15.773 HODL Project Report: Deep Learning for Option Pricing

Yash Gupta, Harsh Kumar, Vanessa Quon, Vivian Chinoda

March 16, 2025

1 Problem Description

A call option is a financial instrument that trades like a stock but gives the buyer the right - but not the obligation - to buy a stock at a predetermined price (strike price) before the option expires. Similarly, a put option gives the seller the right to sell at a predetermined price.

In the financial industry, option pricing plays a critical role in determining the value of financial derivatives. Traditional methods, such as the Black-Scholes model, are helpful as they provide a closed-form solution that is efficient and easy to compute, making it a convenient estimate of option values. However, these models have limitations, namely that they make unrealistic assumptions and fail to capture the nonlinear and changing behavior of underlying factors. For example, Black-Scholes assumes constant volatility, but market data has found that options far out-of-the-money are often more volatile than at-the-money options.

Estimating stock or option prices using only the publicly available information is known to be an extremely non-trivial problem in the highly efficient equity markets, often involving a high MSE. Nonetheless, we plan to investigate if Deep Learning techniques can add value in extracting pricing information from publicly available data on stock prices, option contract details, and bond yields, comparing them against traditional analytical models in the industry (such as the OLS) as well as the widely used Black-Scholes-Merton (BSM) formula.

In this project, we will use several ML models to price vanilla American call and put options. Specifically, we test the utility of deep learning models in: (1) approximating data simulated by the BSM formula (the use of neural networks as a universal function approximator) and (2) fitting real-world data (their potential use as a superior alternative to regression in finance).

2 Approach

2.1 Data: Background and Features

The industry standard for pricing options is the Black-Scholes model which takes as input the current stock price, strike price, time to expiration, risk-free interest rate, and the volatility of the stock and outputs the price of the option.

The Black-Scholes model assumes that the underlying stock follows a Geometric Brownian motion specified as

$$\frac{dS}{S} \sim N(\mu dt, \sigma^2 dt)$$

$$\implies ln(S_T) \sim N[ln(S_0) + (\mu - \frac{\sigma^2}{2})T, \sigma^2 T]$$

where S denotes the current stock price, t the time, T the terminal time (such as the expiration date of an option), μ the drift, and σ the volatility. Then a derivative f, which is a function of S and t, follows the Black-Scholes equation, which may be expressed as

$$\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 f}{\partial S^2} = rf$$

where r is the risk-free rate [1].

At expiration (that is, at time T) the vanilla European put and call options impose the following boundary conditions for the following payoff (visualized in Figure 1):

- Call: $C_T = max(S_T - K, 0)$
- Put: $P_T = max(K - S_T, 0)$

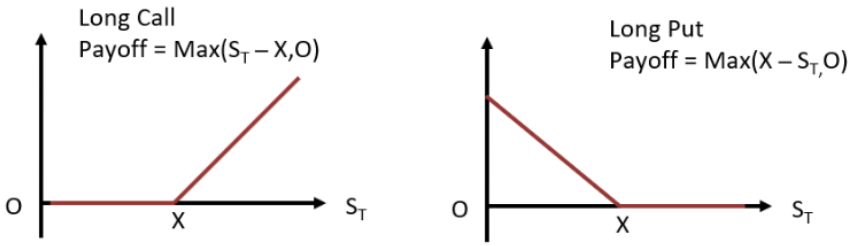

Figure 1: Terminal Payoff Profiles of Call and Put Options (*Image Credit: [2]*)

With these boundary conditions, we arrive at the following popular version of the Black-Scholes Merton (BSM) pricing formula:

$$C = S_0\Phi(d_1) - Ke^{-rT}\Phi(d_2)$$
$$P = Ke^{-rT}\Phi(-d_2) - S_0\Phi(-d_1)$$
$$d_1 = \frac{ln(S_0/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}},\ d_2 = d_1 - \sigma\sqrt{T}$$

where Φ denotes the standard Gaussian CDF, K the option strike price (the price promised to buyer to transact upon later), and S_0 the initial stock price [3].

