Developing Machine Learning Algorithms to Price American-Style Stock Options

ENPH 455 Winter Term Progress Report – Engineering Physics Thesis

Student Author: Nathan Pacey

Supervising Professor: Ryan Martin

Queen's University

Kingston, Ontario Canada

February 23, 2024

Introduction and Project Motivation

This project explores forecasting American-style stock option prices using many-to-many Recurrent Neural Networks (RNNs), particularly focusing on "at the money" options with short-term expirations in large-cap North American equities [1] [2]. By leveraging TensorFlow for iterative validation and parameter adjustments, the project aims to accurately predict bid and ask prices, utilizing different model architectures. Specifically comparing the performance of Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM) models on time series data.

Winter Term Progress

Initial GRU Model and Early Results

After developing and analyzing the LSTM model, a similar Gated Recurrent Unit (GRU) RNN model was created for direct comparison. Both models feature two gated layers with 50 units and a 0.1 recurrent dropout rate, leading to a dense and output layer with an Adam optimizer with a learning rate of 0.001, as shown in Figures 1 and 2 (Appendix A).

Despite the inherent variability typical of machine learning models, comparisons between the LSTM and the initial GRU model's performance can be made since each model was trained and tested on Apple and AMD contracts over identical time frames. To accurately assess the models' predictive accuracy under market-like conditions, the last 20% of the dataset was used for testing, with the ask and bid predictions from both the LSTM and GRU models illustrated in Figures 3 and 4.

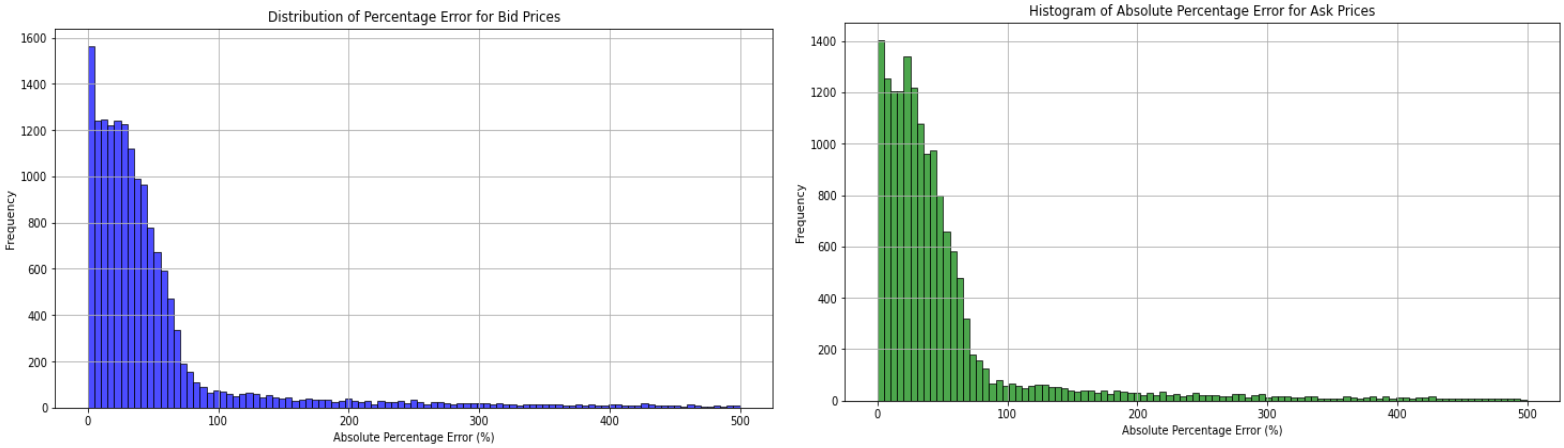

Figure 1. LSTM model performance of ask and bid predictions on realistic data split.

