Proceedings of the 3rd International Conference on Financial Technology and Business Analysis
DOI: 10.54254/2754-1169/2024.26931

Deep Learning Approaches for Pricing Options in Stochastic Volatility Models

Yang Zheng

North Carolina State University, Raleigh, USA
koroly1119@gmail.com

Abstract. Efficient computation of option prices is essential for making quick trading decisions. This paper investigates the use of deep learning to expedite the accurate calculation of European option prices within a local volatility framework that utilizes five parameters. We compared the predictions of the deep neural networks against results obtained from the Monte Carlo method across various scenarios. Our numerical experiments indicate that the approximation network achieves satisfactory accuracy. The network performs exceptionally well within the core region of the parameter domain.

Keywords: Local volatility model, European call option, Finite difference method, Monte Carlo simulation, Deep learning.

1. Introduction

In the realm of financial mathematics, swiftly computing option prices is a vital task [1]. These calculations are critical for tuning option pricing models and facilitating rapid trading decisions. The fluctuating nature of option prices over time is better represented by stochastic volatility models rather than the constant volatility models. Local stochastic volatility models frequently do not offer closed-form solutions. Various numerical methods can be used to price options, such as finite difference methods, Monte Carlo simulation, and finite element methods [2] [3].

Deep learning employs artificial neural networks with multiple layers called deep neural networks to tackle and solve complex problems [4]. Deep neural networks can approximate any function, presenting a novel strategy for option pricing. Compared to traditional methods, they handle complex problems and provide real-time option prices with low latency.

In this study, the local volatility model was selected for its relative simplicity and capacity to facilitate efficient option price computation using classical methods. Nonetheless, machine learning techniques can enhance the calculation of option price of the local volatility model, making this model an excellent test case for such approaches. The financial industry's interest in machine learning is driven by its potential applicability to more complex models.

This paper is organized as follows: Section 2 introduces the finite difference method for the local volatility model and uses Monte Carlo simulations to create a dataset for training deep networks. Section 3 discusses how deep learning can expedite option price computation under the local volatility model. After training, the deep networks can instantly provide option prices based on the input parameters. The final section presents a summary and conclusions.

DOI: 10.54254/2754-1169/2024.26931

2. Option Pricing via Numerical Methods

Numerical techniques can price complex options that lack closed-form solutions. This section introduce the finite difference method of the local volatility model and the Monte Carlo method. The latter one is used to generate the dataset for training the deep network in the next section.

Consider a scenario where the stock price S satisfies a geometric Brownian motion (GBM), which yields a continuous dividend q, adheres to the following stochastic differential equation (SDE) [2],

$$dS_t = (r - q)S_t dt + \sigma(S_t, t)S_t dW_t, \tag{1}$$

where

- r, the risk-free interest rate.
- W, the standard Brownian motion.
- $\sigma(S_t, t)$, a local volatility function.

The local volatility function is defined as

$$\sigma(S_t, t) = \sigma e^{-t}(100/S_t)^{\alpha}.$$

It simplifies to the Constant Elasticity of Variance (CEV) model when the local volatility function has the following form [5]

$$\sigma(S_t, t) = \sigma S_t^{\alpha - 1}.$$

If the parameter $\alpha = 1$, the above CEV model reduces to the classic Black-Scholes model [14].

The foundational principles for option pricing were established by Black and Scholes, refer to the Black¨CScholes equation [14]:

$$\frac{\partial V}{\partial t} + (r - q)S\frac{\partial V}{\partial S} + \frac{1}{2}\sigma(S_t, t)^2 S^2 \frac{\partial^2 V}{\partial^2 S} - rV = 0, \tag{2}$$

which is a partial differential equation (PDE). The simplest financial option is the European option. A European call option gives the holder the right to buy the underlying asset at a prescribed price (K), which can only be exercised at expiration, With strike price K and maturity T, the price of the European call option is:

$$C(S, T) = e^{-(r-q)T} E[(S(T) - K)^+]. \tag{3}$$

The European call option satisfies the following boundary conditions:

- Finally condition: payoff $f(S, T) = (S - K)^+$.
- Dirichlet boundary conditions:

$$\begin{aligned} V(S_{min}, \tau) &= 0, \\ V(S_{max}, \tau) &= e^{-q\tau} S_{max} - e^{-r\tau} K_0. \end{aligned}$$

- Neumann boundary conditions:

$$\begin{aligned} \lim_{S \to 0} \frac{\partial^2 V}{\partial S^2} &= 0, \\ \lim_{S \to \infty} \frac{\partial^2 V}{\partial S^2} &= 0. \end{aligned} \tag{4}$$

DOI: 10.54254/2754-1169/2024.26931

The expiry date is denoted by T and the exercise price by K.

