UNIVERSITY OF SUSSEX

DEEP LEARNING APPLIED IN OPTION PRICING

Pricing options using neural networks

Yashvardhan Singh
Supervisor: Michael Melgaard

Submitted for the degree of MSc Human and Social Data Science

University of Sussex

August, 2022

Declaration

I hereby declare that this project has not been and will not be submitted in whole or in part to another University for the award of any other degree.

Yashvardhan Singh
Candidate no: 245411
1st September 2022

UNIVERSITY OF SUSSEX

Candidate no:245411, MSc Human and Social Data Science

DEEP LEARNING APPLIED IN OPTION PRICING

Pricing options using neural networks

Summary

Options have been priced through mathematical models and methods, most popularly the Black-Scholes Model. This dissertation observes the performance of different Deep Learning methods (MLP, CNN, LSTM, CNN-LSTM, Dual-hybrid Jump-Diffusion CNN-LSTM) artificial, taking into consideration the effect of different volatilities while training it. The model trains and predicts option prices on the Russell 2000 and the NASDAQ 100 Index from 1st Jan 2012 to 30th Dec 2021. I compare the empirical pricing error of the Deep Learning models to the Black Scholes model and among themselves. As well as the performance of all the models on different volatilities in an attempt to identify the best for training deep learning models for predicting option prices. I study the performance of the models as well as their generalizability as I test their performance on a different dataset, i.e the NASDAQ 100 Index (the models are trained on the Russell 2000 index). I devise a dual hybrid model in an attempt to see if the traditional mathematical model(the Merton-Jump-Diffusion model) along with a deep learning model (the CNN-LSTM model) prices European-style options better than other deep learning methods.

Acknowledgements

I would like to express my gratitude to Michael Melgaard for their patience and advice throughout the process, he helped shape the project into one that was rewarding and motivating. Also, thank you to my friends and peers who helped me on the regular by being their for each other and motivating each other throughout the way. Finally, this project could not have been completed without the dataset provided by Wharton University of Pennsylvania's data service, OptionMetrics, Yahoo Finance, CBOE and FRED.

Contents

Chapter 1

Introduction

1.1 Background

Index options have a long and complicated history dating back to the early days of financial markets. At their simplest, index options are contracts that allow investors to buy or sell a specified number of units of an underlying stock or bond at a predetermined price on a particular date in the future. Since the introduction of index options, a number of variations have been introduced including currency options, equity index options and commodity index options.

In the early 17th century, the first financial instruments emerged, including loans, bonds and shares that were traded amongst bankers and merchants as a way of raising funds for businesses. Options contracts could not be priced until the development of pricing models such as the Black-Scholes model in the late 20th century which allowed financial institutions to accurately calculate the risk associated with trading in derivatives such as options and futures.

Options contracts have been around for quite a while, but it was only until 1973 that a method was devised to price these contracts. Black and Scholes, along with Merton formulated a differential equation that according to them was necessary to be satisfied by every option. The PDE's can could be solved for the price of call and put options.This turned out to be what we know today as the Black-Scholes model. It is still regarded as one of the best ways for pricing an options contract.[13](Hayes,2022)

Banks and investment bankers have been using methods and mathematical models to predict the future to make profits. Most amounts of these people have been using the Black-Scholes and now the new Black-Scholes-Merton models. Despite being one of the most popular ways of pricing options and an innovation in option pricing that undoubtedly permanently altered the financial markets, their potential is limited by their underlying unrealistic and restrictive assumptions. The major one is the constant volatility assumption of the Black-Scholes-Merton model. Researchers have tried to figure out a solution that cancels out this assumption like using deterministic volatility functions instead.[10](Dumas, Fleming, & Whaley, 1998)

Earlier the less widespread method of doing the same was through machine learning. [16]Hutchinson, Lo, and Poggio (1994) were the first ones to put effort into this and investigated the applicability of machine learning in option pricing by putting four different machine learning algorithms to the test on both simulated and real-world option prices for the S& P 500 index. Their results were promising despite the small amount of data, less computational power and far fewer research advancements than right now. With the rapid boom of technology and advancements in machine learning paired with the exponential growth of computational power, it makes much more sense now to look into option pricing with machine learning and neural networks.