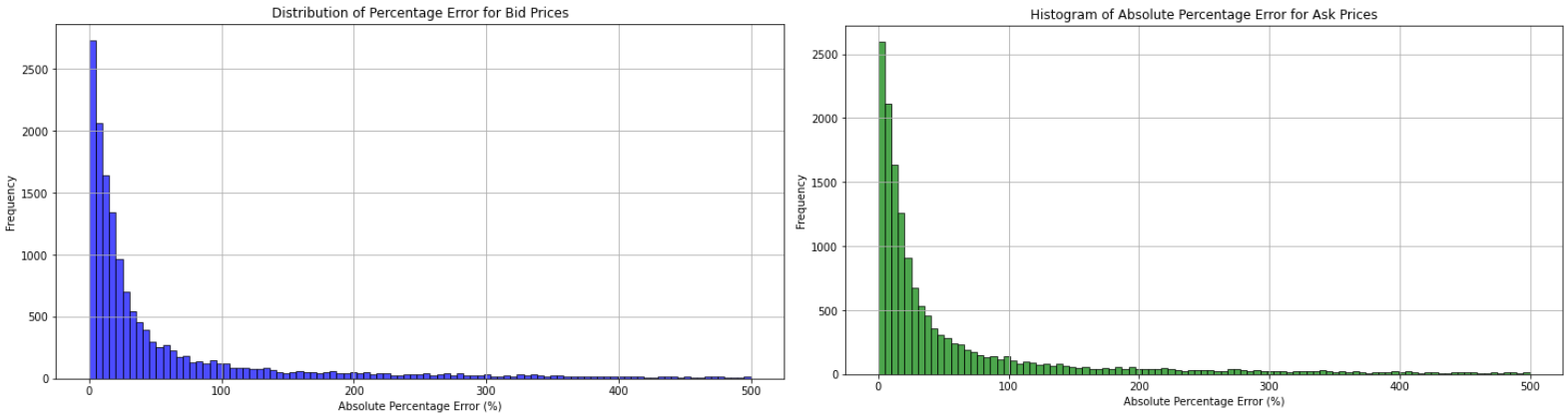

Figure 2. GRU model bid and ask error performance on realistic data split.

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 50.88 and MAE of 5.20, and lower average bid and ask errors of 6.12% and 15.37%, 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, with results as detailed in Figures 5 and 6.

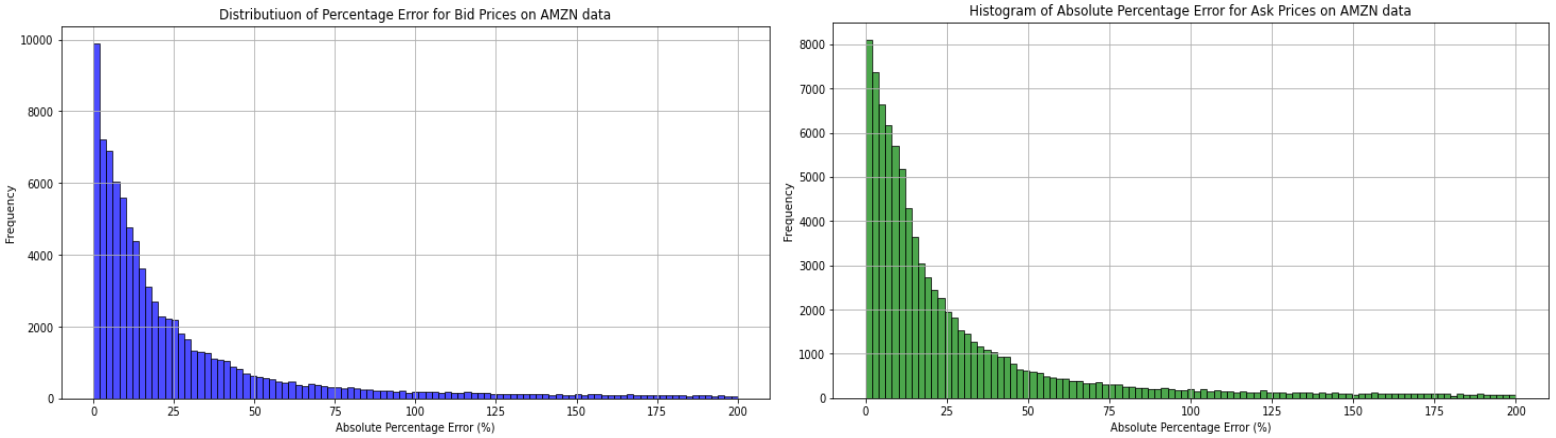

Figure 3. LSTM bid and ask error frequency over Amazon data.

