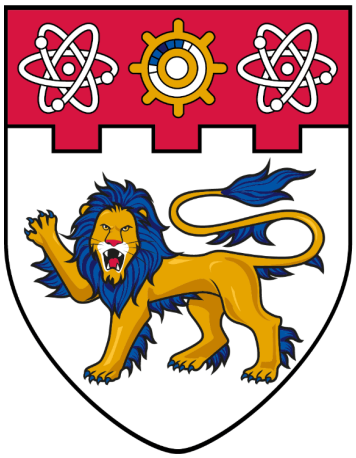


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CZ4042 Neural Network and Deep Learning

Group Project

Members:
Ananya Balehithlu (U2021741A)
Ananya Unnikrishnan (U2021032D)
Roshan Thapa (U2022199F)

1 Introduction

In the dynamic landscape of financial markets, predicting options prices accurately is essential for effective investments and risk strategies [1]. American options are particularly important due to their flexibility in exercise terms which makes their valuation more complex in comparison to European options [2]. Conventional models struggle to account for the non-linear relationships that exist in these financial instruments, even if they offer fundamental insights [3]. This research paper aims to address this gap through the exploration of neural networks.

Neural networks have transformed data-driven analysis in several domains, including image identification and natural language processing [4]. They are especially promising for the task at hand in finance because of their capacity to identify tiny patterns and learn from enormous amounts of data [5]. The purpose of this study is to examine several neural network models, such as Feedforward Neural Networks (FNNs), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) and Modular Neural Networks (MNNs) in order to assess how well they predict American call option prices in comparison to traditional models like the Barone-Adesi and Whaley (B-AW) approximation method currently used in the industry.

2 Literature Review

The Barone-Adesi and Whaley (B-AW) approximation method offers a quasi-analytical solution for American options. The B-AW model specifically accounts for their early exercise premium, a feature absent in the Black-Scholes (B-S) model that is suited only for European options [6]. Despite improvements, the B-AW method's assumptions, such as constant dividend yields and volatility, may not always align with market realities, prompting a shift towards more adaptive techniques like neural networks [7].

Neural networks excel in modeling complex, non-linear interactions and learning from data, making them ideal for the dynamic task of American option pricing. They have the potential to absorb the multifaceted influences on American options, such as the early exercise decision, without explicit modeling. This potential was first demonstrated in the seminal work of Hutchinson, Lo, and Poggio (1994), which showed neural networks could learn to price derivatives from data [8]. Further studies, for instance by Amilon (2003), have built on this foundation, demonstrating neural networks' superiority over traditional models in the context of American options.

Moreover, neural networks offer computational advantages; they can deliver instantaneous pricing post-training, beneficial in time-sensitive trading scenarios, in contrast to the more computationally demanding B-AW method. The evolution of explainable AI is addressing the opacity of neural networks, as research by Dixon, Halperin, and Bilokon (2020) introduces interpretative approaches to these models, making them more transparent and justifiable in finance [9].

Current research is pushing boundaries by integrating neural networks with real-time market data and other qualitative inputs to refine American option pricing even further. The work of Chen, Pelger, and Zhuo (2019) on high-dimensional data integration is a testament to neural networks' capacity to enhance option pricing models [10].

This paper will build on the existing body of work, offering new empirical insights on neural networks in the pricing of American call options and exploring practical considerations for their real-world application.

3 Methodology

3.1 Neural Networks and Keras

Neural networks, as the name suggests, are a subset of machine learning algorithms modeled loosely after the human brain. By learning from data, they can be trained to recognize intricate patterns. In the context of this paper, we will be utilizing Keras, an open-source library for the implementation of our neural networks [11].

Keras serves as an interface for the TensorFlow library and simplifies the entire process of training neural network models. As such, it is ideal for the networks we want to implement, and can run seamlessly on both CPUs and GPUs [12].

The library provides a straightforward way to define and train neural network models in Python. Researchers can easily stack layers to build multi-layer perceptrons, convolutional neural networks, and more complex architectures, with a wide range of pre-defined layer types and support for custom layers [13].

For our purposes, Keras offers a range of functionalities that are particularly relevant to financial computing and option pricing such as facilitating the creation of neural networks with its easy-to-use layer construction, making it ideal for the complex task of option pricing. It also includes a range of optimization algorithms such as Adam and SGD for efficient model training, along with loss functions like mean squared error for regression tasks such as price predictions. To ensure the models generalize well, Keras also offers regularization techniques like dropout and L1/L2 methods to prevent overfitting. Additionally, it provides various metrics to evaluate the performance of neural networks which will allow us to assess the accuracy of option pricing models [14].