Although the exact model for the more complex American options (which allow the user to buy or sell the stock at any point upto expiration and not just at expiration) would be different, we adopt the same broad set of features in our model (for a total of 8 features):

- S: the stock price at the time of pricing the option
- K: the strike price specified in the option contract
- T: the time to maturity for the option
- σ: the volatility of the option, which we calculate as the standard deviation using stock price history. We consider 3 fields: volatility over a week, a month, and a quarter.
- r: the risk-free rate, which we proxy using 2 fields: the long-term treasury yield (10 years) and the short-term yield (1 month)

2.2 Data: Sources

Relevant Notebook Link: Notebook to Fetch and Store Data

As specified in the section on Problem Statement, we seek to measure the effectiveness of deep learning models in both approximating the general form of BSM formula on simulated data as well as to capture the actual pricing relationship for options with real-world data. We, therefore, use two sets of data specified as follows:

I. Simulated Data
We simulate 100,000 instances each for a hypothetical call and option using the BSM formula specified earlier. In particular, we use the lognormal distribution to generate S and σ, add uniform white noise to S to get K, use the Gaussian distribution for r, and a uniform distribution for T. Finally, we plug these inputs into the BSM formula to get C and P, which serve as simulated labels or ground truth for our models.

II. Real-World Data
For this project, we restrict our focus to options on Apple and Google stocks, but the model can be applied to options for any stock. Data for 2019-2025 were obtained from the sources in Table 1. Our final dataset has around 50,000 rows each for call and put options on Apple and Google stocks each. We keep 80% of the data for fitting the model, and the remaining data out-of-sample. Furthermore, as is the standard practice in finance, we use temporal partitioning in this project; that is, we train our model on older instances and test/validate them on the more recent ones. Figure 2 shows a snapshot of the data for a call option on AAPL.

Data	Source	Description
Options	Archived Yahoo Finance data on Dolthub [4]	Daily prices for AAPL and GOOG call and put options at different strike prices and times to expiration
Stock	Yahoo Finance	Daily stock price for AAPL and GOOG
Risk free rate	US Treasury Archive [5]	Yields of risk free bonds with different terms

Table 1: Data Sources

date	option_price	stock_price	strike	time_to_maturity	volatility_week	volatility_month	volatility_quarter	short_term_yield	long_term_yield
2024-12-31	1.345	250.144974	275.0	0.142466	0.012282	0.010245	0.010573	4.4	4.58
2024-12-31	0.890	250.144974	280.0	0.142466	0.012282	0.010245	0.010573	4.4	4.58
2024-12-31	0.420	250.144974	290.0	0.142466	0.012282	0.010245	0.010573	4.4	4.58
2024-12-31	0.305	250.144974	295.0	0.142466	0.012282	0.010245	0.010573	4.4	4.58
2024-12-31	0.230	250.144974	300.0	0.142466	0.012282	0.010245	0.010573	4.4	4.58

Figure 2: Snapshot of the Data for an AAPL Call

2.3 Preliminary Data Analysis

We run the descriptive statistics for the simulated data with the BSM formula and the real-world data for AAPL and GOOG. A few points of interest that emerged are as follows.

Plotting the trace plots for the call option price as well as our 8 features for AAPL stock in Figure 3, we find that the general trend of values differ significantly between the early 2019-2020 and the later timestamps (2024 and after). Of particular note are the rising and relatively volatile call prices in 2020 (in wake of the onset of pandemic) followed by a sharp cliff and the rising treasury yields in 2022. This has significant implications for a financial study, where all the testing is done temporally. A strategy that is backtested on 2019-20 may not perform as well in 2022. However, since we consider the entire period 2019-25 for our study (which cover several volatile periods including the pandemic, the Ukraine crisis, the Trump elections, and so on), the in- and out-of-sample performance of our models should give us a decent idea of its robustness.