Typically, there is no closed-form solution for the European call option price in a general model [14]. However, analytical solutions for European put options are available in Heston [14]. Several numerical techniques can be employed to solve the PDE (2) along with its boundary conditions.

We now explain the procedure to solve the European option pricing problem with the finite difference method, followed by Monte Carlo simulations. By defining $\tau = T - t$, Equation (2) can be rewritten as:

$$\frac{\partial V}{\partial \tau} = \frac{1}{2}\sigma(S,\tau)^2 S^2 \frac{\partial^2 V}{\partial^2 S} - r(\tau)V + (r(\tau) - q(\tau))S\frac{\partial V}{\partial S}. \tag{5}$$

The Black-Scholes equation is defined over the domain $[0,T] \times [0,\infty]$. The option price $V(S,\tau)$ is computed on a grid with $(M+1) \times (N+1)$ points within the domain $[0,T] \times [0,S_{\max}]$:

- S_{max} is a large number to approximate ∞ and $\tau \in [0,T]$.
- $S_1 = S_{min}$ and $S_{N+1} = S_{max}$.

Then the explicit discretization is [2]

$$\frac{V_{j,k+1} - V_{j,k}}{\Delta\tau} - \frac{\sigma_{j,k}^2 S_j^2}{2}\frac{V_{j+1,k} - 2V_{j,k} + V_{j-1,k}}{\Delta S^2} + r_k V_{j,k} - (r_k - q_k)S_j \frac{V_{j+1,k} - V_{j-1,k}}{2\Delta S} = 0. \tag{6}$$

We introduce the following notations

$$m_{j,k} = \frac{\sigma_{j,k}^2 S_j^2}{2}\frac{\Delta\tau}{\Delta S^2}, \qquad n_{j,k} = (r_k - q_k)S_j \frac{\Delta\tau}{2\Delta S}.$$

After rearranging equation (6), we have

$$V_{j,k+1} = (m_{j,k} - n_{j,k})V_{j-1,k} + (1 - r_k\Delta\tau - 2m_{j,k})V_{j,k} + (m_{j,k} + n_{j,k})V_{j+1,k}.$$

Using notations $a_{j,k} = m_{j,k} - n_{j,k}$, $d_{j,k} = 1 - r_k\Delta\tau - 2m_{j,k}$, and $b_{j,k} = m_{j,k} + n_{j,k}$, the above equation reduces to

$$V_{j,k+1} = a_{j,k}V_{j-1,k} + d_{j,k}V_{j,k} + b_{j,k}V_{j+1,k}.$$

In the matrix form, we have

$$\begin{pmatrix} V_{2,k+1} \\ V_{3,k+1} \\ \vdots \\ V_{N,k+1} \end{pmatrix} = \begin{pmatrix} d_{2,k} & b_{2,k} & & & \\ a_{3,k} & d_{3,k} & b_{3,k} & & \\ & \ddots & \ddots & \ddots & \\ & & a_{N-1,k} & d_{N-1,k} & b_{N-1,k} \\ & & & a_{N,k} & d_{N,k} \end{pmatrix} \begin{pmatrix} V_{2,k} \\ V_{3,k} \\ \vdots \\ V_{N,k} \end{pmatrix} + \begin{pmatrix} a_{2,k}V_{1,k} \\ \vdots \\ 0 \\ b_{N,k}V_{N+1,k} \end{pmatrix} \tag{7}$$

The boundary conditions (4) are as follows:

- $$f(T,S_T) = (S_T - K, 0)^+.$$
- The first Neumann boundary condition leads to

$$\frac{\partial^2 V(0,\tau_{k+1})}{\partial S^2} = \frac{\partial^2 V(0+\Delta S,\tau_{k+1})}{\partial S^2} = \frac{V_{1,k+1} - 2V_{2,k+1} + V_{3,k+1}}{\Delta S^2} = 0.$$

which leads to

$$V_{1,k} = 2V_{2,k} - V_{3,k}. \tag{8}$$

DOI: 10.54254/2754-1169/2024.26931

– The second Neumann boundary condition leads to

$$\begin{aligned}\frac{\partial^2 V(S_{max},\tau_{k+1})}{\partial S^2} &= \frac{\partial^2 V(S_{max}-\Delta S,\tau_{k+1})}{\partial S^2}\\ &= \frac{V_{N+1,k+1}-2V_{N,k+1}+V_{N-1,k+1}}{\Delta S^2}\\ &= 0\end{aligned}$$

which leads to

$$V_{N+1,k} = 2V_{N,k} - V_{N-1,k}. \tag{9}$$

By integrating equation (7) with the boundary conditions specified in formulas (8) and (9), we derive an iterative approximate solution to the partial differential equation (2) for the time increment $\tau_k = (k-1)\Delta\tau$, represented as:

$$\begin{pmatrix} V_{2,k+1}\\ V_{3,k+1}\\ \vdots\\ V_{N,k+1}\end{pmatrix} = \begin{pmatrix} d_{2,k}+2a_{2,k} & b_{2,k}-a_{2,k} & & & & \\ a_{3,k} & d_{3,k} & b_{3,k} & & & \\ & \ddots & \ddots & \ddots & & \\ & & a_{N-1,k} & d_{N-1,k} & b_{N-1,k} & \\ & & & a_{N,k}-b_{N,k} & d_{N,k}+2b_{N,k} & \end{pmatrix} \begin{pmatrix} V_{2,k}\\ V_{3,k}\\ \vdots\\ V_{N,k}\end{pmatrix},$$

which is equal to

$$V_{k+1} = A_k V_k \tag{10}$$

Upon completing the iteration, V_{M+1} yields the value of the European call option for $S_0 \in (S_{min}, S_{max})$, K, and T. The complex option pricing with improved finite difference method is detailed in [12].

However, the finite difference method described by formula (10) suffers from stability and convergence issues with certain parameter values in the local volatility model. In contrast, the Monte Carlo method provide a natural probabilistic interpretation of the solution and is more robust. Therefore, we use it to the generate the Data set to train the NN in next section.

The Monte Carlo method simulates the stock's path using the following iterative relation:

$$S(t_{j+1}) = S(t_j)\exp((r-q-\frac{\sigma(S_j,t_j)^2}{2})\Delta t + \sigma(S_j,t_j)^2\sqrt{\Delta t}Z_j), \tag{11}$$

where

$$Z_j \sim N(0,1).$$

To enhance the efficiency of the Monte Carlo estimate, often measured by its variance, antithetic sampling is employed. This reduce this variance by generating an additional set of paths:

$$S(t_{j+1}) = S(t_j)\exp((r-q-\frac{\sigma(S_j,t_j)^2}{2})\Delta t - \sigma(S_j,t_j)^2\sqrt{\Delta t}Z_j). \tag{12}$$

The European call option price is then determined by averaging the outcomes over n paths, given by the following expression:

$$C(S,T) = \frac{e^{-(r-q)T}E[\sum(S(T)_i - K)^+]}{n}. \tag{13}$$

In summary, there are five parameters in the model (1) . Given the values of these five parameters, we can obtain an option price for pair of values of K and T, as shown in Table 1. The dataset for training the deep learning networks will be organized in this form.

DOI: 10.54254/2754-1169/2024.26931

Table 1: Parameters and the corresponding prices.

Parameters	Mesh grids on T and K parameters plane
$r, q, \sigma, \alpha, S_0,$	Option prices

3. Price Generation Using Deep Learning

The local volatility model is relatively straightforward, enabling the rapid computation of option prices through numerical methods [2]. However, this model can be substantially enhanced by employing machine learning techniques, making it an excellent candidate for such innovative approaches [6] [7].

In deep learning, neural networks are designed and trained using datasets. The training process involves determining the optimal weights within the network. Ultimately, predictions are generated by inputting data into the trained network.

3.1. Deep Learning

In [7], the pricing functionals of options $y = F(x, w)$ is approximated by deep neural networks, with model parameters x and option prices y. The initial step involves constructing the neural networks. A single node in these networks is depicted in Fig.(1).