The main benefit that machine learning algorithms are said to have over conventional option pricing strategies is that they are non-parametric. Machine learning algorithms are premise-free, meaning they don't rely on too simplistic assumptions about the underlying asset dynamics. The algorithms function by spotting patterns and correlations in data and then making use of this knowledge to forecast option prices. As a result, machine learning algorithms may be able to recognise intricate correlations in data without making any constraining assumptions. It is evident that machine learning and neural network-based models which are semi/non-parameterized are going to take over the old parameter-based mathematical models.

1.2 Research Questions

This dissertation mainly aims to answer the following research question:

Can Deep Learning models outperform the Balck-Scholes Model model on the Russell 2000 Index by London's FTSE Russell group and the NASDAQ 100 INDEX in pricing options?

I will further look into:

Can a dual-hybrid deep learning model, consisting of a traditional mathematical model for option pricing and a hybrid neural network architecture outperform other neural network models and the Black-Scholes model?

Whether using the volatility forecasted by AutoRegressive Integrated Moving Average (ARIMA) model help the neural network models to capture the volatility better and improve performance?

I would also like to look into which volatility measure turns out to be the best input for the neural networks.

1.3 Literature review

To justify the goal and contribution of this dissertation about current research, I provide a quick summary of the most notable existing studies on the subject of option pricing using artificial neural networks in this part.

To begin with, the idea of using the dual hybrid model, as done by [43](Zhao et al,2022), came from my supervisor after reading the research done by Zhao. I then chose the Merton-Jump-Diffusion Model and a combination of the CNN-LSTM model in order to

make a dual hybrid model, taking inspiration from Zhao.

Also, the work on option pricing using multi-layer-perceptrons by [30](Mads Højer Petersen,2019) has been a great inspiration for the work. Also in a conversation with him[1], he further guided me on how to proceed with the gathered data as the data was overwhelmingly large.

The earliest research done towards predicting option prices using machine learning and artificial intelligence was by [16]Hutchinson et al. (1994). The main aim of their research was can machine learning methods (non - parametric methods) like artificial neural networks for pricing the S & P 500 options over a span of time from 1987 to 1991. The performance of the models was based on the R2 score taken by doing a linear regression of the predicted option price against the actual price. Their results proved Artificial Neural Network model predictions to be having an out-of-sample R2 of 95.53 % solidifying that using machine learning and neural networks might be a better alternative to traditional methods.

Research regarding using machine learning for option pricing has been done for more than the past twenty years, but it has maintained its relevance due to the rapid boom of technology and advancements in machine learning paired with the exponential growth of the data and computational power available right now.

The 2004 research of Lajbcygier tries to make use of non-parametric models along with traditional pricing methods to improve performance and the results demonstrated that it is feasible to outperform the optimum conventional option pricing model by combining the flexibility of nonparametric artificial neural networks with rational option pricing.[19](Lajbcygier, 2004)

[4]Amilon (2003) predicted European option prices of an index of the Swedish stock market from 1997 to 1999 using a multi-layer perceptron artificial neural network. Amilion (2003) however presents the results in the form of the mean-squared error (MSE) making it difficult to have a comparison with [16](Hutchinson et al. (1994)) as they use the R2 score. Amilion's (2003) research showed his model to have an MSE of 5.57 for bid price and 6.40 for ask price. Amilion (2003) had the benefit of being 9 years ahead from Hutchinson with the new technology and better computing.

To make our results easier to compare with other previous and future works we will be using mean squared error (MSE), as it is normalized, hence being comparable across currencies and other research. The normalization occurs as the call price and the close price(the price of the underlying asset) are both divided by the strike price, cancelling out the units. I will talk more about it later in this thesis.

[12]Gradojevic et al. (2009) figured out that Modular neural networks perform better than the rest of the neural networks when it comes to predicting out-of-the-money options. They suggested estimating the pricing functions using the modularity features of the MNN. MNN not only had significant advantages over Black-Scholes but was also better than the other neural network methods.