3.2 Data Pre-processing

This study utilizes a comprehensive dataset consisting of the end-of-day options pricing for prominent stocks, Apple Inc. (AAPL), NVIDIA Corporation (NVDA), and Tesla Inc. (TSLA), from the time periods of January 2023 to September 2023 [15] [16] [17]. These stocks were chosen strategically due to their high liquidity as American options with significant daily trading volumes as well as high market capitalization [18].

Initial data attributes in the dataset consist of the quote date, expiration date, days to expiration, Greek metrics, the underlying asset's price, implied volatility, trading volume, last traded price, bid price, ask price, option strike price, strike distance and its percentage. To enrich this dataset, dividend rate was added from Yahoo Finance and the interest rate from the Federal Reserve Economic Data (FRED) was matched with various U.S. Treasury maturities to the option's days to expiration to substitute for the risk free interest rate. Several technical indicators were also computed for each of the stocks such as Simple Moving Average (SMA), Exponential Moving Average (EMA), Relative Strength Index (RSI), and Bollinger Bands. These indicators are essential in capturing the underlying asset's momentum, trend, and volatility [19], which are significant factors in the valuation of American options.

The data was cleaned by removing rows with NaN values and further refined by excluding rows where implied volatility and days to expiration of the call option is zero.

The final key features selected for the neural network models include the underlying asset's price, implied volatility, risk free interest rate, days to expiration and dividend rate which are all important features required in the Barone-Adesi and Whaley Approximation method [20]. This dataset was supplemented with the trading volume, strike price metrics and the previous call option price and implied volatility to provide a richer feature set.

Additionally, Greek metrics i.e. Delta, Gamma, Vega, Theta and Rho were also included as these provide sensitivities to various factors that affect options pricing such as stock price movement, time decay, volatility, etc., which are crucial for capturing the nuances of American options pricing [21]. And finally the Simple Moving Average (SMA), Exponential Moving Average (EMA), Relative Strength Index (RSI), and Bollinger Bands were added as these technical indicators are pivotal in understanding the historical price movement and volatility, which may influence American call options.

The final dataset was subsetted by each stock option. All subsetted datasets were further divided into training, validation and test sets in a 70-10-20 ratio.

3.3 Feedforward Neural Networks

Feedforward neural networks (FNN) are the simplest type of artificial neural network. They consist of three main layers: an input layer, one or more hidden layers, and an output layer. Neurons make up each layer, with weights connecting each neuron in one layer to the subsequent layer's neurons through weighted connections. An activation function is used to calculate each neuron's output, which introduces nonlinearity into the learning process [22].

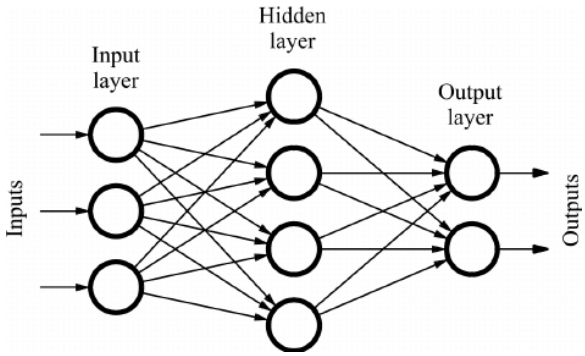


Figure 1: FNN Architecture

FNNs are particularly suited for predicting American options due to their structured approach in modeling complex relationships [23].

Before feeding the data to our FNN model, we scaled our data using the MinMaxScalar to allow the model to converge faster since all the features are on a similar scale [24].

The adam optimizer was chosen due to its consistent performance especially in cases involving noisy gradients and sparse data [25]. The learning rate was set to the default 0.001 and a weight decay regularization of 1e-6 was added to reduce overfitting [26].

To identify the best parameters for our FNN model, the comparison of the performance of a model with one hidden layer of 10 neurons, another with one hidden layer of 128 neurons and a third with two hidden layers of 128 neurons each will be done. This comparison aims to assess how well models of varying complexity are able to discern and replicate the underlying trends within American option data. The performance between models with the activation function of ReLU (Rectified Linear Unit) and Leaky ReLU as well as a batch size of 32 and 64 will also be observed.

And finally to train the FNN model, 100 epochs with early stopping (patience 10) will be used which will stop the training process once the validation loss ceases to decrease, thus preventing overfitting of the model [27].

3.4 Recurrent Neural Networks

Neural networks that are specifically built to handle sequential input are known as recurrent neural networks (RNNs). RNNs process sequences by utilizing their internal state, or memory.