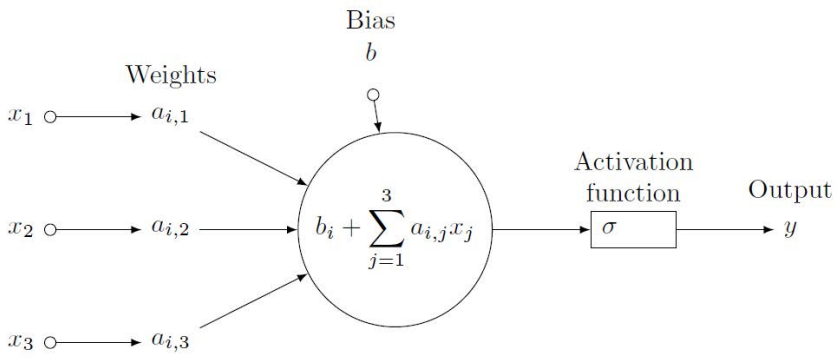

Figure 1: Node in the Network architecture.

The weights terms w and bias terms b for each node are defined by the following functions:

$$F_l : \mathbb{R}^{N_l} \xrightarrow{w^{l+1}x+b^{l+1}} \mathbb{R}^{N_{l+1}} \xrightarrow{\sigma} \mathbb{R}^{N_{l+1}}$$

where the superscript l index the number of layers and N_l is the number of neurons in the lth layer. σ_l is the activation function. Then the Neural Network of L layers is defined as:

$$F = F_L \circ \cdots \circ F_1.$$

with $W^l = w^{l+1}x + b^{l+1}$ and $F_l = \sigma_l \circ W^l$.

It has been proved that Neural networks are capable of approximating any function effectively [9]. The network architecture used to price option is illustrated in Fig.(2), mirroring the one utilized in [7][8]:

DOI: 10.54254/2754-1169/2024.26931

1. A fully connected neural network.
2. 4 hidden layers.
3. Number of input dimensions: 5.
4. Each inner layer contains 30 nodes.
5. Output dimension: $8(maturities) \times 11(strikes)$.
6. Number of weights is calculated as:

$$(5+1) \times 30 + 4 \times (1+30) \times 30 + (30+1) \times 88.$$

7. Activation function is σ_{Elu}.

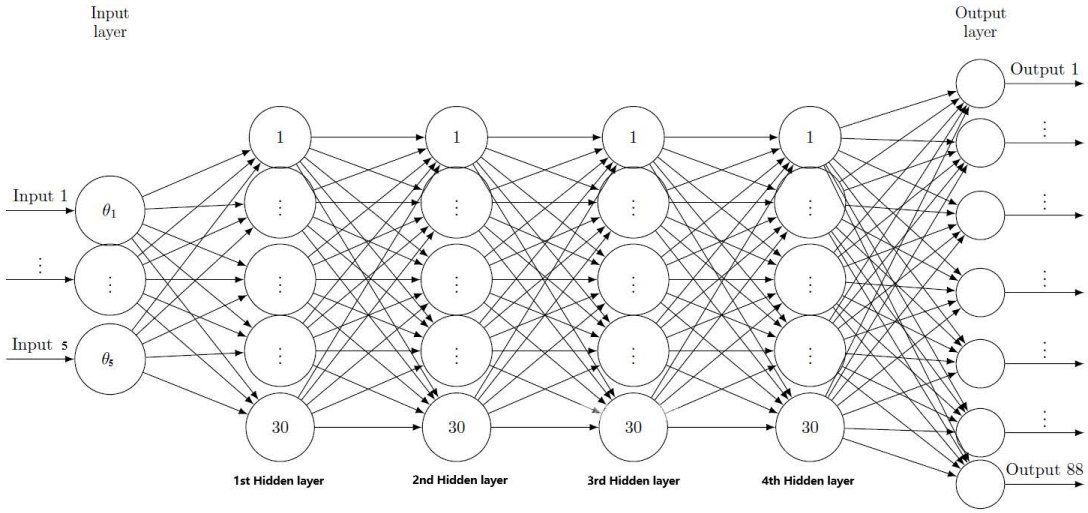

Figure 2: Network architecture.

Note that the number of input parameters of the model is 5. And the number of output dimensions corresponds to the number of mesh grids on K and T coordinates plane, which is related to implied volatility surface directly [6].