[29] Park, Kim & Lee (2014) in their research compare six different parametric and non-parametric machine learning and neural network models that predict option prices on the KOSPI 200 options. They wanted to determine which model best fits the index options data, as a result of the research they found out that non-parametric machine learning models have significantly outperformed the Black-Scholes model in predicting the KOSPI 200 index option prices.

[22]Liu et al (2019) propose an artificial neural network model that aims at reducing the computation time of prediction of option prices, which is especially targeted at high-dimensional financial models. They test ANN models on three different methods, the Black-Scholes equation, the Heston model and Brent's root-finding method for implied volatilities. Their result showed that ANN can predict option prices and figure out implied volatilities efficiently and accurately.

Despite over 20 years of research, the topic of option pricing using machine learning models in particular Neural Networks has stayed relevant. This can be credited to the exponential growth of computational power, more availability of data to work on and more advancements in machine learning and neural networks. Although there is a lot more literature on this subject, the research carried out on the topic is quite uniform with some research

on newer models and techniques.

1.4 Structure of the Thesis

In this section, I would like to explain the structure of the thesis before straight in. I have divided the thesis into several sections, followed by a conclusion at the end. I will provide a brief of each of the chapters.

In chapter 2, I will be covering the Black-Scholes model. To understand the same I will go through what options are and what parameters account for the price of options. I will also discuss Ito's Lemma and the Black-Scholes in brief as their derivation is out of the scope of this research. I will discuss the Black-Scholes formula for option pricing, while also taking into account all of its underlying assumptions. In addition to this, I talk about the ARIMA model and how it can be used to forecast volatility, and ending the section with the put-call parity, assuming that the parity holds in place, it makes our results applicable to put options as well(as I will only be covering call options while carrying out the research).

In chapter 3, I will discuss the Deep neural networks I have used in our research. This section will briefly cover the theory behind deep learning and neural networks, also talk about the properties that provide artificial neural networks with an advantage over traditional methods and discuss the different deep learning models I have used in the research. I will further also explain the activation functions, loss functions and optimizers used in the models.

In chapter 4, I will be going over the methodology that has been used to carry out this thesis, from the collection of data, volatility forecasting, data preprocessing, the structures of the neural network models used, and how these models have been optimized. I will also explain the metrics used while training and evaluating the model.

In chapter 5, I discuss the results of the Deep learning models, compared to the Black-Scholes model. And to answer the research questions I have split the results into parts, the first part shows us the results of each and every model on the Russell 2000 Index and compares those performances with the Black-Scholes model as well as the real values. In the next section, I will check which model performs the best on each volatility, this

will also allow us to see if the ARIMA model presents better results than the rest of the volatilities for each model. Then I use the results to find the best model with the best volatility for the prediction of option prices. Further, I look if the benchmark model also performs the best when it comes to other datasets (in this case the NASDAQ 100), this will show the generalizability of the model.

The codes producing the results presented in this dissertation are available in the following Dropbox folder: Dropbox-to-Python-Notebooks

Chapter 2

Black-Scholes Model for pricing options

In this chapter, I will be discussing the derivation of the Black-Scholes model for pricing options. I will also discuss extensions of the Black-Scholes model, that is the Merton Jump Diffusion [18](M. Joshi,2003). I will use the Black-Scholes model as a benchmark for examining the performance of the CNN models that I will be creating to price options in this research.

I have based this chapter majorly on the resources by [15](Hull, 2015),[24] (Marzi & Turnbull, 2007),[36](Shi, 2019), [30](Petersen 2019),[28](Paolucci, 2020) and [21](Liang & Cai, 2022).

2.1 Options

Options are a type of financial derivative that gives the holder the right, but not the obligation, to buy or sell an underlying security at a set price or by a certain date. To value an option, it is necessary to calculate the future value of the option when the underlying security reaches a particular price.

