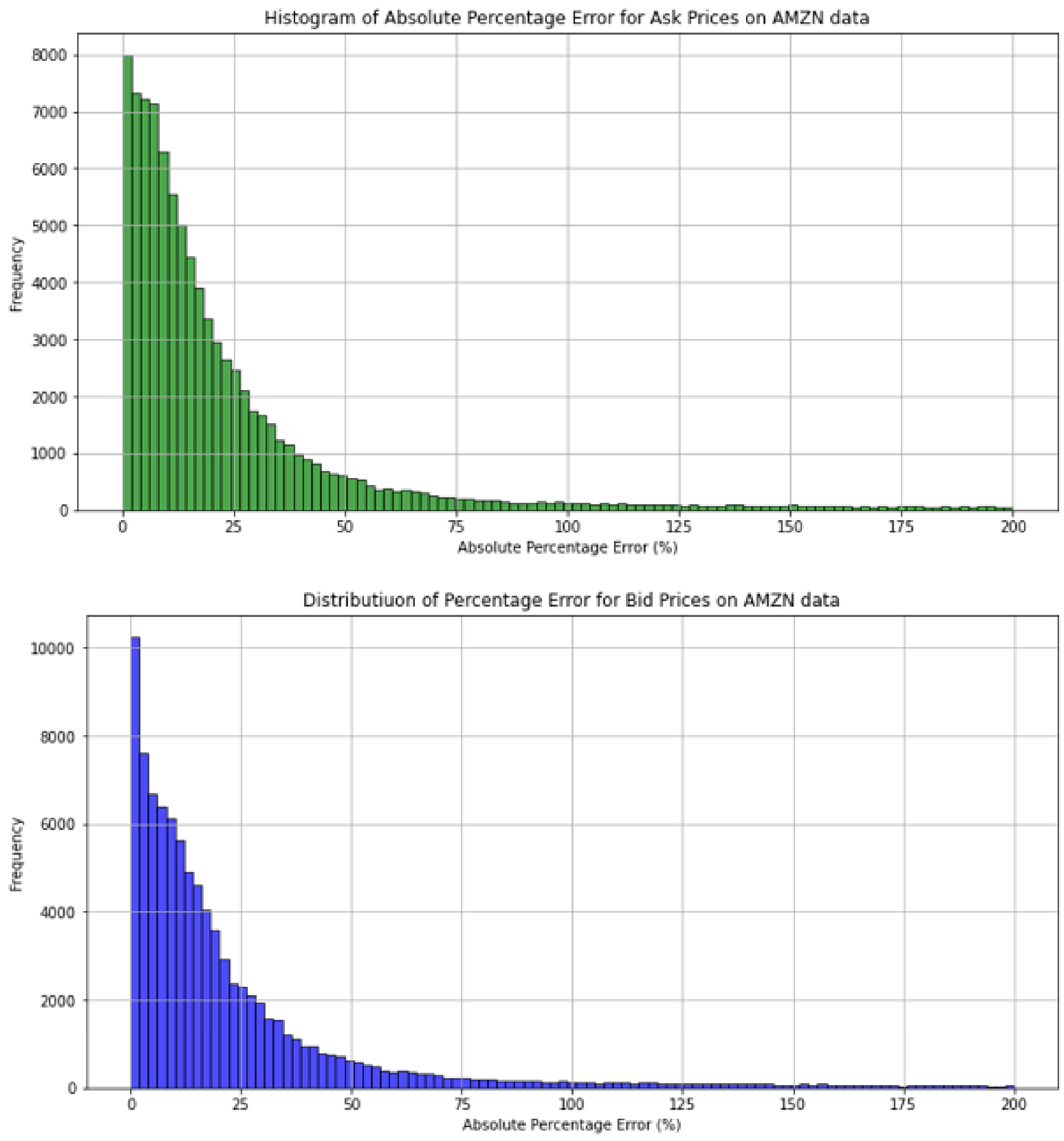


Figure 28. Absolute percent error distribution of the predicted bid and ask prices on Amazon options of the GRU model with additional Google and Microsoft options included in the training/testing dataset.

4893 of 6959 words

Figure 29. Word count of the main body of the report without figure captions.