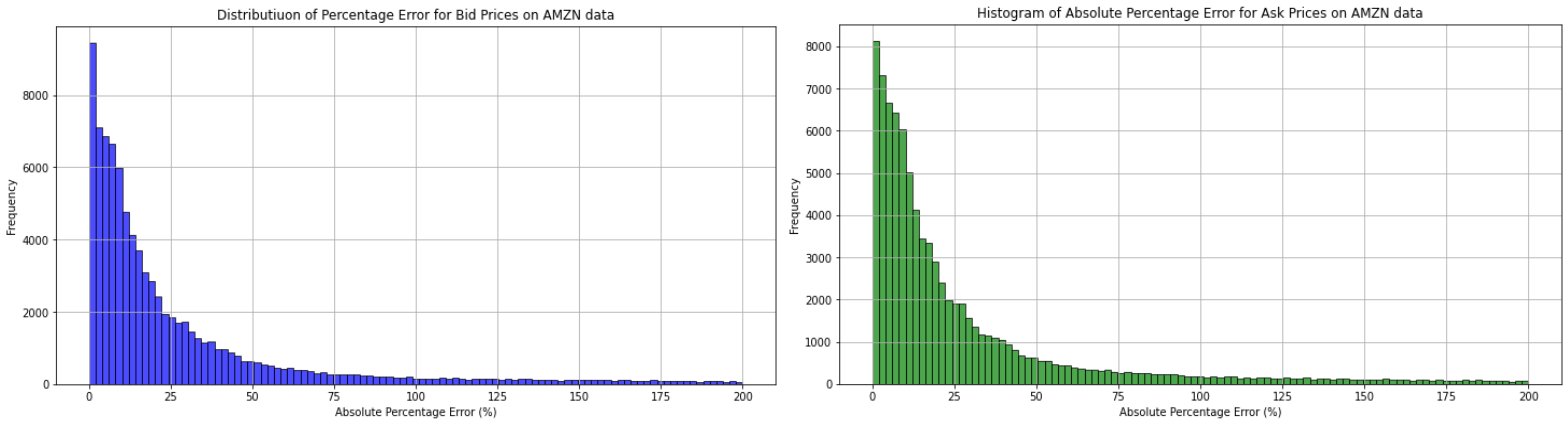

Figure 4. GRU bid and ask error frequency over Amazon data.

The LSTM model's 14% and 13% reduction in bid and ask errors compared to the GRU model on the Amazon dataset is surprising given previous results, but it aligns with the expected variability inherent in machine learning trials. While the LSTM appears superior, the observed difference may not necessarily indicate a definitive advantage but could be attributed to normal prediction variability.

Hybrid GRU Fully Connected Model Progress

To refine the GRU model, a hybrid approach was designed to differentiate temporal from non-temporal elements, aiming to reduce errors by tailoring processing to the specific nature of each data type. This architecture is illustrated in Figure 7.

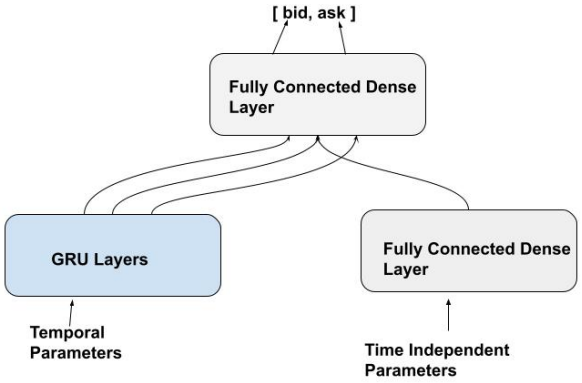

Figure 5. Hybrid GRU and fully connected model illustration.