To price an option is a very complex task and the first method to price options was developed by Black-Scholes in 1973. Nowadays Black-Scholes is one of the most popular and widely used methods to price options. In this research, I will also use the

Black-Scholes model (BSM), which was developed in the 1960s to value stock options. The Black-Scholes Model was developed by Myron Scholes and Robert Merton to value options traded on stock exchanges. They built on previous models developed by Paul Samuelson, who had previously developed a method called the theory of" continuous compounding" to value bonds. The main assumption of the Black-Scholes model is that the stock follows a straight-line movement with no dividends paid out by the company. The BSM also assumes that the underlying asset follows a geometric Brownian motion process, which means that the stock price changes randomly over time according to the volatility of the underlying asset. It also assumes that the price at which the option is purchased is known and that the option can be exercised only on the expiration date. Given this information, it is possible to calculate the value of the option by using the BSM and using it to predict the likelihood of the stock price reaching a particular level at a particular time in the future. The assumptions of the BSM may be unrealistic, particularly for options that are no longer currently priced in the current market. However, the BSM is still widely used due to its simplicity and computational efficiency. Understanding the basic principles of the BSM is an important first step for anyone interested in derivatives trading or financial engineering.

2.2 Different parameters that account for an option

In this section I will be covering the parameters that carry the key information of an option, these are the parameters that will be used throughout the research. This will be covering some of the input, the output and other information-providing parameters. The other variables used in the thesis are explained in 4.1 Data collection.

Close Price (S) - The final transaction price of security before the market formally shuts for regular trading is known as the closing price or cash value. Investors typically use closing prices as a benchmark for evaluating a stock's performance over the previous day, and they are widely used to create line graphs that show historical price changes over time.

Strike Price (K) - A derivative contract's strike price is the predetermined price at which it can be purchased or sold when it is executed. The striking price of a call option is the price at which the option holder can purchase the asset; that of a put option is the price at which the security may be sold. It is also known as the exercise price.

Date - The date on which the option has been issued.

Exdate - The final day when derivative contracts, such as options or futures, are still in effect is known as an expiration date. Investors will have made their decision on their expiring stake on or before this day.

Flag - It can either take the value 'C' for a call option and 'P' of a put option

Volume - The volume of a stock market trade indicates how frequently shares are exchanged between buyers and sellers.

Best Bid - The highest indicated price at which someone is ready to buy a certain asset is referred to as the "best bid," and it thus represents the best price that someone may sell at the market.

Best Offer - The lowest asking price from rival market makers or other sellers for stated security is known as the best ask (also known as the best offer). The ask (offer) price is essentially the lowest price a seller would accept for an item and the highest price a buyer can currently anticipate paying via a market order.

2.3 Ito's Lemma

In 1952, Ito studied a complicated problem involving communication between two stations. He came up with a solution that was much faster than any other method he had tried. He named his method "Ito's algorithm." Ito's lemma is a fundamental result in probability theory. It states that the sum of the successive increments of independent random variables is normally distributed.

Let X_n be a sequence of independent random variables with common expectation value μ and variance σ^2. Then, as $n \to \infty$, the random variable $S_n = X_1 + X_2 + \ldots + X_n$is normally distributed with mean μ and variance σ^2.

More formally, if we denote z_i as the ith element of the sequence $\{X_1, X_2, ...\}$, then $Z_n = \sum_{i=1}^{n} z_i$ is the sum of n i.i.d. random variables and $Z_n/n \to \mu - \sigma^2$ with probability one as $n \to \infty$.

Notice that the above statement involves only random variables and has no reference to

the underlying process that generates these variables. In fact, it can be proved that this lemma is true even if all the random variables are negative numbers or zeros. Furthermore, since $X_1+X_2+..+X_n$ is the sum of two sums of two variables, we can generalize it to show that the sum of three or more independent random variables is also normally distributed.

The above result is called Ito's lemma. Here we show that for sequences in discrete time, this lemma implies that their sum is distributed as a Poisson process with a mean equal to the sum of the individual quantities.