Machine Learning Terms Glossary

1. **Artificial Neurons:** The basic building blocks of a neural network, mimicking the function of biological neurons. They process input values, weighted accordingly, through mathematical operations, culminating in an output [7].

2. **Recurrent Neural Networks (RNNs)**: A class of neural networks designed for processing sequential data, capable of retaining information from previous inputs through internal memory cycles, thus suitable for time series forecasting [9].

3. **Long Short-Term Memory (LSTM)**: A specialized form of RNN designed to address the vanishing gradient problem and retain information over extended sequences. It includes components like input, forget, and output gates to control the flow of information [10].

4. **Gated Recurrent Unit (GRU)**: Similar to LSTM but with a simplified architecture, GRUs combine the input and forget gates into an update gate, making them computationally efficient and suitable for sequence modeling [13].

5. **Mean Squared Error (MSE)**: A metric for evaluating the accuracy of a model by calculating the average of the squared differences between the predicted and actual values, where a lower MSE indicates more accurate predictions [14].

6. **Mean Absolute Error (MAE)**: A measure used in regression tasks to assess model accuracy, calculated as the average of the absolute differences between predicted and actual observations [14].

7. **Normalization**: The process of adjusting the values of numeric columns in a dataset to a common scale without distorting differences in the ranges of values, essential for effective model training.

8. **Activation Functions**: Functions applied to the output of a neuron to introduce non-linearity, enabling the network to learn complex patterns. Common examples include ReLU, sigmoid, and tanh functions [12].

9. **Adam Optimizer**: An optimization algorithm for training neural networks, known for its effectiveness in handling sparse gradients and adapting the learning rate for each parameter [14].

10. **Dropout Layer**: A technique used to prevent overfitting in neural networks by randomly ignoring a subset of neurons during training, reducing the model's dependency on any single neuron [12] [14].

11. **Backpropagation**: A method used in training neural networks, involving the backward propagation of errors from the output layer to update the weights, optimizing the model's performance [11].
12. **Gradient Descent**: An optimization algorithm used to minimize the loss function by iteratively moving towards the minimum value of the function, adjusting the model's parameters accordingly [11].
13. **Vanishing Gradient Problem**: A challenge in training deep neural networks where gradients used for updating network weights can become exceedingly small, leading to minimal learning in the initial layers of the network [11].
14. **Rectified Linear Activation (ReLU)**: An activation function that outputs zero for negative inputs and retains positive values, introducing non-linearity and aiding in complex pattern recognition within the model [15].
15. **Temporal Data**: Information that is organized in relation to time, essential in time series forecasting where the sequence and timing of data points are crucial for accurate predictions [8][14].

Financial Terms Glossary

1. **Stock Options**: Contracts that grant the holder the right, but not the obligation, to buy or sell a stock at a predetermined price within a specific time period [3].
2. **Call Options**: A type of stock option that allows the holder to buy a specified number of shares at a set price before the contract expires, typically used when anticipating a rise in share price [3].
3. **Put Options**: Options that give the holder the right to sell a specified number of shares at a strike price during the contract's term, generally used when expecting a decline in share price [3].
4. **American Options**: Options that can be exercised at any point during the contract's term, providing greater flexibility compared to European options [3] [16].
5. **European Options**: These options can only be exercised on the expiration date, unlike American options which can be exercised at any time before expiry [4] [3].
6. **Strike Price**: The predetermined price at which an option can be exercised, determining the price at which the underlying asset can be bought or sold [3].

7. **At the Money**: Describes an option where the strike price and the market price of the underlying asset are equal, making the option neither in nor out of the money [3].
8. **Out of the Money**: An option that would not be profitable to exercise, such as a call option with a strike price higher than the market price or a put option with a strike price lower than the market price [3].
9. **In the Money**: Refers to an option that would yield a profit if exercised, for example, a call option with a strike price lower than the market price or a put option with a strike price higher than the market price [3].
10. **Option Contracts**: Formal agreements in the options market that provide the holder the right, but not the obligation, to buy or sell an underlying asset at a specified strike price [3].
11. **Arbitrage Opportunities**: Financial strategies that exploit price differences of the same asset in different markets to make risk-free profits, often sought after by traders using sophisticated models to predict price movements [5].
12. **Leverage**: The use of borrowed funds or financial instruments to increase the potential return of an investment, often employed in options trading where a small investment controls a larger amount of the underlying asset [3].
13. **Market Efficiency**: The concept that asset prices fully reflect all available information, making it impossible to consistently achieve higher returns without taking on more risk. Market efficiency challenges the predictability of stock and option prices [5].
14. **Black-Scholes Model**: A mathematical model used for pricing European options, deriving an analytical solution for the prices of options based on factors like the underlying asset's price, the option's strike price, time to expiration, risk-free interest rate, and volatility [4].

