axioms

MDPI

Article

Barrier Options and Greeks: Modeling with Neural Networks

Nneka Umeorah [1,*], Phillip Mashele [2], Onyecherelam Agbaeze [3] and Jules Clement Mba [4]

[1] School of Mathematics, Cardiff University, Cardiff CF24 4AG, UK
[2] School of Economic and Financial Sciences, University of South Africa, Pretoria 0003, South Africa; mashehp@unisa.ac.za
[3] College of Arts and Sciences, Troy University, Troy, AL 36082, USA; onagbaeze@alumni.troy.edu
[4] School of Economics, College of Business and Economics, University of Johannesburg, Johannesburg 2092, South Africa; jmba@uj.ac.za
* Correspondence: umeorahn@cardiff.ac.uk

Abstract: This paper proposes a non-parametric technique of option valuation and hedging. Here, we replicate the extended Black–Scholes pricing model for the exotic barrier options and their corresponding Greeks using the fully connected feed-forward neural network. Our methodology involves some benchmarking experiments, which result in an optimal neural network hyperparameter that effectively prices the barrier options and facilitates their option Greeks extraction. We compare the results from the optimal NN model to those produced by other machine learning models, such as the random forest and the polynomial regression; the output highlights the accuracy and the efficiency of our proposed methodology in this option pricing problem. The results equally show that the artificial neural network can effectively and accurately learn the extended Black–Scholes model from a given simulated dataset, and this concept can similarly be applied in the valuation of complex financial derivatives without analytical solutions.

Keywords: barrier options; Black–Scholes model; polynomial regression; random forest regression; machine learning; artificial neural network; option Greeks; data analysis

MSC: 91G20; 91G30; 62J05; 68T07

Citation: Umeorah, N.; Mashele, P.; Agbaeze, O.; Mba, J.C. Barrier Options and Greeks: Modeling with Neural Networks. *Axioms* **2023**, *12*, 384. https://doi.org/10.3390/axioms12040384

Academic Editor: Oscar Humberto Montiel Ross

Received: 19 December 2022
Revised: 29 March 2023
Accepted: 7 April 2023
Published: 17 April 2023

1. Introduction

The concept, techniques and applications of artificial intelligence (AI) and machine learning (ML) in solving real-life problems have become increasingly practical over the past years. The general aim of machine learning lies in attempting to 'learn' data and make some predictions from a variety of techniques. In the financial industry, they offer a more flexible and robust predictive capacity compared to the classical mathematical and econometric models. They equally provide significant advantages to the financial decision makers and market participants regarding the recent trends in financial modeling and data forecasting. The core applications of AI in finance are risk management, algorithmic trading, and process automation [1]. Hedge funds and broker dealers utilize AI and ML to optimize their execution. Financial institutions use the technologies to estimate their credit quality and evaluate their market insurance contracts. Both private and public sectors use these technologies to detect fraud, assess data quality, and perform surveillance. ML techniques are generally classified into supervised and non-supervised systems. A branch of ML (supervised) techniques that have been fully recognized is deep learning, as it provides and equips machines with practical algorithms needed to comprehend the fundamental principles, and pattern detection in a significant portion of data. The neural networks, the cornerstones of these deep learning techniques, evolved and developed in the 1960s. In the fields of quantitative finance, the neural networks are applied in the optimization of portfolios, financial model calibrations [2], high-dimensional futures [3], market prediction [4], and exotic options pricing with local stochastic volatility [5].

For the methodology employed in this paper, artificial neural networks (ANNs) are systems of learning techniques which focus on a cluster of artificial neurons forming a fully connected network. One aspect of the ANN is the ability to generally 'learn' to perform a specific task when fed with a given dataset. They attempt to replicate or mimic a mechanism which is observable in nature, and they gain their inspiration from the structure, techniques and functions of the brain. For instance, the brain is similar to a huge network with fully interconnected nodes (neurons, for example, the cells) through links, also referred to as synapses, biologically. The non-linearity feature is introduced to the network within these neurons as the non-linear activation functions are applied. The non-linearity aspect of the neural network tends to approximate any integrated function reasonably well. One significant benefit of the ANN method is that they are referred to as 'universal approximators'. This feature implies that they can fit any continuous function, together with functions having non-linearity features, even without the assumption of any mathematical relationship which connects the input and the output variables. Essentially, the ANNs are also fully capable of approximating the solutions to partial differential equations (PDE) [6], and they easily permit parallel processing, which facilitates evaluation processes on graphics processing units (GPUs) [7]. The presence of this universal approximator function is often a result of their typical architecture, training and prediction process.

Meanwhile, due to the practical significance of the use of financial derivatives, these instruments have sharply risen in more recent years. This has led to the development of sophisticated economic models, which tend to capture the dynamics of the markets, and there has also been an increase in proposing faster, more accurate and more robust models for the valuation process. The pricing of these financial contracts has significantly helped manage and hedge risk exposure in finance and businesses, improve market efficiencies and provide arbitrage opportunities for sophisticated market participants. The conventional pricing techniques of option valuation are theoretical, resulting in the formulation of analytical closed forms for some of these option types. In contrast, others rely heavily on numerical approximation techniques, such as Monte Carlo simulations, finite difference methods, finite volume methods, binomial tree methods, etc. These theoretical formulas are mainly based on assumptions about the behavior of the underlying prices of securities, constant risk-free interest rates, constant volatility, etc., and they have been duly criticized over the years. However, modifications have been made to the Black–Scholes model, thereby giving rise to such models as the mixed diffusion/pure jump models, displaced diffusion models, stochastic volatility models, constant elasticity of variance diffusion models, etc. On the other hand, neural networks (NNs) have proved to be emerging computing techniques that offer a modern avenue to explore the dynamics of financial applications, such as derivative pricing [8].

Recent years have seen a huge application of AI and ML, as they have been utilized greatly in diverse financial fields. They have contributed significantly to financial institutions, the financial market, and financial supervision. Li in [9] summarized the AI and ML development and analyzed their impact on financial stability and the micro-macro economy. In finance, AI has been utilized greatly in predicting future stock prices, and the concept lies in building AI models which utilize ML techniques, such as reinforcement learning or neural networks [10]. A similar stock price prediction was conducted by Yu and Yan [11]; they used the phase-space reconstruction method for time series analysis in combination with a deep NN long- and short-term memory networks model. Regarding applying neural networks to option pricing, one of the earliest research can be found in Malliaris and Salchenberger [12]. They compared the performance of the ANN in pricing the American-style OEX options (that is, options defined on Standard and Poor's (S&P) 100) and the results from the Black–Scholes model [13] with the actual option prices listed in the *Wall Street Journal* . Their results showed that in-the-money call options were valued significantly better when the Black–Scholes model was used, whereas the ANN techniques favored the out-of-the-money call option prices.

In pricing and hedging financial derivatives, researchers have incorporated the classical Black–Scholes model [13] into ML to ensure robust and more accurate pricing techniques. Klibanov et al. [14] used the method of quasi-reversibility and ML to predict option prices in corporations with the Black–Scholes model. Fang and George [15] proposed valuation techniques for improving the accuracy rate of Asian options by using the NN in connection with Levy approximation. Hutchinson et al. in [16] further priced the American call options defined on S&P 500 futures by comparing three ANN techniques with the Black–Scholes pricing model. Their results proved the supremacy of all three ANNs to the classical Black–Scholes model. Other comparative research studies on the ANN versus the Black–Scholes model are also applicable in pricing the following: European-style call options (with dividends) on the Financial Times Stock Exchange (FTSE) 100 index [17], American-style call options on Nikkei 225 futures [8], Apple's European call options [18], S&P 500 index call options with an addition of neuro-fuzzy networks [19], and in the pricing call options written on the Deutscher Aktienindex (DAX) German stock index [20]. Similar works on pricing and hedging options using the ML techniques can be found in [21–25].

Other numerical techniques, such as the PDE-based and the DeepBSDE-based (BSDE—-backward stochastic differential equations) methods, have also been employed in valuing the barrier options. For instance, Le et al. in [26] solved the corresponding option pricing PDE using the continuous Fourier sine transform and extended the concept of pricing the rebate barrier options. Umeorah and Mashele [27] employed the Crank–Nicolson finite difference method in solving the extended Black–Scholes PDE, describing the rebate barrier options and pricing the contracts. The DeepBSDE concept initially proposed by Han et al. in [28] converted high-dimensional PDE into BSDE, intending to reduce the dimensionality constraint, and they redesigned the solution of the PDE problem as a deep-learning problem. Further implementation of the BSDE-based using the numerical method with deep-learning techniques in the valuation of the barrier options is found in [29,30].

Generally, the concept of ANN can be classified into three phases: the neurons, the layers and the whole architecture. The neuron, which is the fundamental core processing unit, consists of three basic operations: summation of the weighted inputs, the addition of a bias to the input sum, and the computation of the output value via an activation function. This activation function is used after the weighted linear combination and implemented at the end of each neuron to ensure the non-linearity effect. The layers consist of an input layer, a (some) hidden layer(s) and an output layer. Several neurons define each layer, and stacking up various layers constitutes the entire ANN architecture. As the data transmission signals pass from the input layer to the output layer through the middle layers, the ANN serves as a mapping function among the input–output pairs [2]. After training the ANN in options pricing, computing the in-sample and out-of-sample options based on ANN becomes straightforward and fast [31]. Itkin [31] highlighted this example by pricing and calibrating the European call options using the Black–Scholes model.

This research is an intersection of machine learning, statistics and mathematical finance, as it employs recent financial technology in predicting option prices. To the best of our knowledge, this ML approach to pricing the rebate and zero-rebate barrier options has received less attention. Therefore, we aim to fill the niche by introducing this option pricing concept to exotic options. In the experimental section of this work, we simulate the barrier options dataset using the analytical form of the extended Black–Scholes pricing model. This is a major limitation of this research, and the choice was due to the non-availability of the real data. (A similar synthetic dataset was equally used by [32], in which they constructed the barrier option data based on the LIFFE standard European option price data by the implementation of the Rubenstein and Reiner analytic model. These datasets were used in the pricing of the up-and-out barrier call options via the use of a neural net model.) We further show and explain how the fully connected feed-forward neural networks can be applied in the fast and robust pricing of derivatives. We tuned different hyperparameters and used the optimal in the modeling and training of the NN. The performance of the optimal NN results is compared by benchmarking the results against other ML models,

such as the random forest regression model and the polynomial regression model. Finally, we show how the barrier options and their Greeks can be trained and valued accurately under the extended Black–Scholes model. The major contributions of this research are classified as follows:

- We propose a non-parametric technique of barrier option valuation and hedging using the concept of a fully connected feed-forward NN.
- Using different evaluation metrics, we measure the performance of the NN algorithm and propose the optimal NN architecture, which prices the barrier options effectively in connection to some specified data-splitting techniques.
- We prove the accuracy and performance of the optimal NN model when compared to those produced by other ML models, such as the random forest and the polynomial regression, and extract the barrier option prices and their corresponding Greeks with high accuracy using the optimal hyperparameter.

