BU Questrom School of Business

Option Price Prediction Based on Neural Network Models

MF796 Final Report

April, 2024

Contents

1 Introduction

Option pricing has been a long-standing topic in financial research. Under the assumptions of geometric Brownian motion based on stock prices and the no-arbitrage principle, the Black-Scholes formula has become the most critical and well-recognized pricing model. However, the intricacies of the market in which options are traded mean that these assumptions do not always hold, which leads to pricing biases. Pricing of options through machine-learning modeling algorithms can be achieved without regard to economic principles and beliefs. Using financial data variables as inputs, the algorithm can directly output the price of an option by learning from a large amount of data. Deep learning is a branch of machine learning based on a more advanced form of artificial neural networks. It is a much more powerful technique for learning complex features and patterns of data through a multi-layered neural network structure, often used to process large-scale and high-dimensional data.

In this project, we empirically analyze option pricing using three different neural network models, LSTM, CNN+RNN, and Conv-LSTM, on data from the Chinese financial market, starting from tensor-based data to capture the correlation between different information in the options market. Finally, we build vertical spread and vanilla strategies based on the prediction results to trade all options daily without asset and transaction cost constraints.

2 Related Work

Financial data are characterized by time series. Recurrent neural networks and long and short-term memories are essential models for processing sequence data. A large number of research teams have studied the field of option pricing based on machine learning models with RNN structure.

As early as 1993, Malliaris and Salchenberger proposed a neural network that uses data on the underlying price, strike price, implied volatility, risk-free rate, and past lags to predict option prices [1]. This model outperforms the Black-Scholes model in both in-the-money and out-of-the-money options. Twenty years later, Berger implemented an option pricing model based on Google Cloud's AutoML Regressor and two different types of feed-forward neural networks [2]. The model removes implied volatility and adds the last 20 lags as a single feature variable to predict the midpoint price of an option, ultimately achieving an order of magnitude better optimization of the mean absolute error. However, neural network-based option price prediction suffers from the defect of quickly triggering over-fitting. To ameliorate this shortcoming, Gradojevic, and his team propose a non-parametric fuzzy neural network model, an algorithm that "specializes" in pricing specific options by partitioning the search space [3]. The pricing estimated using the modularized features of the MNN model outperforms the Black-Scholes model and the NN model with hints.

3 Methodology

3.1 Data Description

Our dataset consists of fundamental data and historical data from the Chinese financial markets. To implement the B-S model, we chose parameters including spot price, strike price, expiration date, call or put (option), and implied volatility. To equalize the dimensions for data from different sources, we added inventory variables for four price variables: previous settle price, settle price change, theory price, and theory margin. Considering that option prices are susceptible to external factors, we introduced Delta, Gamma, Theta, Vega, and Rho. These Greeks help us to measure the sensitivity of prices to parameters. We collected 50ETF option and stock data from April 2018 to June 2020 from the Yahoo Finance website. From the original 34 variables, 829 observations with at least 20 days of validity were screened. We end up with a dataset containing 33,210 training data observations and 11,386 test data observations through proper format conversion.

3.1.1 Feature Representation

To feed the data into the neural network, we need to preprocess it into a format that the network can read, which is a tensor.

In Figure 3.1, the large box on the left represents the raw data. We can see that the data contains information for multiple options (Option 1, Option 2, ..., Option m). There are a total of 15 features used for prediction, which we divide into three groups, each containing five features. These groups are denoted as group1, group2, and group3 in the figure.