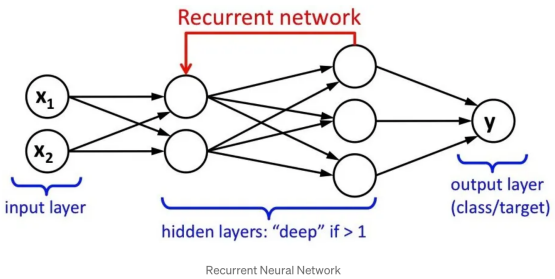


Figure 2: RNN Architecture

Specifically, we made use of Long Short-Term Memory (LSTMs), a unique type of RNN that Hochreiter & Schmidhuber (1997) introduced and which is capable of learning long-term dependencies. They are made to avoid the problem of vanishing gradients, in contrast to traditional RNNs. LSTMs include four interacting layers with a topology resembling a chain, in contrast to typical RNNs, which only have one layer. They can use cell state, forget gates, input, and output to capture long-term dependencies in data [28].

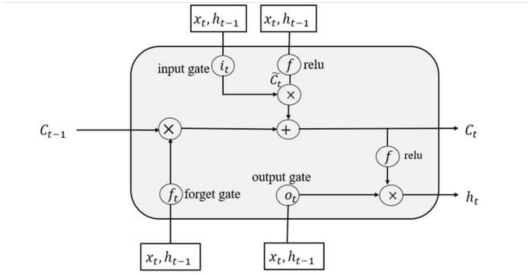


Figure 3: LSTM Architecture

There are also two other RNNs that are LSTM derivatives. One kind is the Gated Recurrent Unit (GRU), an LSTM variant that was suggested by Cho et al. (2014) and intended to be more computationally straightforward and efficient. The cell state and hidden state are combined, and the input and forget gates are combined into a single update gate. Bidirectional LSTM models are a different kind that were first presented by Schuster & Paliwal (1997). In this sort of network, the input sequence is fed both forward and backward, giving the network access to context from the past and future at any point in the sequence.

LSTMs and its derivatives have been used in multiple research papers in the field of option pricing, especially for European options. For instance, Liang, L., & Cai, X. (2022) discovered that LSTM and 1D-CNN outperformed other models in forecasting accuracy and robustness due to their ability to capture time-series information [29] while Liu, Y., & Zhang, X. (2023) found that LSTM with realized skewness did the best in all metrics among the traditional and other machine learning methods. [30] Thus, we believe it will be worth experimenting the use of LSTM in American option pricing.

Before feeding the data into the neural network, MinMaxScaler was used to scale the data, ensuring that all features contribute equally to the model's learning process. The data was also reshaped to match the input requirements of LSTM layers (samples, time steps, features).

Each model was configured with a single 50-unit layer, followed by a dense layer to output the predicted output price. Adam optimiser was utilized, with mean squared error loss function. The training phase lasted for 100 epochs with a batch size of 32. Early stopping (patience of 10) was utilized to avoid overfitting.

3.5 Convolution Neural Networks

Convolutional neural networks (CNN) are a type of neural network generally used for object recognition and classification. In order to generate a predicted output, they first convert an input into a feature map, which is then processed through a number of layers. Three primary parts make up a CNN architecture: the input layer, hidden layers,, and the output layer [31].

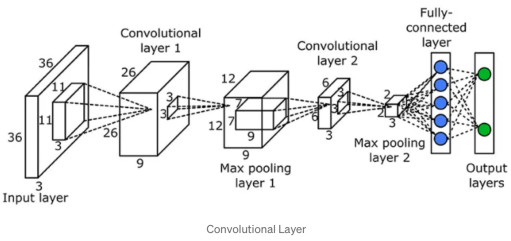


Figure 4: CNN Architecture

Due to CNN's ability to capture identifying patterns and features, especially non-linear relationships, in complex datasets, we believe it will be fruitful to explore applying it to the dynamic field of American option pricing, where large high-dimensional data is commonly found.

StandardScaler was used to fit the training data and transform the other datasets. The data was also reshaped to add an additional time dimension for the The model was initialized with a 1D convolutional layer (Conv1D) with 64 filters and a kernel size of 3, using ‘relu’ activation. Next, a max pooling layer (MaxPooling1D) was added to reduce the dimensionality of the data. The data was then flattened to a single dimension with Flatten. Lastly, a dense layer with 50 neurons and ‘relu’ activation is added, followed by an output dense layer with 1 neuron to suit the regression problem.

Adam optimiser was utilized, with mean squared error loss function. The training phase lasted for 100 epochs with a batch size of 32. Early stopping (patience of 10) was utilized to avoid overfitting.

3.6 Modular Neural Networks

Modular neural networks (MNN) are a type of artificial neural network consisting of several separate networks that each function independently and combine to generate the final result. Each sub-network or module can all be trained independently as they are in charge of a particular task. The architecture's method of processing complicated issues is to divide them into smaller, easier-to-manage subproblems [32].