The format of this paper is presented as follows: In Section 1, we provide a brief introduction to the topic and outline some of the related studies on the applications of ANN in finance. Section 2 introduces the concept of the Black–Scholes pricing model, together with the extended Black–Scholes pricing models for barrier options and their closed-form Greeks. Section 3 focuses on the machine learning models, such as the ANN, as well as its applications in finance, random forest regression and the polynomial regression models. In Section 4, we discuss the relevant results obtained in the course of the numerical experiments, and Section 5 concludes our research study with some recommendations.

2. Extended Black–Scholes Model for Barrier Options

The classical Black–Scholes model developed by Fischer Black and Myron Scholes is an arbitrage-free mathematical pricing model used to estimate the dynamics of financial derivative instruments. The model was initially designed to capture the price estimate of the European-style options defined under the risk-neutral measure. As a mathematical model, certain assumptions, such as the log-normality of underlying prices, constant volatility, frictionless market, continuous trading without dividends applied to stocks, etc., are made for the Black–Scholes model to hold [13]. Though the Black–Scholes model has been criticized over the years due to some underlying assumptions, which are not applicable in the real-world scenario, certain recent works are associated with the model [33–36]. Additionally, Eskiizmirliler et al. [37] numerically solved the Black–Scholes equation for the European call options using feed-forward neural networks. In their approach, they constructed a function dependent on a neural network solution, which satisfied the given boundary conditions of the Black–Scholes equation. Chen et al. [38] proposed a Laguerre neural network to solve the generalized Black–Scholes PDE numerically. They experimented with this technique on the European options and generalized option pricing models.

On the other hand, the valuation of exotic derivatives, such as the barrier options, has been extensively studied by many authors, mainly by imploring a series of numerical approximation techniques. Barrier options are typically priced using the Monte-Carlo simulations since their payoffs depend on whether the underlying price has/has not crossed the specified barrier level. The closed-form solutions can equally be obtained analytically using the extended Black–Scholes models [39], which shall be implemented as a benchmark of the exact price in this work. The structure of the model is described below.

2.1. Model Structure

Generally, the Black–Scholes option pricing formula models the dynamics of an underlying asset price S as a continuous time diffusion process given below:

$$\mathrm{d}S(t) = S(r\mathrm{d}t + \sigma \mathrm{d}B(t)), \tag{1}$$

where r is the risk-free interest rate, σ, the volatility and $B(t)$ is the standard Brownian motion at the current time t. Suppose $V(S,t)$ is the value of a given non-dividend paying

European call option. Then, under the pricing framework of Black and Scholes, $V(S,t)$ satisfies the following PDE:

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

subject to the following boundary and terminal conditions:

$$V(0,t) = 0\,, \forall\ t \in [0,T] \tag{3}$$

$$V(S,t) = S - Ke^{-r(T-t)} \ \text{ for } S \to \infty\,, \tag{4}$$

$$V(S,T) = \max\{S(T) - K, 0\}\,, \tag{5}$$

where K is the strike price and T is the time to expiration.

Since the barrier options are the focus of this study, the domain of the PDE in Equation (2) reduces to $\mathcal{D} = \{(S,t) : B \le S \le \infty; t \in [0,T]\}$ with the introduction of a predetermined level known as barrier B, and that feature distinguishes them from the vanilla European options. The boundary and terminal conditions above remain the same, with the exception of Equation (3), which reduces to $V(B,t) = 0$ for zero-rebate and $V(B,t) = R$ for the rebate barrier option. (In this paper, we shall consider the rebate paid at knock-out. The other type is the rebate paid at expiry, and in that case, Equation (3) becomes $V(B,t) = Re^{-r(T-t)}\,, \forall\ t \in [0,T]$.) The barrier options are either activated (knock-in options) or extinguished (knock-out options) once the underlying price attains the barrier level. The direction of the knock-in or the knock-out also determines the type of barrier options being considered, as this option is generally classified into up-and-in, up-and-out, down-and-in, and down-and-out barrier options. This paper will consider the down-and-out (DO) barrier options, both with and without rebates. For this option style, the barrier level is normally positioned below the underlying, and when the underlying moves in such a way that the barrier is triggered, the option becomes void and nullified (zero rebate). However, when the barrier is triggered, and the option knocks out with a specified payment compensation made to the option buyer by the seller, then we have the rebate barrier options. Under the risk-neutral pricing measure Q, the price of the down-and-out (DO) barrier options is given as

$$V(S,t) = \mathbb{E}^Q\left[e^{-r(T-t)}(S_T - K)^+ \mathbb{I}\{\min_{0\le t\le T} S_t > B\}\right], \tag{6}$$

and the solution to the above is given in the following theorem.

Theorem 1. *Extended Black–Scholes for a DO call option (note that Equations (7) and (8) occurs when the strike price $K \ge B$. For $K < B$, we substitute $K = B$ into d_1 and d_3.) is given by [39]*

$$V(S,t) = SN(d_1) - Ke^{-r\tau}N(d_2) - \left[S\left(\frac{B}{S}\right)^{2\eta} N(d_3) - Ke^{-r\tau}\left(\frac{B}{S}\right)^{2\eta-2} N(d_4)\right] \tag{7}$$

$$\text{for } d_1 = \frac{\log\left(\frac{S}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)\tau}{\sigma\sqrt{\tau}}, \quad d_3 = \frac{\log\left(\frac{B^2}{SK}\right) + \left(r + \frac{\sigma^2}{2}\right)\tau}{\sigma\sqrt{\tau}}, \quad d_5 = \frac{\log\left(\frac{B}{S}\right) + \left(r + \frac{\sigma^2}{2}\right)\tau}{\sigma\sqrt{\tau}}\,,$$

where $\tau = T - t$, $d_{2,4} = d_{1,3} - \sigma\sqrt{\tau}$, $\eta = (2r + \sigma^2)(2\sigma^2)^{-1}$ and $N(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{\frac{-y^2}{2}} dy$ is the cumulative standard normal distribution function. In the presence of a rebate R, the option value becomes

$$V_R(S,t) = V(S,t) + R\left[\left(\frac{B}{S}\right)^{2\eta-1} N(d_5) + \left(\frac{S}{B}\right) N(d_5 - 2\eta\sigma\sqrt{\tau})\right] \tag{8}$$

2.2. Option Greeks

These refer to the sensitivities of option prices with respect to different pricing parameters. The knowledge and the application of option Greeks can equip investors with risk-minimization strategies, which will be applicable to their portfolios. Such knowledge is as vital as hedging the portfolio risk using any other risk management tools. For options that have an analytical form based on the Black–Scholes model or other closed-form models, the Greeks or the sensitivities are normally estimated from these formulas. In the absence of analytical option values, numerical techniques are employed to extract the Greeks. These Greeks are adopted from [40], and we only consider the delta (Δ_{DO}), gamma (Γ_{DO}) and the vega (ν_{DO}).

2.2.1. Delta

This measures the sensitivity of options values to changes in the underlying prices. The delta for the DO call options behaves like the delta of the European call options when the option is deep in-the-money, and it becomes very complicated as the underlying price approaches the barrier level:

$$\frac{\partial V(S,t)}{\partial S} = N(d_1) - \left(\frac{B}{S}\right)^{2\eta-2}\left\{-\frac{B^2}{S^2}N(d_3) + \frac{2\eta-2}{S}\left(\frac{B^2}{S}N(d_4) - Ke^{-r\tau}N(d_3)\right)\right\},$$

where d_1, d_3 and d_4 are given in Theorem 1.

2.2.2. Gamma

This measures the sensitivity of delta to a change in the underlying price, or the second partial derivative of the option value with respect to the underlying price:

$$\begin{aligned}\frac{\partial^2 V(S,t)}{\partial S^2} = = \frac{\phi(d_1)}{\sigma S\sqrt{\tau}} - \left(\frac{B}{S}\right)^{2\eta-2}\Bigg\{&\frac{(2\eta-2)(8\eta-7)}{S}\left(\frac{B^2}{S}N(d_4) - Ke^{-r\tau}N(d_3)\right)\\ &+ \frac{B^2}{S^2}\left(2N(d_3) + \frac{\phi(d_3)}{\sigma\sqrt{\tau}}\right) + 2(2\eta-2)\left(\frac{B^2}{S^2}N(d_3)\right)\Bigg\},\end{aligned}$$

where d_1, d_3 and d_4 are given in Theorem 1; also, $\phi(x) = \frac{1}{\sqrt{2\pi}}e^{\frac{-x^2}{2}}$ is the probability density function of the standard normal distribution.

2.2.3. Vega

This measures the sensitivity of options values to changes to volatility. It is calculated as

$$\frac{\partial V(S,t)}{\partial \sigma} = S\sqrt{\tau}\phi(d_1) - \left(\frac{B}{S}\right)^{2\eta-2}\left\{\sqrt{\tau}Ke^{-r\tau}\phi(d_4) - \frac{4r}{\sigma^3}\left(\frac{B^2}{S}N(d_4) - Ke^{-r\tau}N(d_3)\right)\ln\frac{B}{S}\right\}.$$

where d_1, d_3 and d_4 are given in Theorem 1; also, $\phi(x) = \frac{1}{\sqrt{2\pi}}e^{\frac{-x^2}{2}}$ is the probability density function of the standard normal distribution.

3. Machine Learning Models

Machine learning models, such as the ANN, polynomial regression model and random forest regression models, form the methodology in this research. Here, we briefly describe each of them and their financial application as they relate to the rebate barrier options problem. The numerical experiments are performed on an 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz processor, 16 GB RAM, 64-bit Windows 10 operating system and x64-based processor.

3.1. Artificial Neural Networks

This subsection utilizes the concept of ANN in the approximation of functions which describe the financial model in perspective. It will highlight the whole network environment and the multi-layer perceptron (MLP) idea. In connection with the application of ANN to option pricing, the concept lies first in the generation of the financial data (barrier option pricing data) and then employing the ANN to predict the option prices according to the trained model.