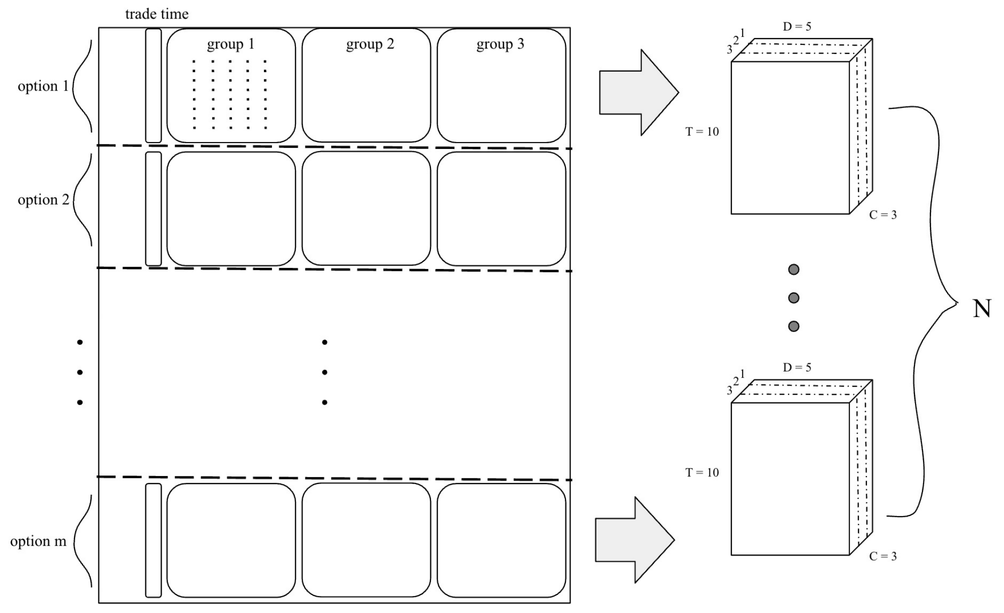


Figure 3.1: the Schematic Diagram of Data Processing

Each option has its own unique ID. For each option, we generate subsequences of length 10 using a sliding window approach. For each subsequence, we organize it into a tensor with a shape of (C, T, D), represented by the cuboids in the figure, where C is the channel, T is the time length, and D is the feature dimension. Here, C = 3, T = 10, and D = 5. We apply this processing to the data of each option, generating N such tensors, with each tensor corresponding to an independent input sample. Ultimately, these data will be used in the neural network model to predict future option prices.

For the entire input dataset, the data shape is (T, C, N, D), where N is the number of tensors. The corresponding target data shape is (N, 1).

3.2 Methodology

3.2.1 CNN+RNN

The CNN+RNN model is widely used in spatio-temporal prediction problems[4]. Figure 3.2 shows the structure of the CNN+RNN model. Looking at the upper part of the model, the data is transposed and reshaped into (N, T, C×D) and then passed through three layers of GRU, outputting data with dimensions (N, T, 16).