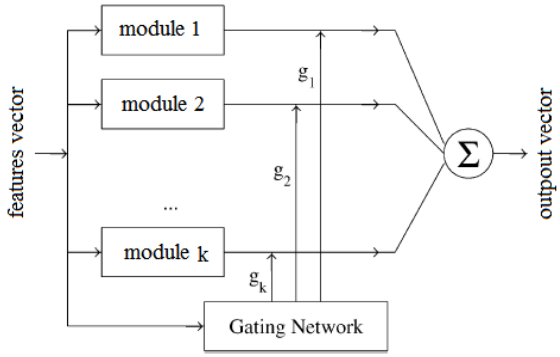


Figure 5: MNN Architecture

Since MNNs have the structural advantage in handling specialization and complexity, they are especially well-suited for predicting American options [33].

For this model, each of the option stock datasets was split into in-the-money, at-the-money, and out-of-the-money categories. In-the-Money options have a strike price below the market price for call options making them profitable to exercise. At-the-Money options have a strike price close to the market price. Out-of-the-Money options have a strike price above the market price for call options making them not profitable to exercise [34]. By subsetting the options data into these three categories, the MNN will be able to tailor its sub-networks to highlight the unique traits and actions of each group. For example, in-the-money options are generally more impacted by implied volatility compared to out-of-the-money options [35].

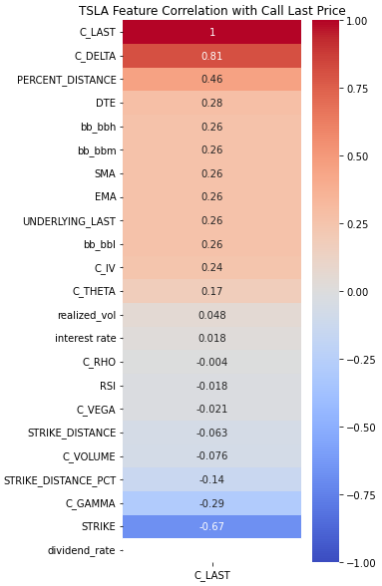
To create our sub modules for each of the three subsets, a feedforward neural network was used since they excel in mapping complex non linear relationships which is essential for capturing the intricate patterns of different moneyness categories in american options. Additionally, their simple structure allows for the training and testing of different combinations of neurons, hidden layers and activation functions that will best capture the complex patterns in each of the moneyness categories.

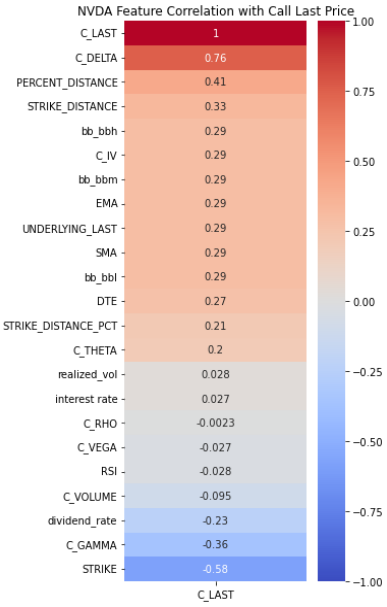
MinMaxScalar was chosen to scale the data after which for the training process, the adam optimizer with a learning rate of the default 0.001 was used. Furthermore, a gradient norm clipping of 1.0 was used to mitigate possible exploding gradient problems [36]. This ensures that the optimizer moves steadily in the direction of the loss function's minima without making abrupt changes that could sabotage the learning process.

And finally to train the combined MNN model, a batch size of 32 and trained for 50 epochs with early stopping (patience 10).