3.1.1. Network Environment

The research computations and the data processing are implemented using `Python` (version 3.9.7), which is an open-source programming tool. The ANN employed in the data analysis and construction of the model, as well as the training and validation, is implemented with `Keras` (https://keras.io/about/, accessed on 6 April 2023), which is a deep learning application programming interface, running concurrently with the machine learning platform `Tensorflow` (version 2.2.0).

3.1.2. Multi-Layer Perceptron

An MLP is a feed-forward ANN category comprising a minimum of three layers: the input layer, the hidden layer and the output layer. The MLP with as little as one hidden layer tends to approximate a large category of non-linear and linear functions with arbitrary accuracy and precision. Except for input nodes, every other node consists of neurons triggered by non-linear activation functions. During the training phase, an MLP employs the supervised learning techniques, also known as backpropagation, and in this section, we use the backpropagation network method, which is by far the most widespread neural network type.

Mathematically, consider an MLP network's configuration with first and second hidden layers $h_k^{(1)}$ and $h_k^{(2)}$, respectively, and input units x_k, where k denotes the number of the units. The non-linear activation function is written as $f(.)$, and we denote $f^{(1)}(.)$, $f^{(2)}(.)$ and $f^{(3)}(.)$ differently since the network layers can have various activation functions, such as the sigmoid (Sigmoid is defined by $f(z) = 1/(1 + \exp(-z))$, where z is the input to the neuron), hyperbolic tangent (Tanh is defined by $f(z) = 2\text{sigmoid}(z) - 1$, where z is the input to the neuron), rectified linear unit (ReLU) (ReLU is defined by $f(z) = \max[0, z]$, where z is the input to the neuron), etc. The weights of the network are denoted by w_{jk}, the activation output value y_j, and the bias b_j, where j denotes the number of units in each layer. Thus, we have the following representation:

$$
\begin{aligned}
h_j^{(1)} &= f^{(1)}\left(\sum_k w_{jk}^{(1)} x_k + b_j^{(1)}\right) \\
h_j^{(2)} &= f^{(2)}\left(\sum_k w_{jk}^{(2)} h_k^{(1)} + b_j^{(2)}\right) \\
y_j &= f^{(3)}\left(\sum_k w_{jk}^{(3)} h_k^{(2)} + b_j^{(3)}\right).
\end{aligned}
$$

3.1.3. The Hyperparameter Search Space and Algorithm

This section further explains the hyperparameter optimization techniques, which aim to search for the optimal algorithm needed for our optimization problem. It is essential to note that the parameters of the NN are internal configuration variables that the models can learn. Examples are the weights and the bias. In contrast, the hyperparameters are external and cannot be learned from the data but are used to control the learning process and the structure of the NN. These parameters are set before the training process, and some examples include the activation function, batch size, epoch, learning rates, etc. The choice of hyperparameters hugely affects the accuracy of the network. As a

result, different optimal methods, such as manual search, Bayesian optimization, random search and grid search, have been developed. We will employ the Keras tuner framework (https://keras.io/keras_tuner/, accessed on 6 April 2023), which encompasses some algorithms, such as the random search, hyperband and Bayesian optimization. For these three search algorithms, we choose the validation loss as the objective with the maximum search configuration of six trials. The following variables define the search space of our NN architecture:

(1) Width and depth: The depth refers to the number of hidden layers, and the width is the number of units in each hidden layer. Thus, the depth ranges from [1, 4] with a step of 1, and the width ranges from [32, 512] with a step of 32.
(2) Activation function: These are non-linear activation functions in the NN, and we only consider Sigmoid, ReLU and Tanh .
(3) Optimizer: This modifies the model parameters and is mostly used for weight adjustments to minimize the loss function. We only consider Adam, SGD, Adagrad and RMSprop.
(4) Learning and dropout rates: Learning rates control the speed at which the NN learns, and we set it at [0.01, 0.001, 0.0001]. The dropout is a regularization technique employed to improve the ability of NN to withstand overfitting. We set it at [0.1, 0.5] with a step size of 0.1.

The activation functions are used in each layer, except the output layer. The network is trained with 45 epochs, 256 batch sizes and an early stopping callback on the validation loss with `patience = 3`. Since the option pricing model is a regression problem, our primary objective is to keep the mean squared error (MSE) of the predicted prices to a minimum. The essence of training a neural network entails minimizing the errors obtained during the regression analysis, and this is done by selecting a set of weights in both the hidden and the output nodes. Thus, to evaluate the performance of our ANN, we consider the MSE as the loss function used by the network and the mean absolute error (MAE) as the network metrics, which are given, respectively, as follows:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(V_i(S,t) - \hat{V}_i(S,t))^2$$
$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|V_i(S,t) - \hat{V}_i(S,t)|,$$

where N is the number of observations, $V_i(S,t)$ is the exact option values and $\hat{V}_i(S,t)$ is the predicted option values. Finally, we alternate the activation functions, optimizers, batch normalization and dropout rates to investigate the effect of the network training on the option valuation and avoid overfitting the models.

3.1.4. Data Splitting Techniques for the ANN

Data splitting is a fundamental aspect of data science, especially for developing data-based models. The dataset is divided into training and testing sets, and an additional set known as the validation can also be created. The training set is used mainly for training, and the model is expected to learn from this dataset while optimizing any of its parameters. The testing set contains the data which are used to fit and measure the model's performance. The validation set is mainly used for model evaluation. If the difference between the training set error and the validation set error is large, there is a case of over-fitting, as the model has high variance. This paper considers supervised learning in which the model is trained to predict the outputs of an unspecified target function. This function is denoted by a finite training set $\mathcal{F}$ consisting of inputs and corresponding desired outputs: $\mathcal{F} = \{[\overrightarrow{a_1}, \overrightarrow{x_1}], [\overrightarrow{a_2}, \overrightarrow{x_2}], \cdots [\overrightarrow{a_n}, \overrightarrow{x_n}]\}$, where n is the number of 2-tuples of input/output samples.

Train–Test Split

This paper considers the train–test split as 80:20 and a further 80:20 on the new train data to account for a validation dataset. Thus, 80% of the whole dataset will account for the training set and 20% for the test dataset. Additionally, 80% of the training set will be used as the actual training dataset and the remaining 20% for validation. After training, the final model should correctly predict the outputs and generalize the unseen data. Failure to accomplish this leads to over-training, and these two crucial conflicting demands between accuracy and complexity are known as the bias–variance trade-off [41,42]. A common approach to balance this trade-off is to use the cross-validation technique.

K-Fold Cross Validation

The *k*-fold cv is a strategy for partitioning data with the intent of constructing a more generalized model and estimating the model performance on unseen data. Denote the validation (testing) set as $\mathcal{F}_{te}$ and the training set as $\mathcal{F}_{tr}$. The algorithm (Algorithm 1) is shown below.

Algorithm 1 Pseudocode for the *k*-fold cross validation

Input the dataset $\mathcal{F}$, number of folds k and the error function (MSE)

1: **Data split** :
   - Randomly split $\mathcal{F}$ into k independent subsets $\mathcal{F}_i, \mathcal{F}_2, \cdots, \mathcal{F}_k$ of same size.
   - For $i = 1, 2, \cdots, k$: $\mathcal{F}_{te} \leftarrow \mathcal{F}_i$ and $\mathcal{F}_{tr} \leftarrow \mathcal{F} \setminus \{\mathcal{F}_i\}$.
2: **Fitting and Training**:
   - Fit and train model on $\mathcal{F}_{tr}$ and evaluate model performance using $\mathcal{F}_{te}$ periodically: $\mathcal{R}_{te}(i)$=Error ($\mathcal{F}_{te}$).
   - Terminate model training when the $\mathcal{R}_{te}(i)$ stop criterion is satisfied.
3: **Evaluation**:
   - Evaluate the model performance using $\mathcal{R}_{te} = \frac{1}{k}\sum_{i=1}^{k}\mathcal{R}_{te}(i)$.

3.1.5. Architecture of ANN

This research considers a fully connected MLP NN in the option valuation for this research, which will consist of eight input nodes (in connection to the extended Black–Scholes for the rebate option parameters). There will be one output node (option value); the hidden layers and nodes will be tuned. There are two main models classified under the data-splitting techniques: Model A (train–test split) and Model B (5-fold cross-validation split). Each model is further subdivided into 3 according to the hyperparameter search algorithm. Thus, Models A1, A2, and A3 represent the models from the data train–test split for the hyperband algorithm, random search algorithm and Bayesian optimization algorithm, respectively. Additionally, Models B1, B2, and B3 represent the models from the k-fold cross-validation data split for the hyperband algorithm, random search algorithm and Bayesian optimization algorithm, respectively. Finally, Tables 1 and 2 present the post-tuning search details and the optimal model hyperparameters for the NN architecture, respectively.

Table 1. Trainable parameter search details.

Search Details	Model A1	Model A2	Model A3	Model B1	Model B2	Model B3
Best MAE score	0.07081	0.02794	0.01602	0.10141	0.03391	0.02911
Trainable parameters	254,305	218,145	113,345	138,945	80,961	4609
Total search time (secs)	620	714	642	642	491	754

Table 1 compares the search time taken by each of the algorithms in tuning the hyperparameters. We observed that the hyperband algorithm is highly efficient regarding the search time for the train–test split and the *k*-fold cross-validation models. The hyperband algorithm search time is generally less when compared to the random search and the Bayesian optimization algorithm. Furthermore, the Bayesian optimization provided the lowest MAE score and required fewer trainable parameters than the hyperband and the random search algorithm. This characteristic is equally observable for both models A and B. From the tuning, we can see that the Bayesian optimization effectively optimizes the hyperparameter when producing the lowest MAE, though it had the disadvantage of a higher search time. In contrast to the Bayesian optimization, the hyperband algorithm is optimal in terms of search time, despite having a higher MAE score. From the results section, the final comparison of optimality will be made in terms of the deviation from the actual values when all the models are used in the pricing process.

Table 2. Architecture of the ANN.

Hyperparameters	Model A1	Model A2	Model A3	Model B1	Model B2	Model B3
Activation Fn	Sigmoid	Tanh	Tanh	ReLU	Tanh	ReLU
Optimizer	Adam	Rmsprop	Adam	Adagrad	Rmsprop	Adam
Learning rate	0.001	0.0001	0.0001	0.1	0.0001	0.0001
Hidden layers	4	3	4	4	2	3
Layer 1 (dropout)	512(0.2)	288(0.2)	32(0.1)	384(0.3)	480(0.2)	512(0.1)
Layer 2 (dropout)	192(0.2)	480(0.3)	32(0.1)	352(0.2)	160(0.3)	32(0.1)
Layer 3 (dropout)	224(0.4)	160(0.5)	512(0.2)	448(0.4)	Nil	352(0.1)
Layer 4 (dropout)	480(0.5)	Nil	224(0.2)	192(0.4)	Nil	Nil

3.2. Random Forest Regression

Random forest combines tree predictors in such a way that each tree in the ensemble is contingent on the values of a randomly sampled vector selected from the training set. This sampled vector is independent and has similar distribution with all the trees in the forest [43]. The random forest regressor uses averaging to improve its predictive ability and accuracy.