2.4 Black-Scholes PDE

he Black Scholes PDE is a famous equation in mathematical finance that helps to price options and other derivatives. The equation is named after the two economists who developed it, John Black and Myron Scholes. It is one of the fundamental equations of financial mathematics. The equation was first published by Scholes and Fischer in 1973.

Since then, it has been widely studied and applied in the analysis of financial derivatives such as options and bonds. It has helped to quantify risk and enable better pricing and hedging strategies.

The partial differential equation (PDE) which describes the price of the option is given by the Black-Scholes model, which is described below. The PDE is mathematically defined as:

$$\frac{\partial S}{\partial t} = -r_s^2 \frac{\partial^2 S}{\partial x^2} + p^2 \frac{\partial^2 S}{\partial p^2},$$

where x is the maturity or expiry date of the option, p is the price of the underlying asset at the time when the option is exercised, and r_s represents the risk-free interest rate at the time of exercise. The price of the option is determined as the solution of the above PDE using calculus methods.

2.5 The Black Scholes PDE and Option Pricing Formula

The Black-Scholes formula is one of the most widely used options pricing models. It was developed by the Nobel Prize-winning economist, Eugene F. Black and Myron Scholes, in 1973. The formula states that the value of an option can be calculated using the following variables: (1) the price of the option, (2) the time to maturity of the option, (3) the risk-free interest rate, and (4) the underlying stock's volatility. I talk about how the Black-Scholes formula is derived from the PDE and how it can be used for pricing call options. The price of a call option is the option buyer's cost for purchasing the option. It is the cost of exercising his right to buy or sell the underlying asset at a specific price on or before a given date. It can be calculated using the following equation :

$$P_C = K * S^2 * e^{-rT}$$

Where P_C is the price of a call option; K is the strike price of the call option; S is the underlying stock's market price; r is the risk-free interest rate; and T is the time to maturity of the option. The underlying stock price is the most important variable in the Black-Scholes model because it can affect the value of the option. This is because the higher the price of the underlying stock, the higher the value of the option will be. The Black Scholes Formula is obtained by solving the black scholes PDE. The initial boundary conditions are p=0 at time t=0 for European options. The European call option can be calculated as :

$$C_I = \int_0^\infty \frac{e^{-\lambda x}}{x} dx = \left(\frac{1}{\lambda}\right) \int_a^b e^{-t} dt$$

where the position p of underlying asset from time t to T is given by :

$$p(t,T) = \begin{cases} a & \text{if } t \leq t_0 \\ b & \text{if} t_0 < t \leq T \end{cases} \quad \text{and } c = 0.$$

The equation above represents the derivative of the payoff function of a European call option with respect to p. Let $F(a,b,c,x)$ be the payoff function and let $f(tT)$ be the corresponding derivative. Then the Black-Scholes PDE can be obtained as follows:

$\frac{df}{dx} = \frac{e^{-x}}{2x} - \frac{e^{-a}}{x} + \frac{e^{-b}}{x} - \frac{2ce^{x}}{x^2+c}$ .Applying the boundary conditions we get:

$$\begin{cases} f(t,T) = 0 \text{ when } t \leq a \\ f(t,T) = b \text{ when } t \geq a+b. \end{cases}$$

Substituting the first boundary condition we get :

$$e^{-x} = \frac{\lambda}{x}$$

where λ is the price of the option. Thus solving for the value C we get:

$$C = \left(1 - e^{-a}\right) b - \frac{\lambda}{a}$$

The value of a European call option is always positive. This is because the buyer of the contract can always exercise the contract at any time up until expiration,meaning that he still has the potential to make profits on the trade even if the stock price doesn't move. Substituting Lambda we get the European call option price formula by Black- Scholes which can be written in terms of variables as :

$$C = S - K \times \left(1 - e^{-a}\right) - b.$$

In the above equation S refers to the price of the underlying stock and K refers to the strike price of the option.

2.6 Black Scholes Model for Option Pricing

Black Scholes model is one of the most popular and widely used models to price options. The Black-Scholes pricing model can be used to calculate the intrinsic value of a call option and put options, given the strike price, time to maturity and the risk-free interest rate. Throughout this thesis I will be looking into call options only.