4 Experiments and Results

4.1 Data Exploration





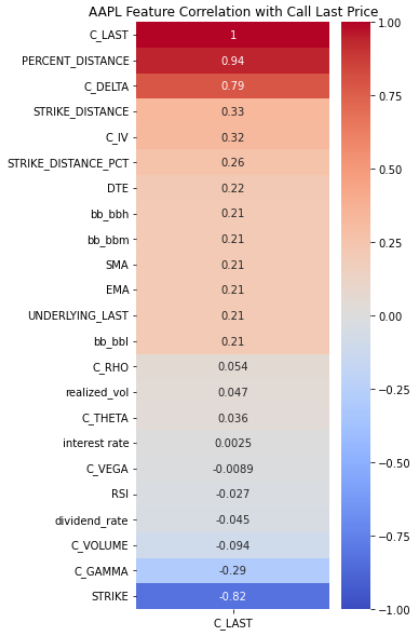
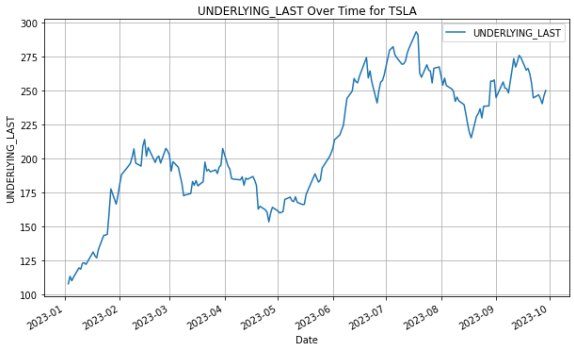
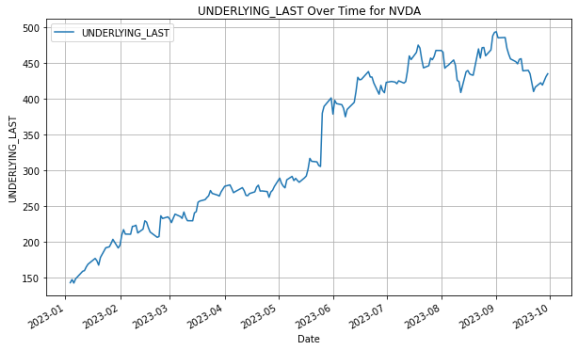
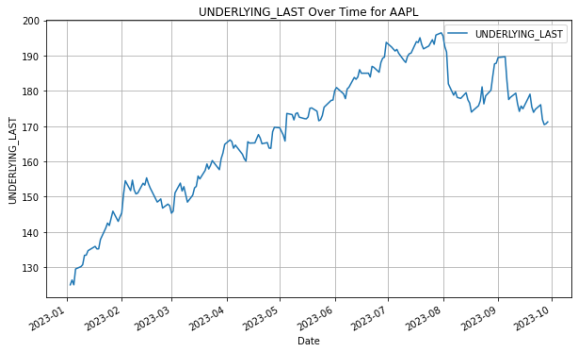


Figure 6, 7, 8: Correlation Matrix for TSLA, NVDA, AAPL

Visualizing the data revealed that while *C_DELTA* had the highest correlation score of 0.81 and 0.76 with Call Last Price (*C_LAST*) for TSLA and NVDA stocks respectively, *PERCENT_DISTANCE* scored significantly higher with a score of 0.94 with Call Last Price for AAPL stock. In comparison, *PERCENT_DISTANCE* had a low correlation with *C_LAST* for TSLA and NVDA (0.46 and 0.41 respectively)







Figures 9, 10, 11: UNDERLYING_LAST over Time for TSLA, NVDA, AAPL

Over the duration of the dataset (January to September), the UNDERLYING_LAST generally increased. TSLA stock experienced a greater range of ups and downs, NVDA climbed relatively steadily, and AAPL experienced a sudden dip at the start of August.

4.2 Feedforward Neural Networks

In our endeavor to optimize our Feedforward neural network model, hyperparameter tuning was done on the TSLA call options data using 5-fold cross validation. This methodical tuning focused on comparing three model architectures: one with a single hidden layer of 10 neurons, another with one layer of 128 neurons and a third configuration with two hidden layers of 128 neurons each.

In addition to the model architecture, our investigation extended to the comparison of two activation functions: ReLU (Rectified Linear Unit) and Leaky ReLU. The impact of the batch sizes, 32 and 64, was also observed as previously mentioned in Section 3.3 in this paper. The main criteria for selecting the optimal model was the minimization of validation Root Mean Squared Error (RMSE) during the cross validation process.

The most effective model, as indicated by the lowest validation RMSE, had 2 layers with 128 neurons each with both having an activation function of Leaky ReLU. For the optimization of this model, the Adam optimizer with a learning rate of 0.001 and a weight decay of 1e-6 was used. The Mean Squared Error Loss function was chosen due to its effectiveness in regression tasks [37].

Subsequently, the finalized model was trained with a batch size of 32.

Stock	RMSE
TSLA	12.716268
AAPL	4.0349474
NVDA	23.618427

Table 1: Root Mean Squared Error (RMSE) for each of the stock option data

4.3 Recurrent Neural Networks

In our endeavor to determine the most effective architecture among various recurrent network types, we commenced by training and evaluating three foundational models—Long Short-Term Memory (LSTM), Bidirectional LSTM (BiLSTM), and Gated Recurrent Unit (GRU)—utilizing standardized parameters on the Tesla (TSLA) stock dataset. Our initial findings indicated that GRU achieved the lowest Root Mean Square Error (RMSE), outperforming LSTM, with BiLSTM trailing behind.

Building upon these insights, we focused on enhancing the two superior models, GRU and LSTM, by experimenting with additional layers. Four distinct configurations were explored: a two-layer stacked LSTM, a two-layer stacked

GRU and an LSTM+GRU hybrid. Of these, the two-layer stacked LSTM model emerged as the most effective, exhibiting the lowest RMSE.