Let $f(\mathbf{x};\beta_n)$ be the collection of tree predictors where $n = 1, 2, \cdots, N$ denotes the number of trees. Here, $\mathbf{x}$ is the observed input vector from the random vector $\mathbf{X}$, and β_n are the independent and identically distributed random vectors. The random forest prediction is given by

$$\bar{f}(\mathbf{x}) = \frac{1}{N} f(\mathbf{x};\beta_n)\,,$$

where $\bar{f}(\mathbf{x})$ is the unweighted average over the tree collection $f(\mathbf{x})$. As the number of trees increases, the tree structure converges. This convergence explains why the random forest does not overfit, but instead, a limiting value of the generalization (or prediction) error is produced [43,44]. Thus, we have that as $n \to \infty$, the law of large numbers ensures that

$$\mathbb{E}_{\mathbf{X},Y}[Y - \bar{f}(\mathbf{X})]^2 \to \mathbb{E}_{\mathbf{X},Y}[Y - \mathbb{E}_{\beta}[\bar{f}(\mathbf{X};\beta)]]^2$$

Here, Y is the outcome. The training data are assumed to be drawn independently from the joint distribution of $(\mathbf{X}, Y)$. In this research, we use the 80:20 train–test split techniques to divide the whole dataset into a training set and a testing set. Using the `RandomForestRegressor()` from the scikit-learn ML library, we initialize the regression model, fit the model, and predict the target values.

3.3. Polynomial Regression

Polynomial regression is a specific type of linear regression model which can predict the relationship between the independent variable to the dependent variable as an nth degree polynomial. In this research, we first create the polynomial features object using the `PolynomialFeatures()` from the scikit-learn ML library and indicate the preferred polynomial degree. We next use the 80:20 train–test split techniques to divide this new polynomial feature into training and testing datasets, respectively. Next, we construct the polynomial regression model, fit the model and predict the responses.

4. Results and Discussion

4.1. Data Structure and Description

For the ANN model input parameters, we generated 100000 sample data points and then used Equation (8) to obtain the exact price for the rebate barrier call options. These random observations will train, test and validate an ANN model to mimic the extended Black–Scholes equation. We consider the train–test split and the cross-validation split on the dataset and then measure these impacts on the loss function minimization and the option values. The generated samples consist of eight variables, that is $(S, K, B, R, T, \sigma, r, V_R)$, which are sampled uniformly, except the option price V_R, and following the specifications and logical ranges of each of the input variables (See Table 3). During the training process, we fed the ANN the training samples with the following inputs $(S, K, B, R, T, \sigma, r)$, where V_R is the expected output. In this phase, the ANN 'learns' the extended Black–Scholes model from the generated dataset, and the testing phase follows suit, from which the required results are predicted. Meanwhile, under the Black–Scholes framework, we assume that the stock prices follow a geometric Brownian motion, and we used $\texttt{GBM}(x = 150, r = 0.04, \sigma = 0.5, T = 1, N = 100{,}000)$ for the random simulation. Table 3 below shows the extended Black–Scholes parameters used to generate the data points, whereas Table 4 gives the sample data generated. The range for the rebate, strike and barrier is from the uniform random distribution, and they are multiplied by the simulated stock price to obtain the final range.

Table 3. Extended Black–Scholes range of parameters—rebate barrier.

Strike	Barrier	Rebate	Time	Volatility	Rate
[0.4, 1]	[0.4, 1]	[0.01, 0.05]	[0.5, 1.5]	[0.1, 0.5]	[0.01, 0.05]

Table 4. Sample training data for rebate barrier option pricing model.

Stock	Barrier	Strike	Rebate	Rate	Volatility	Time	Call Option
98.25745	51.92557	46.33445	0.01835	0.04355	0.22836	1.42274	54.61598
149.79728	96.91339	105.83799	0.04255	0.02863	0.27321	0.85166	47.22227
55.90715	38.39396	35.25565	0.01241	0.01989	0.34164	0.95034	20.32389
63.29343	41.03505	31.03422	0.04143	0.03072	0.21341	0.95572	32.80836
126.83153	116.65153	124.11145	0.02695	0.01888	0.45170	0.74936	9.60775
97.18410	75.27999	92.85564	0.01626	0.02186	0.39194	1.04861	15.67665
112.31872	112.30254	53.26221	0.01876	0.03539	0.43021	1.05975	0.04930

Statistics and Exploratory Data Analysis

In this section, we aim to summarize the core characteristics of our option dataset by analyzing and visualizing them. The descriptive statistics which summarize the distribution shape, dispersion and central tendency of the dataset are presented in Table 5. The following outputs were obtained: the number of observations or elements, mean, standard deviation, minimum, maximum and quartiles (25%, 50%, 75%) of the dataset. We observed that the distribution of the simulated stock is left skewed since the mean is lesser than the median, whereas the distributions of the option values, strike price and barrier levels are right skewed.

Table 5. Descriptive statistics for the rebate barrier.

	Count	Mean	Std	Min	25%	50%	75%	Max
Stock	100,000.0	100.914912	25.412662	49.601351	89.084095	100.055323	110.997594	171.710128
Strike	100,000.0	70.644521	25.269816	20.236066	51.255058	68.083559	86.709133	170.197316
Rebate	100,000.0	0.029946	0.011529	0.010000	0.019957	0.029952	0.039881	0.049999
Barrier	100,000.0	70.574580	25.264409	20.380974	51.162052	67.929254	86.572706	169.477720
Time	100,000.0	1.000314	0.288309	0.500003	0.751330	0.999858	1.250020	1.499976
Sigma	100,000.0	0.299688	0.115622	0.100001	0.199680	0.299840	0.400159	0.499991
Rate	100,000.0	0.030006	0.011557	0.010001	0.019994	0.029978	0.040008	0.050000
OptionV	100,000.0	27.183098	17.482423	0.025235	13.547223	24.047318	38.324748	103.946198

In Figure 1, we consider the visualization using the `seaborn` library in connection with the `pairplot` function to plot a symmetric combination of two main figures, that is, the scatter plot and the kernel density estimate (KDE). The KDE plot is a non-parametric technique mainly used to visualize the nature of the probability density function of a continuous variable. In our case, we limit these KDE plots to the diagonals. We focus on the relationship between the stock, strike, rebate and the barrier with the extended Black–Scholes price (OptionV) for the rebate barrier options. From the data distribution for the feature columns, we notice that the sigma, time and rate columns could be ignored. This is because the density distribution shows that these features are basically uniform, and the absence of any variation makes it very unlikely to improve the model performance. Suppose we consider this problem as a classification problem; then, no split on these columns will increase the entropy of the model.

On the contrary, however, if this was a generative model, then there would be no prior to updating given a uniform posterior distribution. Additionally, the model will learn a variate of these parameters since, by definition of the exact option price (referred to as OptionV) function, these are the parameters which can take on constant values. Another method to consider would be to take these parameters 'sine' functions as inputs to the model instead of the actual values. We observed from our analysis that this concept works, but there is not a significant improvement in model performance, which can be investigated in further research.

4.2. Neural Network Training

The first category (train dataset) is employed to fit the ANN model by estimating the weights and the corresponding biases. The model at this stage tends to observe and 'learn' from the dataset to optimize the parameters. In contrast, the other (test dataset) is not used for training but for evaluating the model. This dataset category explains how effective and efficient the overall ANN model is and the prediction probability of the model. Next and prior to the model training, we perform data standardization techniques to improve the performance of the proposed NN algorithm. The `StandardScalar` function of the `Sklearn python` library was used to standardize the distribution of values by ensuring that the distribution has a unit variance and a zero mean. During the compilation stage, we plot the loss (MSE) and the evaluation metrics (accuracy) values for both the train and validation datasets. We equally observe that the error difference between the training and the validation dataset is not large, and as such, there is no case of over- or under-fitting of the ANN models. Once the 'learning' phase of the model is finished, the prediction phase will set in. The performance of the ANN model is measured and analyzed in terms of the MSE and the MAE. Table 6 gives the evaluation metrics for both the out-sample prediction (testing dataset) and the in-sample prediction (training dataset).