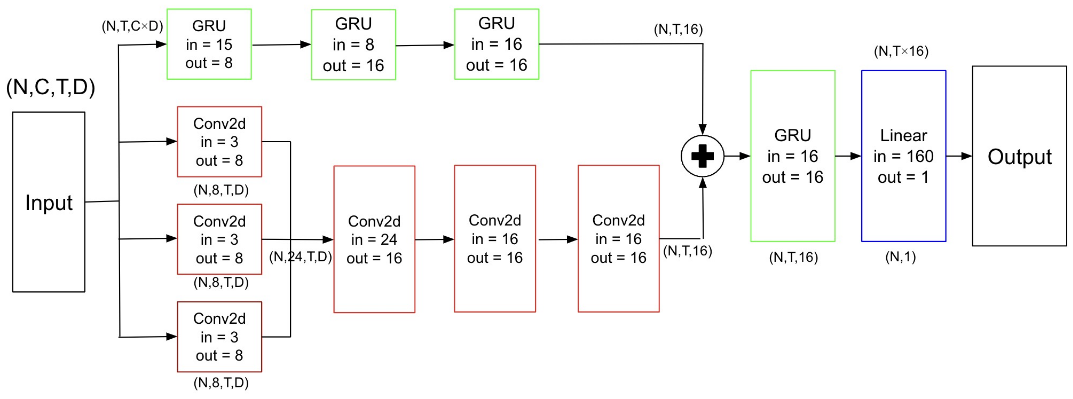


Figure 3.2: CNN+RNN Model Structure

Going back to the input data, the data is simultaneously fed into three convolutional layers without any processing. The input and output dimensions of these layers are the same, but the convolutional kernel settings are different, which is used to learn different aspects of the data. The 1D convolution here is along the timeline (T) and feature (D). Each convolutional layer outputs data with dimensions (N, 8, T, D), which are then concatenated, resulting in a dimension of (N, 24, T, D). The data then enters three convolutional layers, outputting data with dimensions (N, 16, T, D). Although not shown in the figure, after the linear layer and squeeze, the dimension becomes (N, 16, T). After transposing, we have data with dimensions (N, T, 16), which is the same as the output of the GRU. The two outputs are then added as the input for the final calculation. At this point, the data passes through a layer of GRU and a linear layer, yielding an output with dimensions (N, 1).

3.2.2 LSTM and ConvLSTM

We also experimented with the LSTM model. Given the input data with the same dimensions and structure as the original data (N, C, T, D), we first transpose and reshape it. Then, we feed the reshaped data into three layers of LSTM, where the hidden size is set to 8. We use bidirectional LSTM to capture both forward and backward dependencies in the sequence. The output from the LSTM layers has a dimension of (N, T, 16). Finally, we apply a reshaping operation followed by a linear layer to obtain the final output.

The Conv-LSTM model is derived from the LSTM model and is specifically designed to tackle spatio-temporal sequence prediction problems [5]. While the LSTM model uses fully connected layers to compute the input gate, forget gate, output gate, and candidate cell state, the Conv-LSTM model employs convolutional operations for these computations. This allows Conv-LSTM to better capture local features and spatial dependencies in the data.

In our implementation, we also set the number of Conv-LSTM layers to three, keeping it consistent with the LSTM model for a fair comparison. By using convolutional operations, Conv-LSTM can effectively learn and exploit the spatial structure present in the option dataset.

4 Results

4.1 Model Performance

4.1.1 Computational Efficiency

By training the model, we found that the neural network structure based on Conv-LSTM can be accelerated well on GPU, and the running time of the same parameter settings is more than 8 hours on CPU, but the training can be completed in only 3.5 minutes on GPU. There are hardware units and algorithms optimized for convolution operations, meaning that convolution and matrix operations in Conv-LSTM can be carried out simultaneously. In addition, the GPU data parallelism and the qualities of forward and backward propagation effectively increase the model's arithmetic power and significantly improve the training efficiency. Comparing the performance of CNN-RNN and LSTM on CPU and GPU, the performance difference between them could be more explicit. This outcome is due to the smaller model size, which results in more minor arithmetic. Transferring data from CPU memory to GPU memory and transferring results from GPU memory back to CPU memory offsets the speedup effect, resulting in GPU optimization for RNN being less mature and efficient than for CNN.

4.1.2 Prediction Outcomes

The experiments showed that the B-S model-based computational training efficiency is the fastest. However, it has relatively high MSE and RMSE (MSE = 0.00082, RMSE = 0.02872), which implies that its prediction has a more significant average error. The CNN-RNN model has the highest prediction accuracy with the lowest MSE and RMSE (MSE = 0.00017, RMSE = 0.01320). The LSTM and ConvLSTM have relatively close performance, but both outperform the Black-Scholes model. The MAPE shows that the CNN-RNN performs the worst in percentage error (MAPE = 2.39909), which may be because a small number of prediction errors significantly affect the overall percentage error. The Conv-LSTM has the lowest MAPE(0.88298), indicating the slightest percentage error in predicting the price versus the actual cost. Additionally, all models correlate more than 0.99, indicating a high match between the expected and actual price trends.CNN-RNN shows the highest correlation (0.99689), confirming its predictive power.