Further experimentation involved augmenting the model with an additional layer of a LSTM layer. However, this modification resulted in a decline in performance, suggestive of overfitting. Consequently, it was determined that the two-layer stacked LSTM model configuration offered the optimal balance of complexity and efficiency, yielding the best performance for the task at hand. After hyperparameter tuning across the different configurations of 50 and 100 neurons across both LSTM layers and batch sizes of 32 and 64, we found that the optimal batch size is 32 and the optimal configuration is 100 neurons for both layers

Stock	RMSE
TSLA	13.21041
AAPL	4.54117
NVDA	25.39253

Table 2: RMSE for each of the stock option data

4.4 Convolutional Neural Networks

In our quest to identify the most effective Convolutional Neural Network (CNN) architecture, our initial approach involved the training and evaluation of a single-layer CNN model. To address the need for recognizing more intricate patterns within the dataset, we progressively enhanced the model's complexity. This enhancement entailed the incorporation of additional convolutional layers equipped with a greater number of filters, alongside the integration of dropout layers and dense layers. Despite these modifications, the model's performance did not exhibit notable improvements.

Subsequently, we explored a hybrid architecture combining a single CNN layer with a two-layer Stacked Long Short-Term Memory (LSTM) network. Surprisingly, this configuration underperformed in comparison to the standalone two-layer stacked LSTM model. We concluded that further exploration of CNNs may not be a judicious allocation of resources. This decision was based on the preliminary assessment that CNNs, in this context, might not be optimally suited without substantial investment in understanding, computational resources, and development efforts. For our experiments, CNN + 2-layer Stacked LSTM was the best performing model among other CNN models.

Model	TSLA RMSE
Simplex CNN	13.79835
2-convolutional layer Complex CNN	15.27365
3-convolutional layer Complex CNN	14.19028
CNN + 2-layer Stacked LSTM	13.48803

Table 3: RMSE of the various CNN models

4.5 Modular Neural Networks

To optimize our Modular neural network model, hyperparameter tuning was done to find out the best model architecture for each of the three sub modules created according to the different moneyness categories: in-the-money, at-the-money and out-of-the-money.

For each of the sub module architectures, the performance of different number of layers and neurons: one layer of 64 neurons, one layer of 128 neurons, two layers of 64 neurons, two layers of 128 neurons, three layers of 64, 128 and 64 neurons respectively and three layers of 128 neurons each were scrutinized. Additionally, the performance of two activation functions: ReLU (Rectified Linear Unit) and Exponential Linear Unit (ELU) was also investigated. The main criteria for model selection is the minimization of validation loss.

The most effective model for each of the three submodules, as indicated by the lowest

validation loss were: 3 layers with 128 neurons each and an activation function of ELU for in-the-money sub module, 3 layers of 64, 128 and 64 neurons respectively with an activation function of ReLU for at-the-money submodule and 1 layer of 128 neurons with an activation function of ReLU for out-of-the-money submodule.

Stock	RMSE
TSLA	12.693567
AAPL	3.5411253
NVDA	24.175006

Table 4: RMSE of MNN for each of the stock option data

5 Discussion

The purpose of this study was to explore the efficacy of neural networks in the realm of American option pricing which had been traditionally dominated by mathematical models like the Barone-Adesi and Whaley (B-AW) approximation. Our findings present a better understanding on how neural networks can either complement or replace the current conventional methods used in the industry

5.1 Comparison with the Barone Adesi and Whaley (B-AW) Method

The Barone Adesi and Whaley (B-AW) Method has long been known for its relative simplicity in predicting the price of American options [20]. However, our neural network models show a significant difference in how it handles the inherent complexity of American option pricing. Our neural network models have the unique benefit of learning directly from data, in contrast to the B-AW model, which is based on a set of fixed mathematical equations. This data-driven approach allows the model to capture the non-linear dependencies and complex market dynamics that traditional models can miss.

Furthermore, the incorporation of additional features like the Greek metrics and various technical indicators as detailed in Section 3.2 enriches the input data for our neural network models. These features, while not traditionally employed in the B-AW method, offer a more comprehensive dataset allowing our neural network models to approximate the prices of American options more accurately.

This increased accuracy can be demonstrated in Figure x, which shows the Root Mean Squared Error (RMSE) for each of the three stock option data across different models. A critical observation from this analysis is that all the neural network models used in this study, consistently exhibited lower RMSE values compared to the B-AW model. This result not only confirms how well neural networks can capture intricate market patterns that conventional models would miss.

Model	TSLA RMSE	AAPL RMSE	NVDA RMSE
B-AW	21.12745	6.941821	44.93740
FNN	12.71627	4.034947	23.61843
2-layer Stacked LSTM	13.21041	4.54117	25.39253
MNN	12.69357	3.541125	24.17501

Table 5: Root Mean Squared Error (RMSE) for each of the stock option data

5.2 Final Model Analysis

From Table 5, it can be observed that the Modular Neural Network (MNN) model has the lowest RMSE and thus performed the best for the AAPL and TSLA dataset. This can be attributed to the architecture of MNN that allows for the creation of submodules which are tailored to different moneyness categories. Moneyness, being a critical concept in options trading allowed the MNN to more accurately capture the intricacies and patterns in price movement of

such highly volatile options like the AAPL and TSLA options.