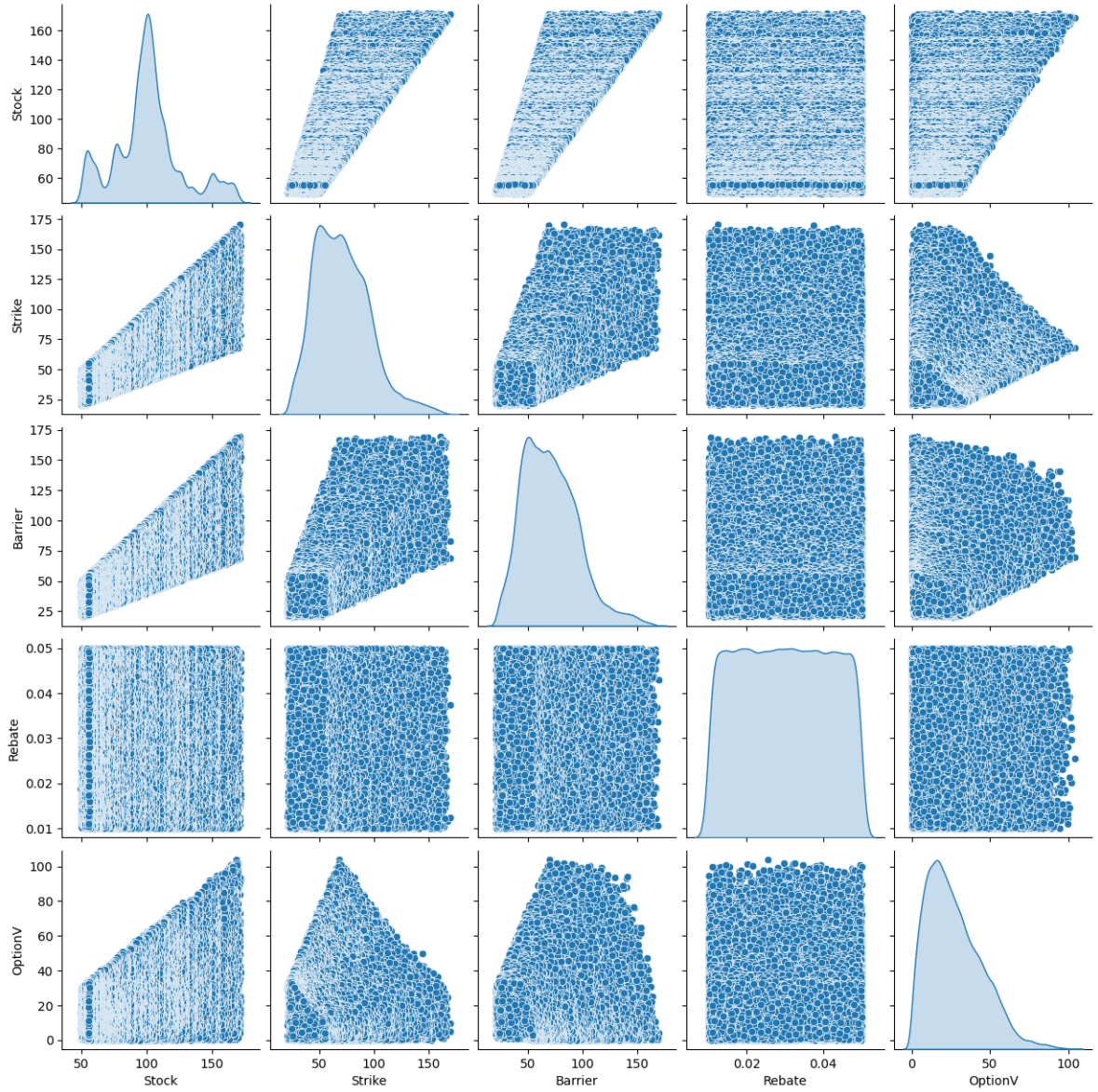


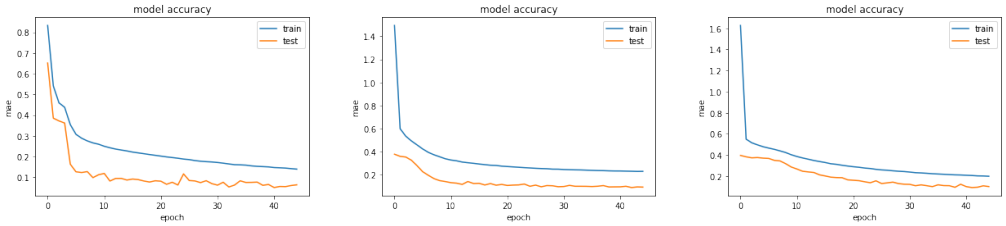
Figure 1. Visualization plot.

Table 6. Model evaluation for testing and training data (shows no over- or underfitting).

Models	Test		Train	
	Loss (MSE)	Metrics (MAE)	Loss (MSE)	Metrics (MAE)
Model A1	0.0214	0.0574	0.0249	0.0557
Model A2	0.0349	0.0987	0.0299	0.0929
Model A3	0.0446	0.1058	0.0423	0.1027
Model B1	0.0229	0.0601	0.0202	0.0584
Model B2	0.0425	0.0883	0.0394	0.0872
Model B3	0.0457	0.0782	0.0411	0.0762

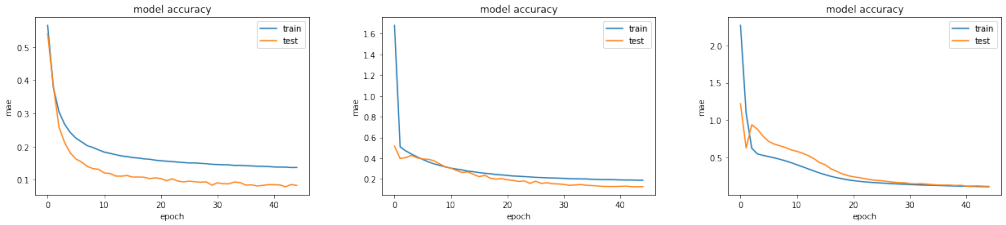
Table 6 shows the model evaluation comparison for the train/test loss and accuracy. It is observed that the test loss is greater than the training loss, and the test accuracy is greater than the training accuracy for all the models. The differences in error sizes are not significant, and thus the chances of having an overfitting model are limited. Figures 2–5 show the training and validation (test) of the loss and MAE values for all the models when the models are fitted and trained on epoch = 45, batch size = 256, and verbose = 1. We

visualize these graphs to ascertain whether there was any case of overfitting, underfitting or a perfect fitting of the model. In underfitting, the NN model fails to model the training data and learn the problem perfectly and sufficiently, leading to slightly poor performance on the training dataset and the holdout sample. Overfitting occurs mainly in complex models with diverse parameters, which happens when the model aims to capture all data points present in a specified dataset. In all the cases, we observe that the models show a good fit, as the training and validation loss are decreased to a stability point with an infinitesimal gap between the final loss values. However, the loss values for Model B3 followed by Model B2 are highly optimal in providing the best fit for the algorithm.



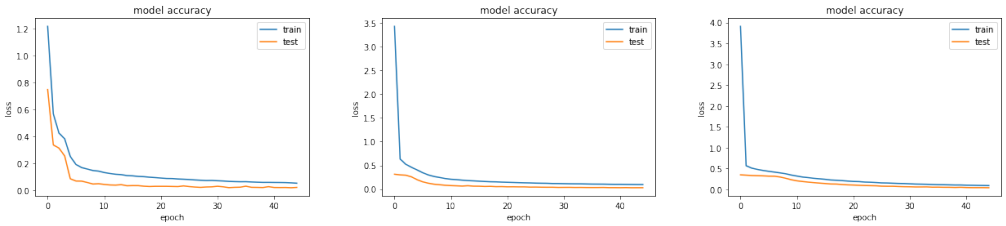
(**a**) (**b**) (**c**)

Figure 2. Train/test MAE values for Models A1, A2 and A3; (**a**) MAE—Model A1; (**b**) MAE—Model A2; (**c**) MAE—Model A3.



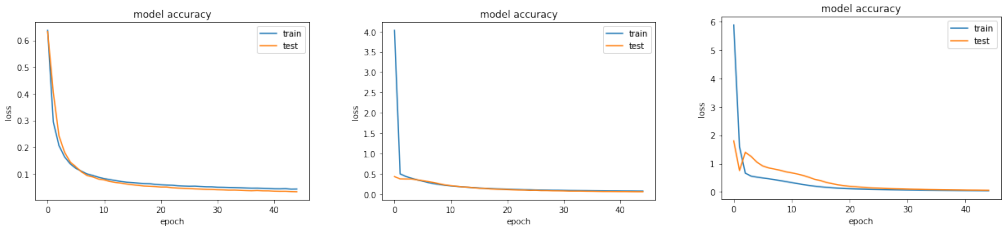
(**a**) (**b**) (**c**)

Figure 3. Train/Test MAE values for Models B1, B2 and B3; (**a**) MAE—Model B1; (**b**) MAE—Model B2; (**c**) MAE—Model B3.



(**a**) (**b**) (**c**)

Figure 4. Train/Test LOSS values for Models A1, A2 and A3; (**a**) LOSS—Model A1; (**b**) LOSS—Model A2; (**c**) MAE—Model A3.



(**a**) (**b**) (**c**)

Figure 5. Train/Test LOSS values for Models B1, B2 and B3; (**a**) LOSS—Model B1; (**b**) LOSS—Model B2; (**c**) LOSS—Model B3.

Next, we display the plots obtained after compiling and fitting the models. The prediction is performed on the unseen data or the test data using the trained models. Figures 6 and 7 give the plot of the predicted values against the actual values, the density plot of the error values and the box plot of the error values for all six models.

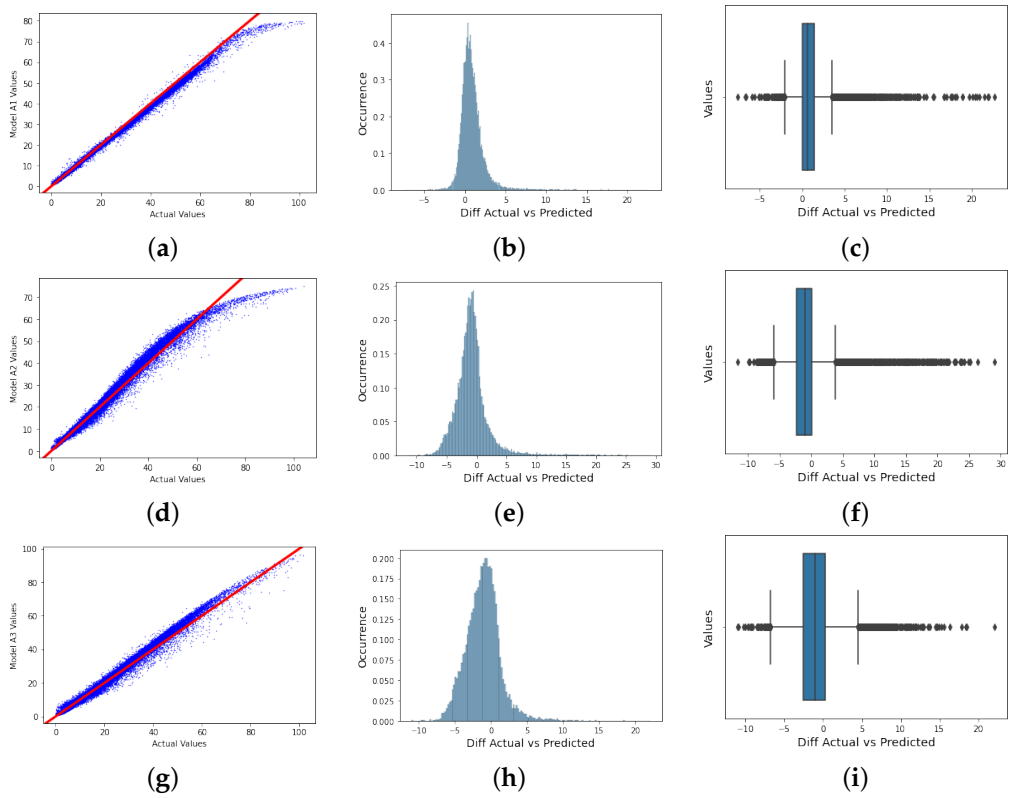


Figure 6. Option values visualization for Models A1, A2, A3; (**a**) Model A1: Regression plot; (**b**) Model A1: Histogram plot; (**c**) Model A1: Box plot; (**d**) Model A2: Regression plot; (**e**) Model A2: Histogram plot; (**f**) Model A2: Box plot; (**g**) Model A3: Regression plot; (**h**) Model A3: Histogram plot; (**i**) Model A3: Box plot.