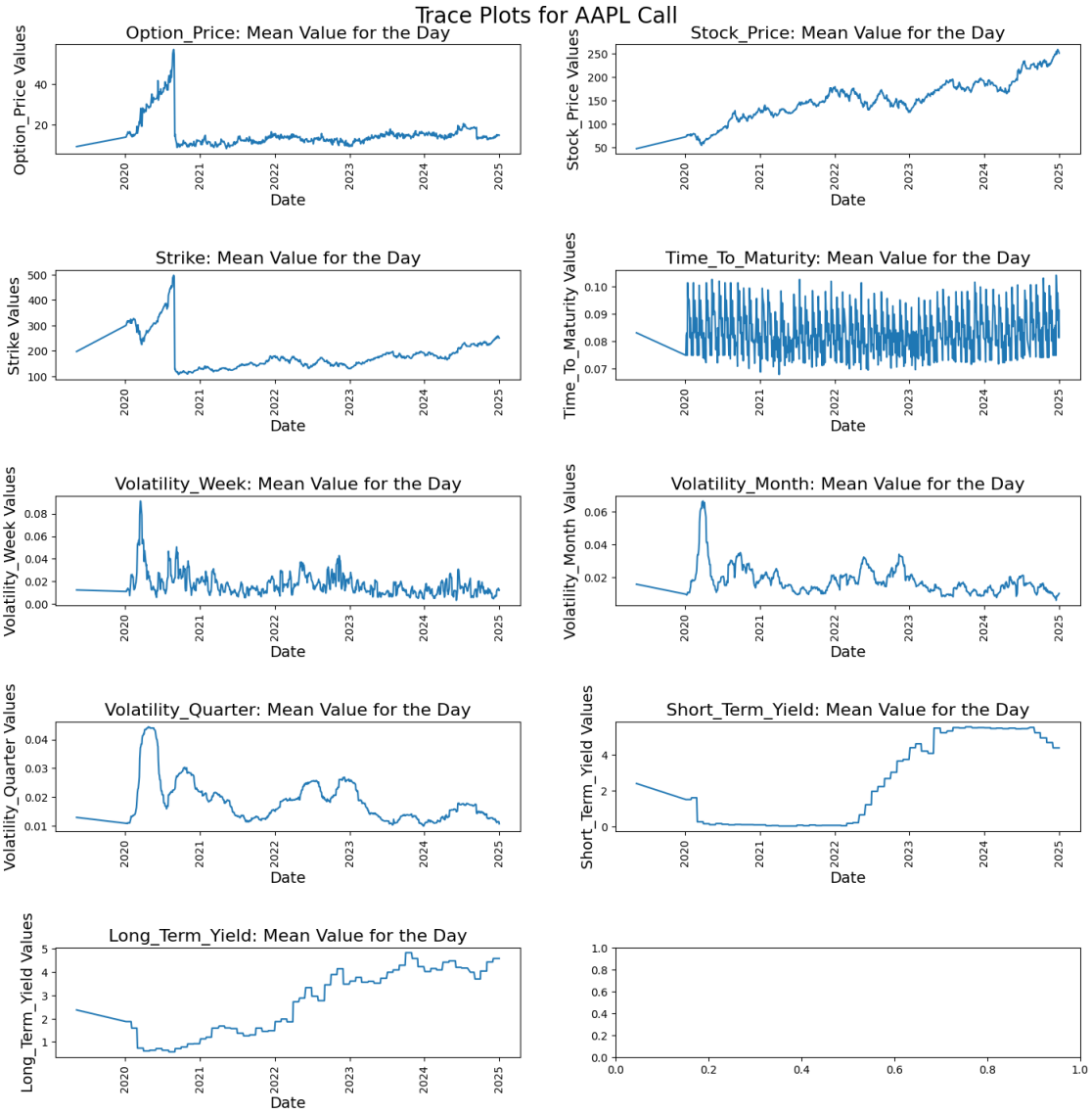

Figure 3: Trace Plots for AAPL Call Prices and Features

Another interesting point is that the actual distribution of the stock price for AAPL is far from lognormal (as shown in Figure 4), violating the Geometric Brownian motion assumption of the Black-Scholes idealized world (Figure 5). Hence, it is no surprise that the distribution of the options based on it also possess distributions significantly different from the Black-Scholes simulated

options (details in the code notebooks).