The general structure and implementation of this model in Python can be seen below in Figure 8 (Appendix A). In the hybrid GRU model, time-independent parameters like option data and contract names are processed through a fully connected layer, whereas time-sensitive stock data are handled by a GRU layer with 32 units and tanh activation. Layer normalization is applied to mitigate errors from data exceeding the training set's range, addressing the potential for stocks and options to reach new highs and lows. The model integrates these layers to output bid and ask prices. Ongoing adjustments are being made to address encoding issues with contract names for effective dataset testing.

Future Progress

In the coming weeks, the focus will be on code optimization for quicker testing, transitioning from Jupyter notebooks to Python scripts for GPU efficiency, and configuring local libraries. Efforts will also concentrate on resolving issues with the Hybrid GRU model, expanding the dataset with more tickers and input parameters, and refining the dataset to exclude distortions like stock splits. Additionally, exploring hyperparameters and normalization methods will be key to minimizing prediction errors. A detailed schedule is available in Appendix B.

References

[1] 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].

[2] 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].

[3] 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].

[4] 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].

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

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

[7] Tensorflow, "Time series forecasting," Tensorflow, 27 July 2023. [Online]. Available: https://www.tensorflow.org/tutorials/structured_data/time_series. [Accessed 9 October 2023].

[8] Quant Next, "Artificial Neural Network for Option Pricing with Python Code," Youtube, July 2023. [Online]. Available: https://www.youtube.com/watch?v=r6KTnKim_BA&ab_channel=QuantNext. [Accessed 3 October 2023].

Appendix A. Project Code

```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 6. LSTM model constructed in Python using TensorFlow.

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

```python
def build_gru_model_2(string_lookup_layer, option_data_shape, stock_data_shape):

    # Assuming `x_contract_names` is your list or array of contract names.
    string_lookup = StringLookup(output_mode="multi_hot")
    # Adapt with actual contract names data
    string_lookup.adapt(x_contract_names)

    # Define inputs for each type of data
    contract_name_input = Input(shape=(None,), dtype='int32', name='encoded_contract_name')
    inputOptionData = Input(shape=option_data_shape, name='option_data')
    inputStockData = Input(shape=stock_data_shape, name='stock_data')

    # Contract name processing
    x0 = string_lookup(contract_name_input)
    x0 = Dense(16, activation="relu")(x0)
    x0_embedding_output = Dense(4, activation="relu", name='embedding_output')(x0)

    # Option data processing with normalization
    x1_norm = LayerNormalization()(inputOptionData)
    x1 = layers.Flatten()(x1_norm)
    x1 = Dense(32, activation="relu")(x1)
    x1_option_dense_output = Dense(8, activation="relu", name='option_dense_output')(x1)

    # Stock data processing with GRU and normalization
    x2_norm = LayerNormalization()(inputStockData)
    x2 = GRU(32, return_sequences=True, activation='tanh')(x2_norm)
    x2 = Dropout(0.2)(x2)
    x2 = layers.Flatten()(x2)
    x2_stock_gru_output = Dense(8, activation="relu", name='stock_gru_output')(x2)

    # Combine processed inputs
    combined = Concatenate(name='concat_output')([x0_embedding_output, x1_option_dense_output, x2_stock_gru_output])
    combined = Dense(16, activation="relu")(combined)
    outputs = Dense(2, activation="linear")(combined)