The neural network model shows higher accuracy than the traditional B-S model, especially the excellent performance of CNN-RNN in most performance metrics. Considering the high value of MAPE, the CNN-RNN model may have significant prediction errors in some extreme cases, which requires further optimization. The neural network model requires longer training time and higher computational resources but provides more accurate predictions. The specific model parameter settings and performance are shown in Table 4.1.

We compared the forecast results of option prices with actual price data from the Chinese stock market. As shown in Figure 4.1, the top half of the chart is the result of the training set, and the bottom half is the result of the test set. For the data in the training set, every 100 data points are sampled, and for the data in the test set, every 50 data points are sampled.

Table 4.1: Comparison of Different Models

Model	BS	CNN_RNN	LSTM	convlstm
max_epoch	-	100	100	200
batch_size	-	1024	1024	2048
learning_rate	-	0.01	0.001	0.001
Chip	Apple M1	Apple M1	Apple M1	L20(48GB)GPU
Training Time	2s	7m17s	1m16s	3m46s
MSE	0.00082	0.00017	0.00022	0.00039
RMSE	0.02872	0.01320	0.01497	0.01983
MAP	0.02123	0.01026	0.01124	0.01188
MAPE	0.47957	2.39909	1.15552	0.88298
Correlation	0.99472	0.99689	0.99285	0.99257

The predictions of all three models show tight coverage, which implies a high degree of overlap with the actual prices. There are prediction biases in some peaks, especially with sharp price swings, which suggests that the models may need further optimization when dealing with extreme variation or noise. The models' predictions perform differently at peaks, with CNN-RNN and LSTM predicting closer to the valid price, while Conv-LSTM performs slightly worse at extremes.

4.2 Trading Strategy

4.2.1 Vertical Spread Strategy

Utilizing the predicted results from the CNN-RNN model, this study explores a vertical spread option trading strategy that capitalizes on price variances across call options with shared expiration dates but different strike prices. The approach assumes that all options can be traded daily without limitations on capital or incurring transaction costs.

The strategy encompasses 90 distinct 'spotID' groups of options derived from the same underlying assets (indicated by identical 'tradeDate' and 'spotPrice') and spans 116 trading days. Options within each group are tracked over the 30 days leading up to their expiration, offering a granular view of pricing dynamics. Mispricing is assessed by comparing model-estimated prices with actual market values, identifying overvaluation (positive 'diff') and undervaluation (negative 'diff'). Trading signals within each group direct the selling of the most overpriced options and buying the most under-priced ones based on these discrepancies. Positions are reversed at each discrete time point from T+1 through T+5, during which the strategy's profitability is evaluated against the actual prices observed on these respective days.

Results in mean and cumulative profits analysis (See Appendix I and II) suggest that trade initiated at time T and reversed at T+5 yield optimal returns, with the incremental increase in holding time correlating with enhanced profitability. The trend demonstrates the effectiveness of the vertical spread approach, particularly near option expiration, where heightened volatility and sharper price discrepancies present opportunities for capitalization.

4.2.2 Vanilla Strategy and Comparative Analysis

Under a similar grouping approach, this simplistic strategy executes trades on all mispriced options within each 'spotID' group. Maximized mean and cumulative profits are obtained when positions are squared off at T+5 (See Appendix III and IV).

We observe a marked divergence in outcomes after scrutinizing the mean returns of two distinct trading strategies across five days (See Appendix V). The vertical spread strategy displays a consistent return profile, with slight fluctuation in average profitability, regardless of the trade duration. In stark contrast, the vanilla strategy unfolds a pronounced upward trend in profit gains, intensifying with each successive trading day within the observed window.

The outperformance of the vanilla strategy can largely be ascribed to its expansive market engagement, effectively leveraging a broad spectrum of pricing inefficiencies. This phenomenon becomes increasingly evident over time. In comparison, the vertical spread strategy, which meticulously selects options with the most substantial perceived mispricing, maintains a consistency in returns that does not parallel the increasing profit trajectory observed in the vanilla strategy as the trading interval extends.