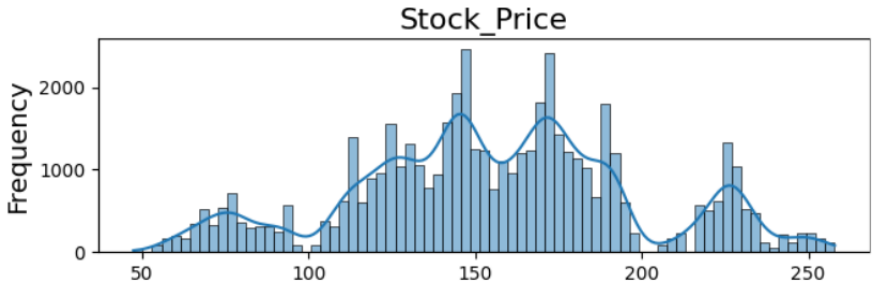

Figure 4: Distribution of AAPL Stock Price S

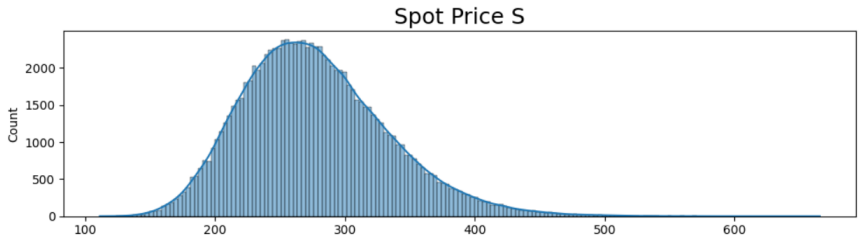

Figure 5: Distribution of Idealized Black-Scholes Stock Price S (Lognormal)

2.4 Model Overview

We fit the following models and compared their performance in the results section.

- **Polynomial regression:** We normalize our features using Standard scaling and then fit a linear regression to the standardized features, their squares, and their interaction terms, thereby getting a model of the general form

$$y = \beta_0 + \sum_{i=1}^{n} \beta_i x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_i \beta_j x_i x_j + \epsilon$$

  where y is the put or call price and x_i's are the features.

- **Deep Neural network:** We trained a neural network with the architecture shown in Figure 6 with ReLU activation in the hiddent layers over 50 epochs. We tracked the validation loss for early stopping to prevent overfitting. We predict the neural network will perform better than the linear regression since it can capture complex nonlinear relationships between volatility/risk-free rate and option prices that have been proven to exist in the real world.

- **Transformer:** As an example of a more complex model, we fit a transformer with embedding dimension = 32, 4 attention heads per transformer block, 32 units in the feedforward layer of the transformer block, 4 transformer blocks, and a final neural network with two layers with 64 and 32 units respectively. The transformer may further improve performance since it can model long-range dependencies and capture the effect of past events. This will be helpful since prices often show trends, i.e., past prices often impact future prices. The basic idea behind this model is adapted from the 2020 paper on TabTransformers [6].

- **Black-Scholes-Merton Model:** Finally, we inputted real-world data into the standard closed-form equation used for option pricing specified in Section 1. Note that the BSM formula is used only for the real-world data since the simulated data is already generated using it.

Layer (type)	Output Shape	Param #
dense_9 (Dense)	(None, 64)	576
dense_10 (Dense)	(None, 32)	2,080
dense_11 (Dense)	(None, 16)	528
dense_12 (Dense)	(None, 1)	17

Figure 6: Neural Network Architecture

3 Results

3.1 On Simulated Data

Relevant Notebook Link: Notebook for Analysis, Model Fitting, and Results on Simulated Data

Table 2 displays the MSE and RMSE noted in- and out-of-sample for our models on the data simulated using the BSM formula. Note that on the simulated data, we additionally also fitted a Shallow NN (an MLP with a single hidden ReLU layer). But since the deep model performed substantially better, we dropped it out of comparison on the real-world data in the subsequent section.

Table 2: Performance on Simulated Data

	Polynomial Regression		**Shallow NN**		**Deep NN**		**Transformer**	
	MSE	**RMSE**	**MSE**	**RMSE**	**MSE**	**RMSE**	**MSE**	**RMSE**
IS	49.267	7.019	11.569	3.401	1.937	1.392	5.789	2.406
OOS	49.893	7.063	11.647	3.412	1.964	1.401	5.229	2.286

The difference between the neural network-based models and the traditional regression-based approach is substantial: the out-of-sample MSE shrinks by more than a factor of $25\times$ in going from polynomial regression to deep NN. This points to the rightful claim that NNs can be extremely powerful universal function approximators [7].