To get the Black-Scholes close form solution for European option price, the values a and b need to be calculated. Calculation of a and b involves rigorous mathematics and hence is out of the scope of this thesis. By substituting the a and b values and simplifying the equation derived at the end of the last section, the Black Scholes formula for pricing and European call options is given as:

$$c(S_0, r, q, \alpha, K, T) = S_0 e^{-qT} N(d_1) - K e^{-rT} N(d_2)$$

c	Call option price
S	Current stock price
K	Strike price of the option
r	Risk-free interest rate (a number between 0 and 1)
σ	volatility of the stocks return (a number between 0 and 1)
t	time to option maturity (in years)
N	normal cumulative distribution function

Where d_1, d_2 are given by:

$$d_1 = \frac{\ln \frac{S_0}{K} + r - q + \frac{\sigma^2}{2} T}{\sigma \sqrt{T}}$$

$$d_2 = d_1 - \sigma \sqrt{T}$$

And N is given by:

$$N(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{\frac{x^2}{2}} dx$$

2.7 Merton-Jump-Diffusion

Robert Merton first proposed the jump-diffusion model in 1976. It describes the behaviour of stock prices by combining minor, daily "diffusive" movements with bigger, sporadic "jumps." Jumps make "crash" scenarios more plausible and render the usual dynamic replication hedging strategy of the basic Black-Scholes model ineffective. As a result, option prices rise relative to the Black-Scholes model and rely on investors' level of risk aversion.

The Merton-Jump-Diffusion model is a seminal theoretical framework for option pricing. The model is based on the assumption that there is a relationship between volatility and option prices. The model assumes that the variance of the stock's return over time follows a random walk, i.e. the stock returns have a random but unpredictable distribution (i.e. a normal distribution).

The stock price S_t follows the random process

$$dS_t/S_t = \mu dt + \sigma dW_t + (J-1)dN(t)$$

μ represents the drift rate, σ represents volatility, W_t shows Weiner Process, J is the jump and $N(t)$ is the total number of jumps up till time t.

Merton also proposed closed form solutions for the price of put and call options. The closed form solution for pricing european call options is:

$$P_{MJDcall}(S,K,T,r,\sigma,m,v,\lambda) = \sum_{k=0}^{\infty} \frac{\exp(-m\lambda T)(m\lambda T)^k}{k!} V_{BScall}\left(S,K,T,r_n,\sigma_k\right)$$

The volatility and drift from the formula above, depends on the number of jumps happening. Mathematically these values can be given as:

$$\sigma_k = \sqrt{\sigma^2 + k\frac{v^2}{T}}$$
$$r_k = r - \lambda(m-1) + \frac{k\ln(m)}{T}$$

I will be using the Merton-Jump Diffusion model to form dual hybrid models, i.e a hybrid neural network which takes input from another mathematical model. This might help the neural network model by taking the advantage of the logic and the structure of the traditional models.

2.8 ARIMA

Volatility is one of the most important factors to consider when forecasting future stock or option prices. A popular volatility forecasting model is the ARIMA model. In this section, I will explore the ARIMA model and how it can also be used to forecast option prices.

The ARIMA model was developed by Box and Jenkins (1970) in an attempt to provide a statistical model for time series data that better accounted for non-linearities in data. The ARIMA model is comprised of three components - a seasonal component, a trend

component and a residual component. Each of these components can be modelled using regression techniques or complex mathematical equations. ARIMA modelling is particularly useful for predicting stock prices because it is used to project the future evolution of an underlying time series. However, because these models depend on past values to calculate future values, they cannot account for any unforeseen events that may occur in the future. Furthermore, they are not effective at forecasting stock prices that do not exhibit any significant trends in historical data. Despite these limitations, ARIMA models are still widely used by analysts to forecast stock prices because they offer an efficient alternative to complex and sophisticated mathematical models.