3.2. Option Pricing via Deep Learning

The second step in deep learning involves using a dataset to train the networks.

Dataset:

Let the price of a European call option based on the model outlined in Equation (13) be

$$C^E(r, q, \sigma, \alpha, S_0, K, T).$$

The parameter ranges based on existing market data are as follows:

- $r \in (0.01, 0.09)$: Risk-free interest rate.
- $\sigma \in (0.1, 0.5)$: Volatility of the underling variance.

DOI: 10.54254/2754-1169/2024.26931

- $q \in (0.01, 0.09)$: Dividend yield.
- $\alpha \in (0, 2)$.
- $S_0 \in (30, 100)$: Spot stock price.
- $T \in (0.8, 1.5)$: 8 fixed maturity times in intervals of 0.1..
- $K \in (20, 120)$: 11 strike prices in intervals of 10.

The parameters q, α, σ, S_0, and r are generated randomly within their respective ranges. For each combination of T and K from the mesh grid, we compute the call option price using Monte Carlo simulation as specified in equation (12). The structure of one row of the dataset is illustrated in the following Table 2.

Table 2: One sample of the dataset.

Parameters:5 columns	Option Prices:88 columns
$r, q, \sigma, \alpha, S_0$	Corresponding to T and K mesh grids

We generated a dataset containing 5000 rows through the Monte Carlo method described earlier, resulting in 5000×88 European call option prices. The entire data generation process took approximately 1.5 hours, which is within acceptable limits.

Training the network and output:

The architecture of our neural network is shown in Fig.(2). The input dimension is 5, corresponding to the parameters r, q, σ, α, and S_0. The output dimension is 11×8, corresponding to the strike prices (K) and maturities (T). The parameters T and K are encoded in the output plane as coordinates. Each hidden layer in the network consists of 30 nodes. The total number of network parameters is calculated as follows:

$$(5 + 1) \times 30 + 3 \times (1 + 30) \times 30 + (30 + 1) \times 88 = 5698.$$

With 88×5000 option price samples, the dataset is sufficiently large to train the network effectively, ensuring it performs well. The dataset is split into training and test sets at a ratio of 0.15.

Our neural network demonstrates proficiency in estimating option prices.

1. Fig.(3) illustrates one row of the sample in the test set and the network's prediction. The eight subplots correspond to different maturities T ranging from 0.8 to 1.5. Each subplot shows the relationship between option prices and the defined variable $\ln(\text{strikes}/S_0)$, given the same five input parameters. From the figure, the predictions of the neural network are almost identical to those given by the test sample for the first six subplots. The discrepancy in the last two subplots is larger than in the first six, but the trends of both curves are the same.
2. Table 3 below showcases several call option prices determined via both the neural network (NN) and the Monte Carlo (MC) method. The relative error of the two methods is defined as follows:

$$e\% = \left| \frac{NN - MC}{NN} \right| \times 100.$$

The table indicates that the discrepancies between the MC and NN prices are generally minimal. The maximum observed relative error is:

$$e\%_{largest} = 2.3232\%.$$

DOI: 10.54254/2754-1169/2024.26931

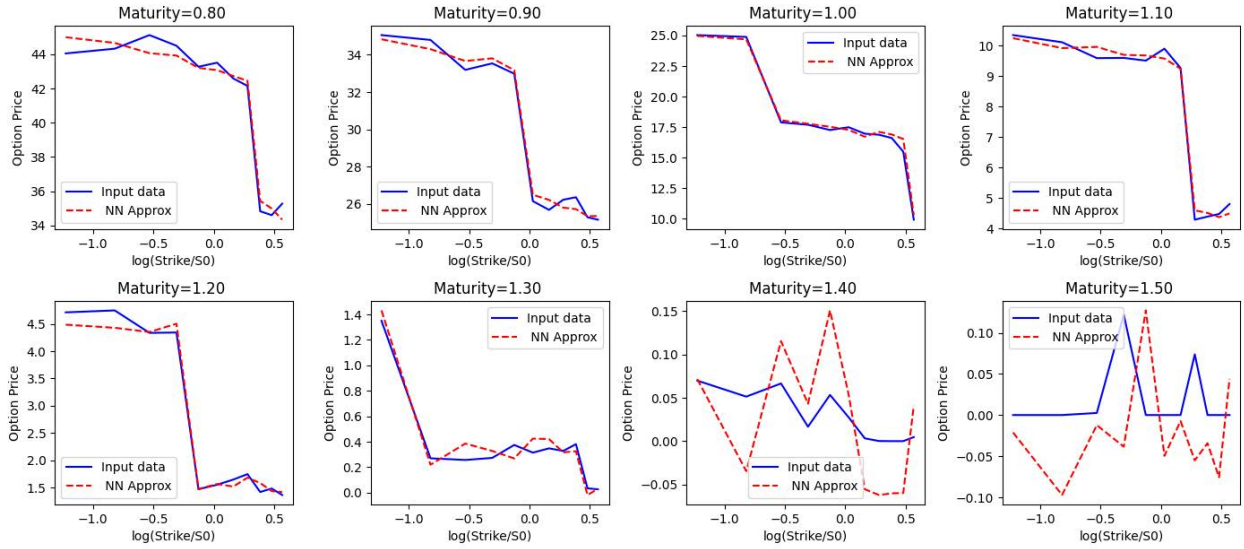

Figure 3: Option prices of one sample and the prediction given by the neural network with the same parameters.

This relative error is acceptable through is bigger than the other ones. The primary reason is that $r = 0.01$ lies at the parameter range's boundary, where the network¡¯s performance is less optimal due to the dataset having samples only from one side of this boundary.

Table 3: Call option prices computed using MC and NN methods.

r	q	σ	α	S_0	K	T	NN	MC	$e\%$
0.01	0.02	0.25	0.3	50	20	0.8	29.5338	28.8632	2.3232
0.02	0.03	0.25	0.3	60	30	0.9	29.2858	28.7724	1.7845
0.03	0.04	0.25	0.3	70	40	1	28.6143	28.7114	0.3380
0.04	0.05	0.25	0.3	90	50	1.1	28.0427	27.5850	1.6590

3. Two side-by-side charts are plotted: one showing the European call option prices calculated using the MC method (left) and the other using the NN method (right) with the following parameters:
   - $r = 0.02, q = 0.03, \sigma = 0.25, \alpha = 0.3$, and $T = 1$ as

$$S_0 \in (40, 50, 60, 70, 80, 90, 100, 110).$$

   - Three lines in Fig.(4) correspond to $K = 70$, $K = 80$, and $K = 90$.

   The lines on the right chart closely resemble those on the left, demonstrating that the NN method yields option prices similar to those obtained by the MC method.

The findings from Fig.(3), Table 3, and Fig.(4) consistently show that the option prices produced by NN are nearly identical to those derived from MC under the same parameters. While the relative error increases when the input parameters approaching the boundary of the domain, the overall discrepancy remains acceptable, indicating that the neural network's performance is satisfactory.

DOI: 10.54254/2754-1169/2024.26931

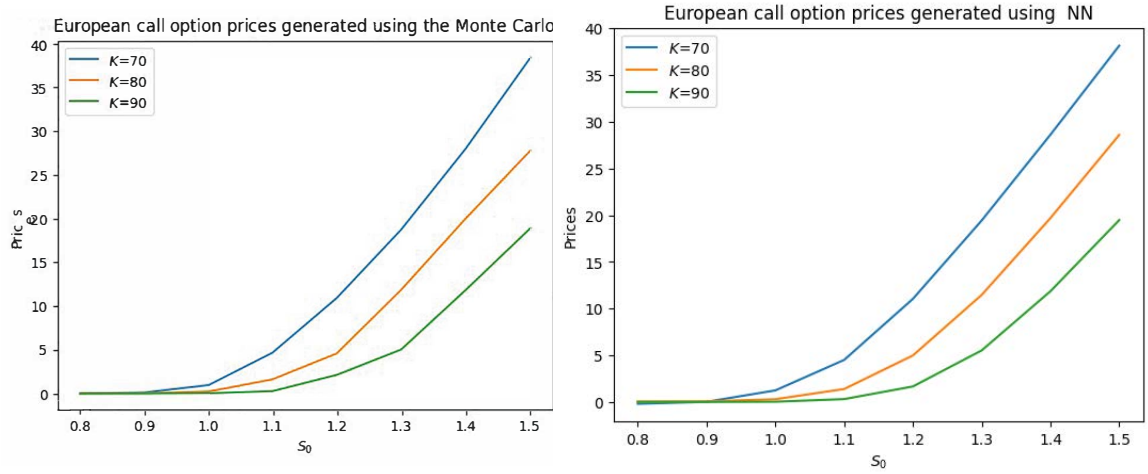

Figure 4: Option prices computed using MC and NN methods, with fixed parameters values $r = 0.02, q = 0.03, \sigma = 0.25, \alpha = 0.3$, and $T = 1$.