References

[1] M. Scorsese, Director, *The Wolf of Wall Street.* [Film]. United States of America: Paramount Pictures, 2013.

[2] C. Isidore, "Machines are driving Wall Street's wild ride, not humans," CNN, 6 February 2018. [Online]. Available: https://money.cnn.com/2018/02/06/investing/wall-street-computers-program-trading/index.html. [Accessed 11 March 2024].

[3] J. Chen, "What are Options? Types, Spreads, Example, and Risk Metrics," Investopedia, 24 April 2023. [Online]. Available: https://www.investopedia.com/terms/o/option.asp. [Accessed 13 October 2023].

[4] A. Hayes, "Black-Scholes Model: What It Is, How It Works, Options Formula," Adam Hayes, 5 May 2023. [Online]. Available: https://www.investopedia.com/terms/b/blackscholes.asp. [Accessed 12 October 2023].

[5] L. Downey, "Efficient Market Hypothesis (EMH): Definition and Critique," Investopedia, 24 April 2023. [Online]. Available: https://www.investopedia.com/terms/e/efficientmarkethypothesis.asp. [Accessed 18 November 2023].

[6] D. Kalita, "A Brief Overview of Recurrent Neural Networks (RNN)," Analytics Vidhya, 3 August 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2022/03/a-brief-overview-of-recurrent-neural-networks-rnn/. [Accessed 13 October 2023].

[7] S. Cristina, "Calculus in Action: Neural Networks," Machine Learning Mastery, 16 March 2022. [Online]. Available: https://machinelearningmastery.com/calculus-in-action-neural-networks/. [Accessed 22 November 2023].

[8] V. Madisson, "Stock market predictions with RNN using daily market variables," Towards Data Science, 11 June 2019. [Online]. Available: https://towardsdatascience.com/stock-market-predictions-with-rnn-using-daily-market-variables-6f928c867fd2. [Accessed 11 October 2023].

[9] M. Saeed, "An Introduction to Recurrent Neural Networks and the Math That Powers Them," Machine Learning Mastery, 6 January 2023. [Online]. Available: https://machinelearningmastery.com/an-introduction-to-recurrent-neural-networks-and-the-math-that-powers-them/. [Accessed 13 October 2023].

[10] M. Phi, "Illustrated Guide to LSTM’s and GRU’s: A step by step explanation," Medium, 24 September 2018. [Online]. Available: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21.

[11] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation,* vol. 9, no. 8, pp. 1735-1780, 1997.

[12] I. Goodfellow, Y. Bengio and A. Courville, "1.2.1 The Many Names and Changing Fortunes of Neural Networks," in *Deep Learning*, Boston, MIT Press, 2016, p. 18.

[13] S. Kostadinov, "Understanding GRU Networks," Towards Data Science, 16 December 2017. [Online]. Available: https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be. [Accessed 1 March 2024].

[14] I. Jahan, "Stock Price Prediction Using Recurrent Neural Networks," North Dakota State University, Fargo, 2018.

[15] B. Krishnamurthy, "An Introduction to the ReLU Activation Function," BuiltIn, 26 Febuary 2024. [Online]. Available: https://builtin.com/machine-learning/relu-activation-function. [Accessed 20 March 2024].

[16] "IEEE Citation Style Resources," Queen's University LIbraries, [Online]. Available: https://guides.library.queensu.ca/c.php?g=501793&p=3436604. [Accessed 17 February 2020].

[17] Queen's School of Graduate Studies, "Thesis formatting and other resources," [Online]. Available: https://www.queensu.ca/sgs/current-students/degree-completion/thesis-formatting-other-resources. [Accessed 17 February 2020].

[18] M. Lesuisse, "What are the advantages of pricing American options using artificial neural networks?," Université de Liège, Liège, 2022.