5 Conclusion

In this project, we successfully applied three neural network models (CNN+RNN, LSTM, and Conv-LSTM) to capture market and fundamental information, resulting in more accurate option price predictions. By utilizing machine learning models inherently insensitive to financial markets, we avoided subjective decision-making bias and were able to more objectively assess the relationship between markets and assets.

Experimental results showed that the CNN+RNN model performed best on most performance metrics, with the lowest mean squared error (MSE=0.00017) and root mean squared error (RMSE=0.01320), demonstrating the highest prediction accuracy. However, its mean absolute percentage error (MAPE=2.39909) was relatively high, indicating that there may be significant prediction errors in some extreme cases, requiring further optimization. Conv-LSTM had the lowest MAPE (0.88298), suggesting the smallest percentage error in predicting prices compared to actual prices. All models had a correlation greater than 0.99, indicating a high match between predicted and actual price trends. CNN+RNN had the highest correlation (0.99689), further confirming its predictive power.

Based on the prediction results from the CNN+RNN model, we explored a vertical spread option trading strategy. The results indicated that initiating trades at time T and reversing positions at T+5 yielded optimal returns, with the increase in holding time correlating with enhanced profitability. This trend demonstrated the effectiveness of the vertical spread strategy, particularly near option expiration, where heightened volatility and sharper price discrepancies presented more opportunities for profit. In contrast, the vanilla strategy displayed a pronounced upward trend in average returns over five days. This can be attributed to its broad market engagement, effectively leveraging various pricing inefficiencies.

However, the limited size of the dataset and model resulted in a limited degree of model learning. We made predictions using only 15 independent variables, leaving a significant gap in the information that can be covered. The great success of neural networks in image recognition and natural language processing has been due to the emergence of large datasets and models. We believe that with access to more financial data and models of greater size and arithmetic power, neural network-based predictive models can achieve a broader range of applications, more accurate predictions, and greater intelligence.

Overall, the neural network models demonstrated higher prediction accuracy compared to the traditional Black-Scholes model. Despite the limitations of the current study, we have reason to believe that machine learning-based option pricing methods will have broad application prospects as data and models are further optimized.

Bibliography

[1] M. Malliaris and L. Salchenberger, “Beating the best: A neural network challenges the black-scholes formula,” in *Proceedings of 9th IEEE Conference on Artificial Intelligence for Applications*, pp. 445–449, IEEE, 1993.

[2] J. E. Berger, “Pricing european options with google automl, tensorflow, and xgboost,” *arXiv preprint arXiv:2307.00476*, 2023.

[3] N. Gradojevic, R. Gençay, and D. Kukolj, “Option pricing with modular neural networks,” *IEEE transactions on neural networks*, vol. 20, no. 4, pp. 626–637, 2009.

[4] M. Alfarraj and G. AlRegib, “Semi-supervised learning for acoustic impedance inversion,” in *SEG Technical Program Expanded Abstracts 2019*, pp. 2298–2302, Society of Exploration Geophysicists, 2019.

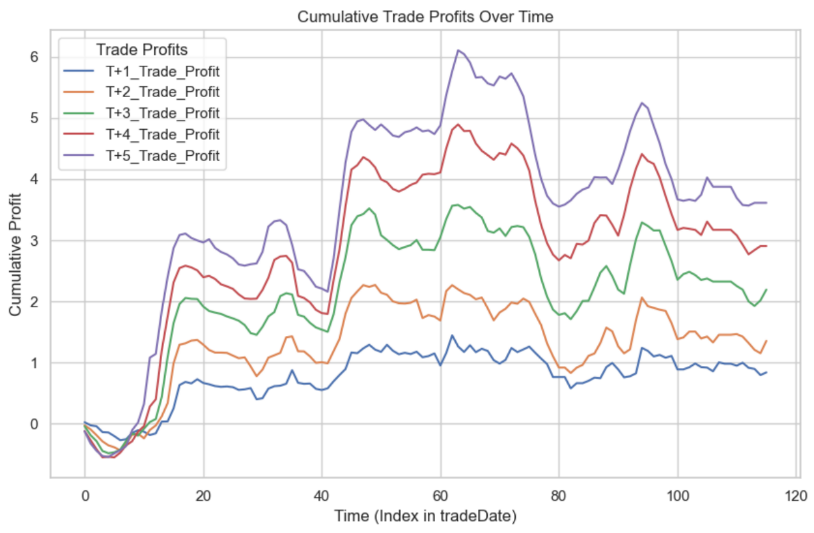
[5] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, “Convolutional lstm network: A machine learning approach for precipitation nowcasting,” *Advances in neural information processing systems*, vol. 28, 2015.