4. Conclusions

Fast computation of option prices facilitates quicker trading decisions. However, stochastic volatility models, like the CEV and Heston models, lack closed-form solutions for options, necessitating numerical methods. Deep learning emerges as a robust alternative to traditional techniques like finite difference (FD) and Monte Carlo (MC), adept at managing the complexities and high-dimensional data of modern financial markets. Its real-time pricing capability makes deep learning particularly appealing for option pricing.

In this project, European options under a stochastic volatility model with parameters $r, q, \sigma, \alpha, S_0$, and K were priced using both MC and NN methods, and their results were compared. The employed network architecture is fully connected, comprising four hidden layers with thirty nodes each. A dataset with 5000 rows was generated using the MC method to train the network. The input parameters include r, q, σ, α, and S_0, corresponding to the first five columns in the dataset. The next 88 columns correspond to prices on the mesh grids on the K and T plane, which are the 11×8 output. The dataset's sample size greatly influences network quality; generally, more samples lead to improved performance. Our dataset, generated over several hours, contained significantly more samples than the network weights, ensuring sufficient training.

We assessed the network¡¯s prediction accuracy in various aspects. According to Figs.(3)(4), the neural network¡¯s approximation performance is excellent, with option prices closely matching those obtained by the MC method. Several European call option prices are listed in Table 3. Two charts in Fig.(4) show European call option prices obtained by different methods. Relative errors are small within the parameter domain, but the discrepancy at the boundary is due to insufficient training data at the boundary.

It's crucial to recognize that the success of deep learning models hinges on high-quality data, appropriate model selection, and meticulous training and validation. Clearly, there is more work to be done. Instead of the fixed output dimension, we could design a universal network to give the option price directly. The next step involve conduct deep calibration of observed data in market [6].

References

[1] C Bayer, P Friz, and J Gatheral. *Pricing under rough volatility*. Quantitative Finance, 16(6):887904, 2016.

DOI: 10.54254/2754-1169/2024.26931

[2] P Wilmott, S Howson, S Howison, J Dewynne, et al. *The mathematics of financial derivatives: a student introduction*. Cambridge university press, 1995.

[3] Lipton A. *Mathematical Methods For Foreign Exchange: A Financial Engineer's Approach*. 2001; World Scientific.

[4] Ian Goodfellow, Yoshua Bengio, Aaron Courville. *Deep Learning*. The MIT Press (November 18, 2016).

[5] Cox, J. *Notes on Option Pricing I: Constant Elasticity of Diffusions*. Unpublished draft, Stanford University, 1975.

[6] C Bayer, B Horvath, A Muguruza, B Stemper, and M Tomas. *On deep pricing and calibration of (rough) stochastic volatility models*. Preprint.

[7] B Horvath, A Muguruza, and M Tomas. *Deep learning volatility*. Available at SSRN 3322085, 2019.

[8] https://github.com/amuguruza/NN-StochVol-Calibrations

[9] K. Hornik. M. Stinchcombe and H. White., *Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks*. Neural Networks. Vol. 3:11, 1990.

[10] Emanuel.D, J.MacBeth, *Further Results on the Constant Elasticity of Variance Call Option Pricing Model* , Journal of Finance and Quantitative Analysis, 1982, 17, 533-554.

[11] Gatheral, J., *The Volatility Surface: A Practitioners Guide*, 2006, NewYork, NY: John Wiley & Sons.

[12] Jichao Zhao, Matt Davison, Robert M. Corless *Compact finite difference method for American option pricing*, Journal of Computational and Applied Mathematics, 206,2007, 306-321.

[13] Schroder.M, *Computing the constant elasticity of variance option pricing formula*, Journal of Finance, 1989:44,Mar., 211-219.

[14] S.L.Heston. *A closed-form solution for options with stochastic volatility with applications to bond and currency options*. The review of financial studies, 6(2):327343, 1993.