Additionally, the transformer, while suited to complex NLP and vision problems, performs slightly worse than a deep NN in this case. What is more concerning, however, is that the transformer learns spurious patterns for extreme values of the parameters (which is precisely where we would need an ML model over the traditional BSM model for pricing). This is apparent in Figure 7, which plots predicted call C price against the risk-free rate r for the models, also showing the true BSM-based simulated price. The transformer diverges *very* substantially for high values of interest rate, observed in highly inflationary economy (think US in 1970s [8])

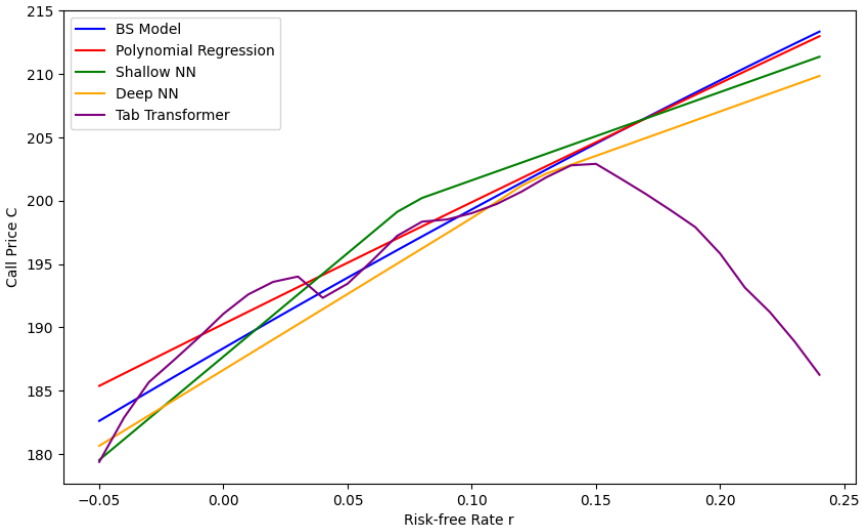

Figure 7: Predicted Call Price on Simulated Data against Risk-free Rate r

3.2 On Real-World Data

Relevant Notebook Links:

- Notebook for Models and Results for Call Options on AAPL Stock
- Notebook for Models and Results for Put Options on AAPL Stock
- Notebook for Models and Results for Call Options on GOOG Stock

Table 3 shows the relative performance of the models both in- and out-of-sample, as measured by MSE and RMSE. It is easy to see that the fully connected deep neural network emerges as the winner in either case, shrinking the MSE of BSM formula by more than a factor of 4. Also, while the transformer and polynomial regression beat the BSM formula in-sample, they do not perform well out-of-sample, indicating that they might have learnt non-generalizable or noisy patterns.

Table 3: Performance Comparison of Models: AAPL Call

	BSM Formula		**Polynomial Reg.**		**Deep NN**		**Transformer**	
	MSE	**RMSE**	**MSE**	**RMSE**	**MSE**	**RMSE**	**MSE**	**RMSE**
IS	487.58	22.08	269.90	16.43	83.48	9.14	107.28	10.36
OOS	4.53	2.13	3672.01	60.60	1.08	1.04	12.84	3.58

Next, we demonstrate that these results are not particular to a specific stock or to a specific family of options by showing results for a call option on GOOG (Table 5) and a put option on AAPL (Table 4) below. Note that the deep NN again outperforms the other models, while polynomial regression (despite being the favored model in most quant desks in hedge funds and investment banks) stands out as the worst performing model for the application.