The box plot enables visualization of the skewness and how dispersed the solution is. Model A2 behaved poorly, as this can be observed with the wide range of dispersion of the solution points, and the model did not fit properly. For a perfect fit, the data points are expected to concentrate along the 45 deg red line, where the predicted values are equal to the actual values. This explanation is applicable to Models A2 and A3, as there was no perfect alignment in the regression plots. We could retrain the neural network to improve this performance since each training can have different initial weights and biases. Further improvements can be made by increasing the number of hidden units or layers or using a larger training dataset. For the purpose of this research, we already performed the hyperparameter tuning, which solves most of the above suggestions. To this end, we focus on Model B, another training algorithm.

Models B3 and B1 provide a good fit when their performance is compared to the other models, though there are still some deviations around the regression line. The deviation of these solution data points is also fewer than in the other models. It is quite interesting to note that the solution data points of Models B1 and B3 are skewed to the left, as can be seen in the box plots. This could be a reason for their high performance compared to other models, such as A1, A2, and A3, which are positively skewed. However, this behavior would be worth investigating in our future work.

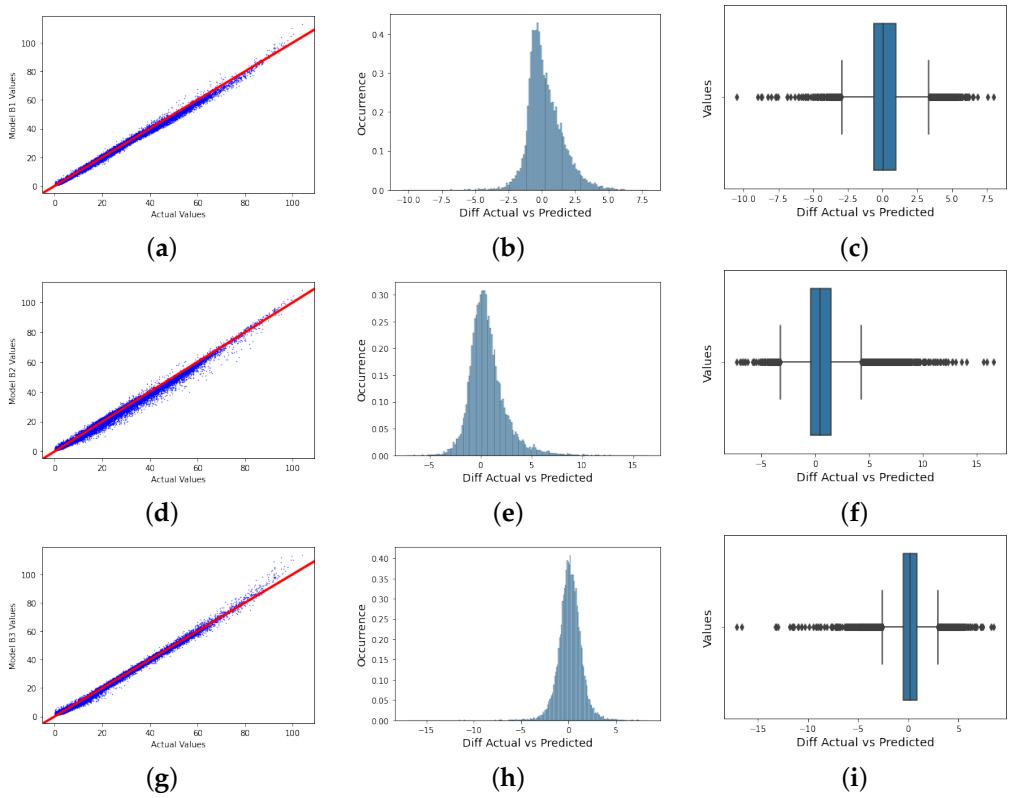


Figure 7. Option values visualization for Models B1, B2, B3; (**a**) Model B1: Regression plot; (**b**) Model B1: Histogram plot; (**c**) Model B1: Box plot; (**d**) Model B2: Regression plot; (**e**) Model B2: Histogram plot; (**f**) Model B2: Box plot; (**g**) Model B3: Regression plot; (**h**) Model B3: Histogram plot; (**i**) Model B3: Box plot.

Table 7 shows the error values in terms of the MSE, MAE, mean squared logarithmic error (MSLE), mean absolute percentage error (MAPE) and the R^2 (coefficient of determination) regression score. It also shows the models' comparison in terms of their computation speed, and it must be noted that the computation is measured in seconds. Mathematically, the MSLE and MAPE are given as

$$\text{MSLE} = \frac{1}{N}\sum_{i=1}^{N}[\log_e(1+V_i(S,t)) - \log_e(1+\hat{V}_i(S,t))]^2,$$

$$\text{MAPE} = \frac{100\%}{N}\sum_{i=1}^{N}\left|\frac{V_i(S,t)-\hat{V}_i(S,t)}{V_i(S,t)}\right|,$$

where N is the number of observations, $V_i(S,t)$ is the exact option values and $\hat{V}_i(S,t)$ is the predicted option values. For the MAPE, all the values lower than the threshold of 20% are considered 'good' in terms of their forecasting capacity [45]. Thus, all the models have good forecasting scores, with Model A1 possessing a highly accurate forecast ability. Similarly, the values for the MSLE measure the percentile difference between the log-transformed predicted and actual values. The lower, the better, and we can observe that all the models gave relatively low MSLE values, with Models A1 and B1 giving the lowest MSLE values.Please check that intended meaning is retained.

From Table 7, the R^2 measures the capacity of the model to predict an outcome in the linear regression format. Models B1 and B3 gave the highest positive values compared to the other models, and these high R^2-values indicate that these models are a good fit for our options data. It is also noted that for well-performing models, the greater the R^2, the smaller the MSE values. Model B3 gave the smallest MSE, with the highest R^2, compared to the least performed model A2, which had the largest MSE and the smallest R^2 score. The MAE

measures the average distance between the predicted and the actual data, and the lower values of the MAE indicate higher accuracy.

Table 7. Error values and computation time for various NN models.

Models	Error Values					Comp Time (secs)
	R^2-Score	MAE	MSLE	MSE	MAPE	
Model A1	0.989839	1.128247	0.006056	3.143079	0.076672	93.22
Model A2	0.969974	2.080077	0.013861	9.175201	0.129603	76.93
Model A3	0.977491	1.991815	0.019283	6.840152	0.143385	60.21
Model B1	0.994019	0.990397	0.006361	1.821165	0.080418	51.48
Model B2	0.987908	1.328247	0.015289	3.681967	0.125486	51.85
Model B3	0.994371	0.932689	0.014919	1.714182	0.125429	31.17

Finally, we display the speed of the NN algorithm models in terms of their computation times, as shown in Table 7. The computation time taken by the computer to execute the algorithm encompasses the data splitting stage, standardization of set variables, ANN model compilation and training, fitting, evaluation and the prediction of the results. As noted in Models A1 and A2, the use of Sigmoid and Tanh activation functions accounted for higher computation time, and this is due to the presence of exponential functions, which need to be computed. Model A1 was the least performed in terms of the computation time, and Model B3 was the best, accounting for a 66.56% decrease in time. We observe that the computation time is reduced when the k-fold cross-validation split is implemented prior to the ANN model training, as compared to the traditional train–test split. This feature is evident as a further 41.62% decrease was observed when the average computation time for Model B was used against Model A.

The overall comparison of the tuned models is presented in Figure 8. Here, we rank the performance of each MLP model with regards to ST:TP, algorithm computation time, and finally, the errors spanning from the R^2 score, MAE and the MSE. The ST:TP ratio denotes the search time per one trainable parameter. The ranking is ascending, with 1 being the maximum preference and 6 being the least preference. From the results and regardless of the number of search times per one trainable parameter, we observe that Model B3 is optimal, followed by Model B1, and the lowest performing is Model A2. Hence, we can conclude that models which consist of the k-fold data split performed significantly well in the valuation of the rebate barrier options using the extended Black–Scholes framework.

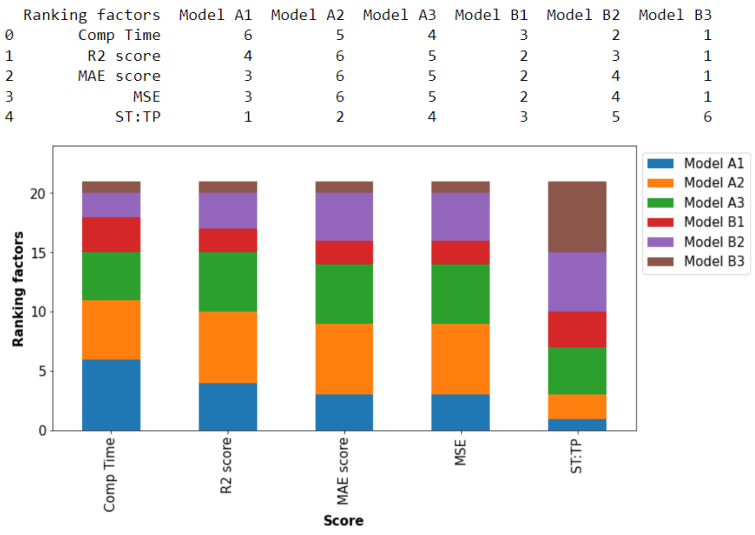


Figure 8. Ranking of models for optimality.

4.3. Analysis of Result Performance

One avenue to show the accuracy of our proposed model is to test the architecture on a non-simulated dataset for the rebate barrier options. At present, we are not able to obtain such real market data due to non-accessibility, and this is one of the limitations of the research. However, we compare the NN results with other machine learning models, such as the polynomial regression and the random forest regression on the same dataset. Both techniques are capable of capturing non-linear relationships that exist amongst variables.

Polynomial regression provides flexibility when modeling the non-linearity. Improved accuracy can be obtained when the higher-order polynomial terms are incorporated, and this feature makes it easier to capture the non-complex patterns in the dataset. It is also very fast when compared to both our proposed NN methodology and the random forest regression (Table 8). In this work, we only present the results obtained using the 2nd-, 3rd- and 4th-degree polynomial regressions. We observed that in terms of accuracy, polynomials of higher degrees gave rise to more accurate results and a significant reduction in their error components.