In comparison, NVDA's market behavior might exhibit more predictability and be less influenced by complex factors like moneyness which must have resulted in the simple Feedforward Neural Network (FNN) model to better capture its price movements.

The 2-layer Stacked Long Short-Term Memory (LSTM) model might not have performed as well as the FNN and MNN model due to the options pricing data lacking strong temporal dependencies which could have been a result of these stocks being highly volatile and dependent on current market movement compared to their historical price.

5.3 Limitations

When discussing the limitation of our neural network models in comparison to the traditional methods like the Barone-Adesi and Whaley (B-AW) method, it is important to acknowledge certain aspects. The neural network models generally require a large amount of data to train and capture the complex patterns that lie in American option data. Moreover, this data needs to be of high quality to ensure accurate results. The lack of interpretability in neural networks also makes it challenging to interpret how they arrived at those pricing decisions. In comparison, the traditional models like B-AW, while less flexible in handling complex market behavior, are more transparent making their prediction more interpretable which is essential in the industry. However, the introduction of explainable AI (XAI) techniques as detailed in Section 2 significantly addresses this opacity issue. This will enable financial experts to gain a better understanding and trust in the decision making process of these models, thus paving the way for their more informed application in the financial sector.

6 References

[1] Dejanovski, A. (2014). The Role and Importance of the Options as a Unstandardized Financial Derivatives. TEM Journal, 3(1), 81-87. https://www.temjournal.com/documents/vol3no1/TemJournalFebruary2014_81_87.pdf

[2] Ekström, E. (2004). Properties of American option prices. *Stochastic Processes and Their Applications*, *114*(2), 265-278. https://doi.org/10.1016/j.spa.2004.05.002

[3] Tudor, A. (2022). Comparison between traditional and modern option pricing models. https://essay.utwente.nl/91977/1/Tudor_BA_BIT.pdf

[4] Alishahi, A., Chrupała, G., & Linzen, T. (2019). Analyzing and interpreting neural networks for NLP: A report on the first BlackboxNLP workshop. Natural Language Engineering, 25(4), 543-557. https://doi.org/10.1017/S135132491900024X

[5] Dase, R., & Pawar, D. (2010). Application of artificial neural network for stock market predictions: A review of literature. International Journal of Machine Intelligence, 2(2), 14-17. https://doi.org/10.9735/0975-2927.2.2.14-17

[6] BARONE-ADESI, G. and WHALEY, R.E. (1987), Efficient Analytic Approximation of American Option Values. The Journal of Finance, 42: 301-320. https://doi.org/10.1111/j.1540-6261.1987.tb02569.x

[7] Just, D. R., Khantachavana, S. V., & Just, R. E. (2010). Empirical challenges for risk preferences and production. Annual Review of Resource Economics, 2(1), 13-31. https://doi.org/10.1146/annurev.resource.012809.103902

[8] Hutchinson, J.M., LO, A.W. and POGGIO, T. (1994), A Nonparametric Approach to Pricing and Hedging Derivative Securities Via Learning Networks. The Journal of Finance, 49: 851-889. https://doi.org/10.1111/j.1540-6261.1994.tb00081.x

[9] Dixon, M. F., Halperin , I., & Bilokon, P. (2020). Machine Learning in Finance: From Theory to Practice (1st ed.). Springer Cham. https://doi.org/10.1007/978-3-030-41068-1

[10] Chen, L., Pelger, M., & Zhu, J. (2018). *Deep Learning in Asset Pricing* [Stanford University]. https://doi.org/10.2139/ssrn.3350138

[11] Chollet, F. (2017). Deep Learning with Python (2nd ed.). Manning Publications.

[12] Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (2nd ed.). O'Reilly Media, Inc.

[13] Goodfellow, I., Bengio , Y., & Courville, A. (2016). Deep Learning. The MIT Press.

[14] Brownlee, J. (2016). Deep Learning With Python (1st ed., pp. 19-22). Machine Learning Mastery.