Something that might have immediately stood as odd to a keen eye would be that the in-sample errors are distinctly larger than the out-of-sample errors in almost all the results displayed here. In ML and inferential statistics, in general, a model performs better on the data it is fitted upon than

Table 4: Performance Comparison of Models: AAPL Put

	BSM Formula		Polynomial Reg.		Deep NN		Transformer	
	MSE	**RMSE**	**MSE**	**RMSE**	**MSE**	**RMSE**	**MSE**	**RMSE**
IS	8422.24	91.77	93.38	9.66	3.35	1.83	14.49	3.81
OOS	3.64	1.91	290.4	17.03	1.51	1.23	1.69	1.30

Table 5: Performance Comparison of Models: GOOG Call

	BSM Formula		Polynomial Reg.		Deep NN		Transformer	
	MSE	**RMSE**	**MSE**	**RMSE**	**MSE**	**RMSE**	**MSE**	**RMSE**
IS	52716.7	229.6	7467.3	86.4	176.8	13.3	2859.1	53.5
OOS	4.2	2.1	46995.0	216.8	1.1	1.0	29.8	5.5

new unseen data. This perceived anomaly, however, is resolved if we inspect how the residuals, or errors, vary over time. Figure 8 plots the mean absolute errors for each day for the entire period studied with deep NN model (our best model) on GOOG Call. It is immediately obvious that there is a period in 2020 (lying within the training period), where residuals are distincly larger. This can be tied back to the especially volatile option prices at the onset of the pandemic, evident in Figure 3. Thus, we conclude that the training period was actually harder to explain than the later out-of-sample period, even for a human analyst.

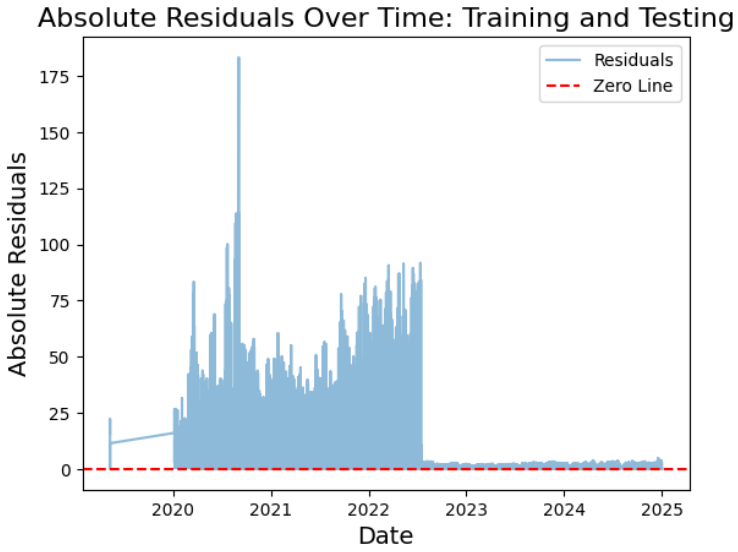

Figure 8: Daily Mean Absolute Residuals with Deep NN for GOOG Call

Finally, we investigate how the errors behave over the range of values assumed by the features. Fortunately, the errors remained fairly constant across all parameter values (details in the code notebooks) save for those of Time to Maturity T, for which the performance seemed to deteriorate for extremely small or large values (Figure 9). This may be explained as the inability of the models to learn about cases where the time value of option (measured by θ in financial literature [9]). This

is a fairly known problem in finance when pricing options with statistical models, and is typically solved with Finance-informed NNs [10].

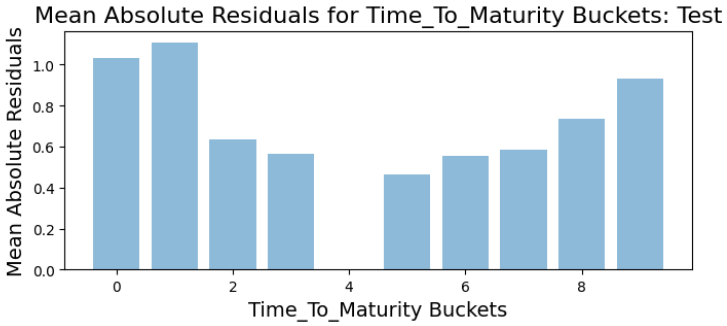

Figure 9: Distribution of Residuals for Different Values of T

4 Lessons Learned

This project explored the effectiveness of deep learning models in option pricing, comparing their performance with traditional regression-based methods and the widely used Black-Scholes-Merton (BSM) model. The following key insights emerged from our analysis:

I. Reluctance to Non-Interpretable Models in Finance
A major takeaway from our experiments was the unjustified skepticism and reluctance [11] in the financial industry toward using non-interpretable models such as deep NNs. Our results demonstrated that DL models significantly outperformed traditional regression techniques, including polynomial regression, on both real-world and simulated data. This suggests that while transparency and interpretability remain a concern, the predictive power of neural networks should not be overlooked in financial applications.