However, one of the issues facing the polynomial regression is model complexity; when the polynomial degree is high, the chances of model overfitting will be significantly high. Thus, we are faced with the trade-off between accuracy and over-fitting of the model. Regression random forest, on the other hand, combines multiple random decision trees, with each tree trained on a subset of data. We build random forest regression models using 10, 30, 50, and 70 decision trees, then we fit the model to the barrier options dataset, predict the target values, and then compute the error components. Finally, we compare these two models to the optimal model obtained with the NN results (Model B3), and we have the following table.

Table 8. Error values and computation time for Model B3, polynomial regression and random forest regression.

Models	**Error values**					**Time (secs)**
	R^2**-Score**	**MAE**	**MSLE**	**MSE**	**MAPE**	
Random Forest Regr.						
Decision trees (10)	0.990380	1.182121	0.016714	2.938727	0.145200	6.02
Decision trees (30	0.992352	1.056031	0.013461	2.293958	0.118368	15.56
Decision trees (50)	0.992449	1.037465	0.015827	2.299464	0.117032	26.34
Decision trees (70)	0.992825	1.022187	0.014146	2.211590	0.127853	36.02
Polynomial Regr.						
Polynomial order (2)	0.967764	2.156955	N/A	9.843225	0.327491	1.05
Polynomial order (3)	0.987900	1.269433	N/A	3.666175	0.206076	2.25
Polynomial order (4) [1]	0.996147	0.689323	N/A	1.177380	0.114092	3.75
Neural Network						
Model B3	0.994371	0.932689	0.014919	1.714182	0.125429	31.17

[1] We consider polynomials of order ≥ 4 to be higher-order, and this is because of the increase in their complexity. The accuracy of the 4th-order polynomial regression is actually higher than our proposed model, but the former has the issue of overfitting the data, which comes with a higher degree of polynomial regression. Additionally, the N/A in the MSLE cells is due to some negative values in the prediction set, which makes the logarithm of the values N/A.

Increasing the number of decision trees leads to more accurate results, and Oshiro et al. (2012) explained that the range of trees should be between 64 and 128. This feature will make it feasible to obtain a good balance between the computer's processing time, memory usage and the AUC (area under curve) [46]; we observed this feature in our research. The model was tested on 80, 100, 120, 140, 160, 180, and 200 decision trees, and we obtained the following coefficient of determination R^2 regression score (computation time): 0.9924 (34 secs), 0.9928 (52 secs), 0.9929 (62 secs), 0.9925 (75 secs), 0.9929 (83 secs), 0.9929 (89 secs) and 0.9926 (102 secs), respectively. We obtained the optimal decision tree to be between 110 and 120 with an R^2 score of 0.9929, and any other value below 110 will give rise to a less

accurate result. Any value above 120 will not lead to any significant performance gain but will only lead to more computational cost.

Table 8 compares the performance of our optimal NN model to the random forest and the polynomial regressions. The performance is measured based on the error values and the computational time. The NN model performed better than the random forest regression regardless of the number of decision trees used, and this was obvious from the results presented in Table 8 above. On the other hand, polynomial regression of the 2nd and 3rd orders underperformed when compared to the NN model, but maximum accuracy was obtained when higher order (≥ 4) was used. This higher order posed a lot of complexity issues, which our optimal NN model does not face. Hence, more theoretical understanding is needed to further explain the phenomenon, and this current research does not account for it.

4.4. Option Prices and Corresponding Greeks

To compute the zero-rebate DO option prices and their corresponding Greeks, we simulate another set of data (1,000,000) in accordance with the extended Black–Scholes model, and the Table 9 below gives a subset of the full dataset after cleansing.

Table 9. Data subset of option values and Greeks.

S	B	K	R	r	σ	T	$V(t,S)$	Δ_{DO}	Γ_{DO}	ν_{DO}
190.14286	80.0	100.0	0.0	0.05	0.25	1.0	95.04795	0.99811	0.00013	1.14425
83.61440	80.0	100.0	0.0	0.05	0.25	1.0	2.08913	0.48534	−0.00174	7.12088
108.12702	80.0	100.0	0.0	0.05	0.25	1.0	17.72886	0.74465	0.01029	30.44055
146.39879	80.0	100.0	0.0	0.05	0.25	1.0	51.77793	0.96780	0.00194	10.39359
121.23493	80.0	100.0	0.0	0.05	0.25	1.0	28.42419	0.86428	0.00678	24.95840

For the NN application, we used the hyperparameters of Model B3 to construct the NN architecture and train and predict the option values and their corresponding Greeks. The risks associated with the barrier options are complicated to manage and hedge due to their path-dependent exotic nature, which is more pronounced as the underlying approaches to the barrier level. For the Greeks considered here, we focus on predicting the delta, gamma and vega using the optimal NN model, and the following results were obtained.

Figure 9 shows the plot of the predicted and actual values of the DO option prices, together with the delta, gamma and vega values. For the option value, the DO call behaves like the European call when the option is far deep in-the-money, and this is because the impact of the barrier is not felt at that phase. The option value decreases and tends to zero as the underlying price approaches the barrier since the probability of the option being knocked out is very high. The in-the-money feature is equally reflected in the delta and gamma as they remain unchanged when the barrier is far away from the underlying. Here, the delta is one, and the gamma is zero.

Gammas for this option style are typically large when the underlying price is in the neighborhood of the strike price or even near the barrier, and it is the lowest for out-of-money options or knocked-out options. From Figure 9c, gamma tends to switch from positive to negative without switching from long to short options. The values of gammas are usually bigger than the gamma for the standard call option. These extra features pose a great challenge to risk managers during the rebalancing of portfolios. Lastly, vega measures the sensitivity of the option value with respect to the underlying volatility. It measures the change in option value based on a 1% change in implied volatility. Vega declines as the options approach the knock-out phase; it falls when the option is out-of-money and deep in-the-money, and it is maximum when the underlying is around the strike price. Overall, Figure 9a–d display how accurately Model B3 predicts the option values and their Greeks, as little or no discrepancies are observed in each dual plot.

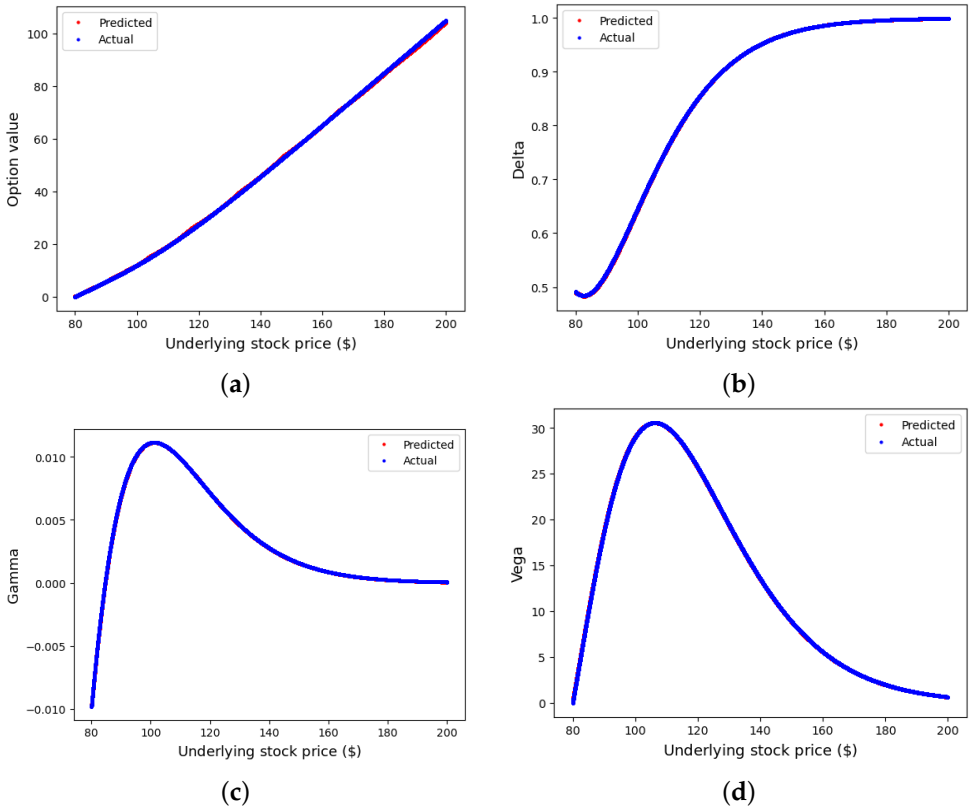


Figure 9. Option values and Greeks; (**a**) DO option value; (**b**) DO delta; (**c**) DO gamma; (**d**) DO vega.

5. Conclusions and Recommendations

This research suggested a more efficient and effective means of pricing the barrier call options, both with and without a rebate, by implementing the ANN techniques on the closed-form solution of these option styles. Barrier options belong to exotic financial options whose analytical solutions are based on the extended Black–Scholes pricing models. Analytical solutions are known to possess assumptions which are not often valid in the real world, and these limitations make them ideally imperfect in the effective valuation of financial derivatives. Hence, through the findings of this research, we were able to show that neural networks can be employed efficiently in the computation and the prediction of unbiased prices for both the rebate and non-rebate barrier options. This study showed that it is possible to utilize an efficient approximation method via the concept of ANN in estimating exotic option prices, which are more complex and often require expensive computational time. This research has provided an in-depth concept into the practicability of the deep learning technique in derivative pricing. This was made viable through some statistical and exploratory data analysis and analysis of the model training provided.

From the research, we conducted some benchmarking experiments on the NN hyperparameter tuning using the Keras interface and used different evaluation metrics to measure the performance of the NN algorithm. We finally estimated the optimal NN architecture, which prices the barrier options effectively in connection to some data-splitting techniques. We compared six models in terms of their data split and their hyperparameter search algorithm. The optimal NN model was constructed using the cross-validation data-split and the Bayesian optimization search algorithm, and this combination was more efficient than the other models proposed in this research. Next, we compared the results from the optimal NN model to those produced by other ML models, such as the random forest and the polynomial regression; the output highlights the accuracy and the efficiency of our proposed methodology in this option pricing problem.