[15] EODData. (n.d.). NASDAQ, AAPL End of Day and Historical Stock Data [Apple Inc] https://eoddata.com/stockquote/NASDAQ/AAPL.htm

[16] EODData. (n.d.). NASDAQ, NVDA End of Day and Historical Stock Data [Nividia Corp] https://eoddata.com/stockquote/NASDAQ/NVDA.htm

[17] EODData. (n.d.). NASDAQ, TSLA End of Day and Historical Stock Data [Tesla Inc] https://eoddata.com/stockquote/NASDAQ/TSLA.htm

[18] Daly, L. (2023, November 3). The Largest Companies by Market Cap in 2023. The Motley Fool. https://www.fool.com/author/20014/

[19] Seth, S. (2022, August 10). The Top Technical Indicators for Options Trading. Investopedia. https://www.investopedia.com/articles/active-trading/101314/top-technical-indicators-options-trading.asp#toc-bollinger-bands

[20] Fatone, L., Mariani, F., Recchioni, M. C., & Zirilli, F. (2015). The Barone-Adesi Whaley Formula to Price American Options Revisited. Applied Mathematics, 06(02), 382-402. https://www.doi:10.4236/am.2015.62036

[21] Umeorah, N., Mashele, P., Agbaeze, O., & Mba, J. C. (2023). Barrier Options and Greeks: Modeling with Neural Networks. Axioms, 12(4), 384. https://doi.org/10.3390/axioms12040384

[22] Laudani, A., Lozito, G. M., Fulginei, F. R., & Salvini, A. (2015). On Training Efficiency and Computational Costs of a Feed Forward Neural Network: A Review. Computational Intelligence and Neuroscience. https://doi.org/10.1155/2015/818243

[23] Eskiizmirliler, S., Günel, K., & Polat , R. (2021). On the Solution of the Black–Scholes Equation Using Feed-Forward Neural Networks. Computational Economics, 58, 915–941. https://doi.org/10.1007/s10614-020-10070-w

[24] Paczkowski, W. R. (2020). Business Analytics: Data Science for Business Problems (pp. 127-157). Springer Cham. https://doi.org/10.1007/978-3-030-87023-2_5

[25] Mahendra, S. (2023, June 13). What is the Adam Optimizer and How is It Used in Machine Learning. Artificial Intelligence +. https://www.aiplusinfo.com/blog/what-is-the-adam-optimizer-and-how-is-it-used-in-machine-learning/

[26] Zhang, G., Wang, C., Xu, B., & Grosse, R. (2018). Three Mechanisms of Weight Decay Regularization. https://doi.org/10.48550/arXiv.1810.12281

[27] Caruana, R., Lawrence, S., & Giles, L. (2000). Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early

Stopping. Advances in Neural Information Processing Systems, 13, 402-408. https://proceedings.neurips.cc/paper_files/paper/2000/file/059fdcd96baeb75112f09fa1dcc740cc-Paper.pdf

[28] Chatterjee, C. C. (2021, December 11). Implementation of RNN, LSTM, and GRU - towards data science. Medium. https://towardsdatascience.com/implementation-of-rnn-lstm-and-gru-a4250bf6c090

[29] Liu, Y., & Zhang, X. (2023). Option Pricing Using LSTM: A Perspective of Realized Skewness. Mathematics, 11(2), 314. https://www.mdpi.com/2227-7390/11/2/314

[30] Liang, L., & Cai, X. (2022). Time-sequencing European options and pricing with deep learning – Analyzing based on interpretable ALE method. Expert Systems With Applications, 187, 115951. https://doi.org/10.1016/j.eswa.2021.115951

[31] Shahriar, N. (2023, February 1). What is Convolutional Neural Network — CNN (Deep Learning). Medium. https://nafizshahriar.medium.com/what-is-convolutional-neural-network-cnn-deep-learning-b3921bdd82d5

[32] Auda, G., & Kamel, M. (1999). Modular Neural Networks: A Survey. International Journal of Neural Systems, 9(2), 129-151. https://doi.org/10.1142/s0129065799000125

[33] Gradojevic, N., Gencay, R., & Kukolj, D. (2009). Option Pricing With Modular Neural Networks. IEEE Transactions on Neural Networks, 20(4), 626-637. https://doi.org/10.1109/TNN.2008.2011130

[34] (n.d.). In-The-Money, At-The-Money or Out-of-The-Money Calls? Desjardins Online Brokerage. https://www.disnat.com/en/learning/trading-basics/desjardins-online-brokerage/in-the-money-at-the-money-or-out-of-the-money-calls

[35] Hamida, S. B., Abdelmalek, W., & Abid, F. (2014). Applying Dynamic Training-Subset Selection Methods Using Genetic Programming for Forecasting Implied Volatility. Computational Intelligence, 32, 369–390. https://doi.org/10.1111/coin.12057

[36] Zhang, J., He, T., Sra, S., & Jadbabaie, A. (2019). Why gradient clipping accelerates training: A theoretical justification for adaptivity [Conference Paper at ICLR 2020, Massachusetts Institute of Technology]. https://doi.org/10.48550/arXiv.1905.11881

[37] Jadon, A., Patil, A., & Jadon, S. (2022). A Comprehensive Survey of Regression Based Loss Functions for Time Series Forecasting. https://doi.org/10.48550/arXiv.2211.02989