A

Appendix I: Mean profit by reverse transaction day

Reverse Transaction Day	Mean profit
T+1	0.007228
T+2	0.011678
T+3	0.018906
T+4	0.025039
T+5	0.031139

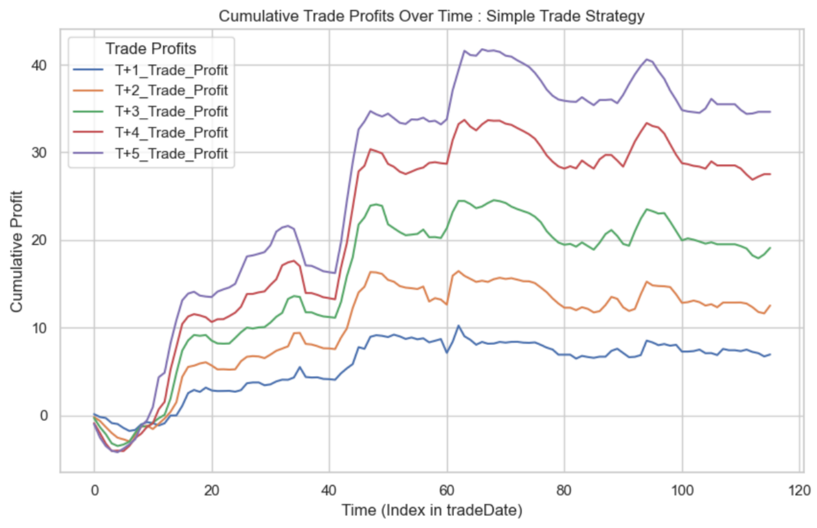
Appendix II: Cumulative trade profits over time



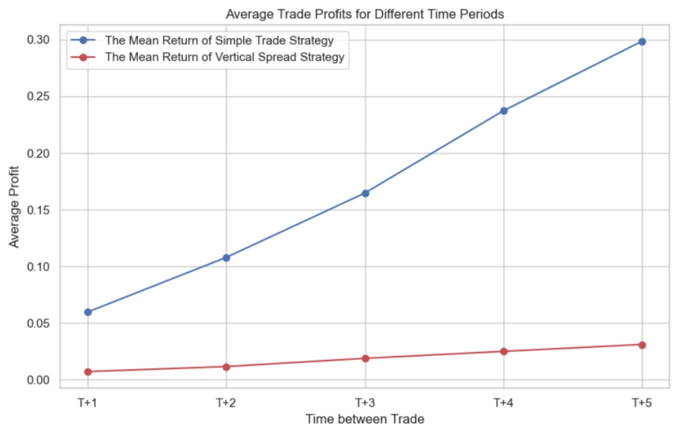
Appendix III: Mean profit by reverse transaction day

Reverse Transaction Day	Mean profit
T+1	0.059796
T+2	0.107938
T+3	0.164659
T+4	0.237204
T+5	0.298472

Appendix IV: Cumulative trade profits over time



Appendix V: Comparative results of average trade profit across time


Average Trade Profits for Different Time Periods
The Mean Return of Simple Trade Strategy
The Mean Return of Vertical Spread Strategy
Average Profit
Time between Trade
T+1
T+2
T+3
T+4
T+5