Finally, hedging and risk management of barrier options are complicated due to their exotic nature, especially as the underlying is near the barrier. Our research extracted the

barrier option prices and their corresponding Greeks with high accuracy using the optimal hyperparameter. The predicted and accurate results showed little or no difference, which explains our proposed model's effectiveness. For future research direction, more theoretical underpinning seems to be lacking in connection to the evaluation/error analysis for all the proposed models used in this research. Another limitation of this work is the use of a fully simulated dataset; it will suffice to implement these techniques on a real dataset to estimate the effectiveness. The third limitation of this research lies in the convergence analysis of the proposed NN scheme, and future research will address this issue. In addition, more research can be conducted to value these exotic barrier options from the partial differential perspective, that is, solving the corresponding PDE from this model using the ANN techniques and extending the pricing methodology to other exotic options, such as the Asian or the Bermudian options.

Author Contributions: These authors contributed equally to this work. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data supporting this study's findings are available from the corresponding author upon reasonable request.

Acknowledgments: This work commenced while the first author was affiliated with the Center for Business Mathematics and Informatics, North-West University, Potchefstroom and the University of Johannesburg, both in South Africa. The authors wish to acknowledge their financial support in collaboration with the Deutscher Akademischer Austauschdienst (DAAD).

Conflicts of Interest: The author declares that they have no competing interests.

References

1. Aziz, S.; Dowling, M.M.; Hammami, H.; Piepenbrink, A. Machine learning in finance: A topic modeling approach. European *Financ. Manag.* **2022**, *28*, 744–770 [CrossRef]
2. Liu, S.; Borovykh, A.; Grzelak, L.A.; Oosterlee, C.W. A neural network-based framework for financial model calibration. *J. Math. Ind.* **2019**, *9*, 9. [CrossRef]
3. Beck, C.; Becker, S.; Grohs, P.; Jaafari, N.; Jentzen, A. Solving stochastic differential equations and Kolmogorov equations by means of deep learning. *arXiv* **2018**, arXiv:1806.00421.
4. Borovykh, A.; Bohte, S.; Oosterlee, C.W. Conditional time series forecasting with convolutional neural networks. *arXiv* **2017**, arXiv:1703.04691.
5. Babbar, K.; McGhee, W.A. A Deep Learning Approach to Exotic Option Pricing under LSVol; University of Oxford Working Paper. 2019. Available online: https://www.bayes.city.ac.uk/__data/assets/pdf_file/0007/494080/DeepLearningExoticOptionPricingLSVOL_KB_CassBusinessSchool_2019.pdf (accessed on 20 November 2022).
6. Sirignano, J.; Spiliopoulos, K. DGM: A deep learning algorithm for solving partial differential equations. *J. Comput. Phys.* **2018**, *375*, 1339–1364. [CrossRef]
7. Liu, S.; Oosterlee, C.W.; Bohte, S.M. Pricing options and computing implied volatilities using neural networks. *Risks* **2019**, *7*, 16. [CrossRef]
8. Yao, J.; Li, Y.; Tan, C.L. Option price forecasting using neural networks. *Omega* **2000**, *28*, 455–466. [CrossRef]
9. Li, C. The Application of Artificial Intelligence and Machine Learning in Financial Stability. In Proceedings of the International Conference on Machine Learning and Big Data Analytics for IoT Security and Privacy, Shanghai, China, 6–8 November 2020; pp. 214–219.
10. Lee, J.; Kim, R.; Koh, Y.; Kang, J. Global stock market prediction based on stock chart images using deep Q-network. *IEEE Access* **2019**, *7*, 167260–167277. [CrossRef]
11. Yu, P.; Yan, X. Stock price prediction based on deep neural networks. *Neural Comput. Appl.* **2020**, *32*, 1609–1628. [CrossRef]
12. Malliaris, M.; Salchenberger, L. A neural network model for estimating option prices. *Appl. Intell.* **1993**, *3*, 193–206. [CrossRef]
13. Black, F.; Scholes, M. The pricing of options and corporate liabilities. *J. Political Econ.* **1973**, *81*, 637–654. [CrossRef]
14. Klibanov, M.V.; Golubnichiy, K.V.; Nikitin, A.V. Application of Neural Network Machine Learning to Solution of Black-Scholes Equation. *arXiv* **2021**, arXiv:2111.06642.
15. Fang, Z.; George, K.M. Application of machine learning: An analysis of Asian options pricing using neural network. In Proceedings of the 2017 IEEE 14th International Conference on e-Business Engineering (ICEBE), Shanghai, China, 4–6 November 2017; pp. 142–149.
16. Hutchinson, J.M.; Lo A.W.; Poggio, T. A non-parametric approach to pricing and hedging derivative securities via learning networks. *J. Financ.* **1994**, *49*, 851–889. [CrossRef]

17. Bennell, J.; Sutcliffe, C. Black–Scholes versus artificial neural networks in pricing FTSE 100 options. *Intell. Syst. Acc. Financ. Manag. Int. J.* **2004**, *12*, 243–260. [CrossRef]
18. Yadav, K. Formulation of a rational option pricing model using artificial neural networks. In Proceedings of the SoutheastCon 2021, Virtual, 10–14 March 2021; pp. 1–8.
19. Ghaziri, H.; Elfakhani, S.; Assi, J. Neural, networks approach to pricing, options. *Neural Netw. World* **2000**, *1*, 271–277.
20. Anders, U.; Korn, O.; Schmitt, C. Improving the pricing of options: A neural network approach. *J. Forecast.* **1998**, *17*, 369–388. [CrossRef]
21. Buehler, H.; Gonon, L.; Teichmann, J.; Wood, B.; Mohan, B.; Kochems, J. *Deep Hedging: Hedging Derivatives under Generic Market Frictions using Reinforcement Learning*; SSRN Scholarly Paper ID 3355706; Social Science Research Network: Rochester, NY, USA, 2019.
22. Culkin, R.; Das, S.R. Machine learning in finance: The case of deep learning for option pricing. *J. Invest. Manag.* **2017**, *15*, 92–100.
23. De Spiegeleer, J.; Madan, D.B.; Reyners, S.; Schoutens, W. Machine learning for quantitative finance: Fast derivative pricing, hedging and fitting. *Quant. Financ.* **2018**, *18*, 1635–1643. [CrossRef]
24. Gan, L.; Wang, H.; Yang, Z. Machine learning solutions to challenges in finance: An application to the pricing of financial products. *Technol. Forecast. Soc. Change* **2020**, *153*, 119928. [CrossRef]
25. Hamid, S.A.; Habib, A. *Can Neural Networks Learn the Black-Scholes Model? A Simplified Approach*; Working Paper No. 2005–01; School of Business, Southern New Hampshire University: Manchester, NH, USA, 2005.
26. Le N.T.; Zhu, S.P.; Lu, X. An integral equation approach for the valuation of American-style down-and-out calls with rebates. *Comput. Math. Appl.* **2016**, *71*, 544–564. [CrossRef]
27. Umeorah, N.; Mashele, P. A Crank-Nicolson finite difference approach on the numerical estimation of rebate barrier option prices. *Cogent Econ. Financ.* **2019**, *7*, 1598835. [CrossRef]
28. Han, J.; Jentzen, A.; Weinan, E. Solving high-dimensional partial differential equations using deep learning. *Proc. Natl. Acad. Sci. USA* **2018**, *115*, 8505–8510. [CrossRef] [PubMed]
29. Ganesan, N.; Yu, Y.; Hientzsch, B. Pricing barrier options with DeepBSDEs. *arXiv* **2020**, arXiv:2005.10966.
30. Yu, B.; Xing, X.; Sudjianto, A. Deep-learning based numerical BSDE method for barrier options. *arXiv* **2019**, arXiv:1904.05921.
31. Itkin, A. Deep learning calibration of option pricing models: Some pitfalls and solutions. *arXiv* **2019**, arXiv:1906.03507.
32. Xu, L.; Dixon, M.; Eales, B.A.; Cai, F.F.; Read, B.J.; Healy, J.V. Barrier option pricing: Modelling with neural nets. *Phys. Stat. Mech. Its Appl.* **2004**, *344*, 289–293. [CrossRef]
33. Ghevariya, S. PDTM approach to solve Black Scholes equation for powered ML-Payoff function. *Comput. Methods Differ. Equ.* **2022**, *10*, 320–326.
34. Mehdizadeh, Khalsaraei, M.; Shokri, A.; Mohammadnia, Z.; Sedighi, H.M. Qualitatively Stable Nonstandard Finite Difference Scheme for Numerical Solution of the Nonlinear Black–Scholes Equation. *J. Math.* **2021**, *2021*, 6679484.
35. Rezaei, M.; Yazdanian, A. Pricing European Double Barrier Option with Moving Barriers Under a Fractional Black–Scholes Model. *Mediterr. J. Math.* **2022**, *19*, 1–16. [CrossRef]
36. Torres-Hernandez, A.; Brambila-Paz, F.; Torres-Martínez, C. Numerical solution using radial basis functions for multidimensional fractional partial differential equations of type Black–Scholes. *Comput. Appl. Math.* **2021**, *40*, 1–25. [CrossRef]
37. Eskiizmirliler, S.; Günel, K.; Polat, R. On the solution of the black–scholes equation using feed-forward neural networks. *Comput. Econ.* **2021**, *58*, 915–941. [CrossRef]
38. Chen, Y.; Yu, H.; Meng, X.; Xie, X.; Hou, M.; Chevallier, J. Numerical solving of the generalized Black-Scholes differential equation using Laguerre neural network. *Digit. Signal Process.* **2021**, *112*, 103003. [CrossRef]
39. Rich, D.R. The mathematical foundations of barrier option-pricing theory. *Adv. Futur. Options Res.* **1994**, *7*, 267–311.
40. Zhang, P.G. *Exotic Options: A Guide to Second Generation Options*; World Scientific: Singapore, 1997.
41. Kononenko, I.; Kukar, M. *Machine Learning and Data Mining*; Horwood Publishing: Chichester, UK, 2007
42. Reitermanová, Z. Data Splitting. In Proceedings of the WDS'10—19th Annual Conference of Doctoral Students, Prague, Czech Republic, 1–4 June 2010.
43. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
44. Segal, M.R. *Machine Learning Benchmarks and Random Forest Regression*; Division of Biostatistics, University of California: San Francisco, CA, USA, 2004.
45. Blasco, B.C.; Moreno, J.J.M.; Pol, A.P.; Abad, A.S. Using the R-MAPE index as a resistant measure of forecast accuracy. *Psicothema* **2013**, *25*, 500–506.
46. Oshiro, T.M.; Perez, P.S.; Baranauskas, J.A. How many trees in a random forest?. In Proceedings of the Machine Learning and Data Mining in Pattern Recognition: 8th International Conference, MLDM 2012, Berlin, Germany, 13–20 July 2012; pp. 154–168.