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

    return model
```

Figure 8. Hybrid GRU and folly connected layer implemented in Python utilizing TensorFlow.

Appendix B. Revised Project Plan

Week 6: Code Optimization and Environment Setup

- Transition Jupyter notebooks to Python scripts for GPU efficiency.
- Establish a local environment mirroring the Jupyter configuration.
- Initiate code cleanup for improved testing speed.

Week 7: Model Refinement and Dataset Expansion

- Address and rectify issues within the Hybrid GRU model.
- Augment the dataset with more tickers for comprehensive training and testing.

Week 8: Dataset Cleaning and Hyperparameter Adjustment

- Refine the dataset, removing anomalies such as stock splits.
- Begin experimenting with hyperparameters and normalization methods to ensure critical market data is preserved.

Week 9: Advanced Model Testing and Documentation Start

- Conduct in-depth testing, validation, and tuning of the model.
- Start comprehensive documentation, detailing the project's approach, data processing, and preliminary insights.

Week 10: Documentation Enhancement and Presentation Drafting

- Continue refining documentation with a focus on methodologies, findings, and analysis.
- Draft the initial version of the final presentation, highlighting significant outcomes and model efficacy.

Week 11: Final Revisions and Presentation Finalization

- Complete any remaining revisions to the documentation, ensuring clarity and completeness.
- Finalize the presentation, preparing to showcase the project's achievements, insights, and potential impacts.

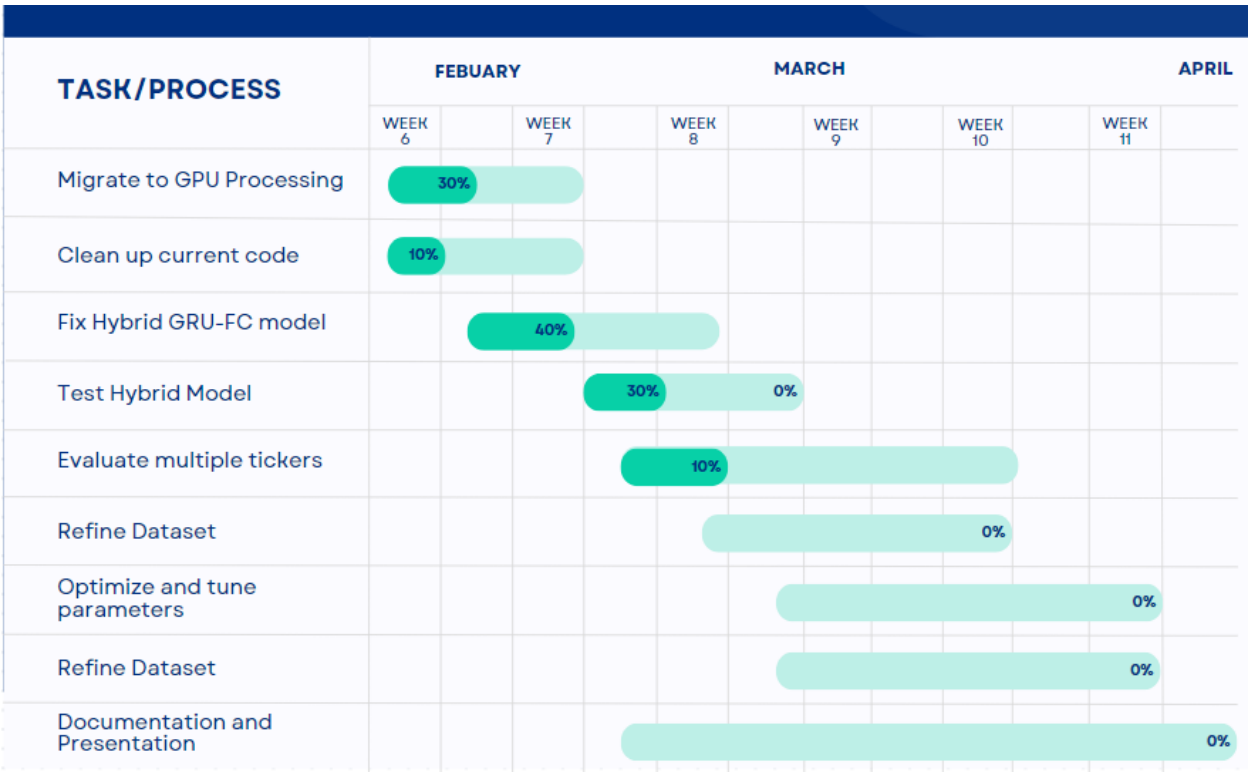

Figure 9. Gannt Chart of the updated project plan including approximate task progress.

Appendix C. Word Count of Report Body

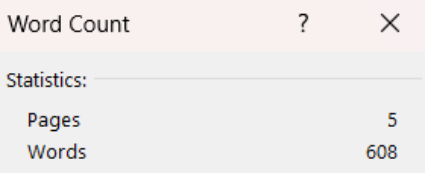

Figure 10. Word Count of the main body of the report including figure captions.