II. Complexity and Parsimony Tradeoff in Model Selection
Although deep learning did prove more effective than the closed form BSM formula as well as the traditional regression model, the performance deteriorated when the more complex transformer architecture was introduced. While transformers have been immensely successful in NLP and vision applications, they are far from a universal panacea. We conclude that the model should be complex enough to capture the underlying essential non-linearities in the data, but parsimonious enough to not overfit and be robust over the entire range of data (unlike the transformer-based model in our case, whose performance degraded for high values of risk-free rate).

III. Modeling Considerations for Real-World Application
The temporal partitioning strategy (common in financial settings) we employed showed that model performance can vary significantly over time. In fact, the in-sample MSE was found larger than out-of-sample MSE for our models due to an especially turbulent pandemic year. This suggests the industry should consider adaptive learning approaches to maintain model performance over long periods and to account for such volatile periods, including combining data from periods covering diverse ranges of parameter values, augmenting with simulated data, and rolling the models frequently to capture the newest information.

5 Conclusion and Future Work

While DL models show promise in financial modeling, particularly in option pricing, their adoption requires overcoming industry resistance to non-interpretable methods. The tradeoff between model complexity and robustness must be carefully managed, and while deep networks offer substantial performance gains over regression, excessively complex architectures, such as transformers, may not always be suitable. Future work should focus on explainable AI techniques to bridge the gap between predictive performance and interpretability, fostering greater trust in these models within financial institutions. Additionally, robustness of the models may be investigated further by increasing the duration of the period studied. Finally, augmenting a deep learning model with financial closed-form equations (similar to Physics-informed models) may increase the performance over the entire domain of the paramater values.

References

[1] J.C. Hull, *Options, Futures, and Other Derivatives*, 11th ed., Pearson Education, 2021.

[2] J. Forjan, "Options calculations: Payoff and profit," *AnalystPrep*, [Online]. Available: https://analystprep.com/blog/options-calculations-payoff-profit/. [Accessed: March, 2025].

[3] R. C. Merton, "Theory of rational option pricing," *Bell J. Econ. Manage. Sci.*, vol. 4, no. 1, pp. 141–183, 1973, doi: 10.2307/3003143.

[4] DoltHub. (n.d.). Options. [Online]. Available: https://www.dolthub.com/repositories/post-no-preference/options. [Accessed: March, 2025].

[5] U.S. Department of the Treasury. (n.d.). Interest rate statistics. [Online]. Available: https://home.treasury.gov/policy-issues/financing-the-government/interest-rate-statistics. [Accessed: March, 2025].

[6] X. Huang, A. Kheltan, M. Cvitkovic, Z. Karnin, "TabTransformer: Tabular Data Modeling Using Contextual Embeddings," *arXiv preprint arXiv:2012.06678*, 2020. [Online]. Available: https://arxiv.org/abs/2012.06678. [Accessed: March, 2025].

[7] S Liang, R. Srikant, "Why Deep Neural Networks for Function Approximation?" *arXiv preprint arXiv:1610.04161*, 2016. [Online]. Available: https://arxiv.org/abs/1610.04161. [Accessed: March 2025].

[8] M. Bryan, "The Great Inflation," *Federal Reserve History*, [Online], Available: https://www.federalreservehistory.org/essays/great-inflation. [Accessed: March, 2025]

[9] R. McKeon, "Empirical Patterns of Time Value Decay in Options," *China Finance Review International*, Emerald Group Publishing Limited, vol 7(4), pp. 429-449, September 2016.

[10] Y. Qu and M. -X. Wang, "The Option Pricing Model Based on Time Values," 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 2021, pp. 1-8, doi: 10.1109/IJCNN52387.2021.9533740.

[11] S.K. Das, S. Anwar, U. Tulsyan, Y. Gupta, R. Vudutta, S. Gardezi, "Role of AI in Financial Markets: Impacts on Trading, Portfolio Management, and Price Prediction," Journal of Electrical Systems, Vol 20, No 6s, 2024