The ARIMA model can also be used for pricing options in the stock market. This can be very useful for risk managers because it allows them to assess the level of risk associated with a particular option position. For example, a portfolio manager may decide to purchase call options on stock ABC. Using the ARIMA model, it is possible to calculate the implied volatility of stock ABC over a certain period. This information can then be used to determine the risk profile of the portfolio and take the appropriate measures to minimize potential losses. It can also be used to determine whether a potential investment is worthwhile. The ARIMA model can also be used to make accurate predictions about the direction of the stock market and the impact on the price of individual stocks. Overall, this indicates that the ARIMA model is a useful tool that can be used by investors to determine the best way to manage risk and maximize profits. I will look into how the volatility forecasted by the ARIMA model affects the performance of the neural networks examined in this thesis.

2.9 Put-Call Parity

Put-call parity is a technical analysis concept that states that the price of a put option should equal the price of a call option with the same strike price and expiration date. This equality is determined by the principle of maximum profit. Mathematically it can be defined as:

$$\boxed{p + Se^{-qT} = c + Ke^{-rT}}$$

A portfolio holding a long position in the underlying asset and a put option should equal a portfolio holding a long position in the call option and the strike price, according to the

above equation illustrated in this combination. This connection should exist based on the put-call parity; otherwise, there would be a chance for arbitrage.

Re-arranging the above equation as

$$\boxed{p = c + Ke^{-rT} - Se^{-qT}}$$

From the above equation, the put-call parity states that a portfolio consisting of a long position in the underlying asset, a long position in the call option, and a short position in the strike price should be equivalent to the put option. As the value of the put option has been obtained as a function of a call option, the underlying asset and the strike price, all the results obtained for the call option can be applied to put options (given that the price of the underlying asset and strike price are same). [40](Witzany, 2020)

Chapter 3

Deep Learning

This chapter's main goal is to introduce the deep Learning theory that underpins this thesis's methodology. I begin by explaining machine learning, followed by a simple explanation of artificial neural networks which includes Front propagation, backpropagation and optimization. Then I explain the different neural network models I apply throughout the thesis, which are MLP, CNN, LSTM, CNN-LSTM and the dual hybrid model. I further talk about the activation functions, loss functions and optimizers of a neural network.

I have based the exposition of this chapter majorly on [20](Lee et al., 2021), [26](Mostafa et al, 2017),[42](Zhang et al., 2015), [25](Mena-Oreja & Gozalvez, 2020),[30](Peterson,2019 [8](Dohare & Tulika, 2021) and [33] (Sawant et al ,2021)

3.1 Machine Learning

Machine learning which can be seen as a part of AI allows systems to learn from their past performance without having to be explicitly programmed. The goal of machine learning is to create computer systems that can access data and utilise it to acquire knowledge on their own. With more data to learn from, the algorithms are better equipped to give insights into the underlying probability distribution of the data and generate accurate predictions.[34].(Seligh,2022)

Throughout the course of this thesis, the ML algorithms we focus on are supervised learning type. We minimise a loss function by modifying the model parameters using hyperparameter optimization.[3](Alpaydin,2014). A machine learning algorithm based on su-

pervised learning is linear regression. It executes a regression operation. Regression uses independent variables to model a goal prediction value. It is mostly used to determine how variables and forecasting relate to one another. These relationships are identified by modifying the model parameters using ordinary least squares to reduce the mean squared error between the estimated model predictions and the actual data. This structure is prevalent even in more superior supervised learning algorithms, the optimization methods, specifications and loss functions are different. Alpaydin defined supervised learning as a machine learning algorithm where the focus is to observe the relationship between an input x and an output y.

Supervised learning has two categories Regression and Classification. Classification algorithms focus on finding relationships between the input parameters and categorical output. The following classification techniques are often used: decision trees, k-nearest neighbour, random forest, support vector machines (SVM), linear classifiers, and SVM. Regression algorithms as discussed above focus on observing the relationships between inputs and an output y. Linear regression, logistical regression, and polynomial regression are popular regression algorithms. The main focus of this thesis is to figure out a relation between the Option Parameters and their price, therefore making it a regression task.

3.2 Artificial Neural Networks

When the model complexity is such that the model makes accurate and reliable predictions, there is a favourable trade-off between bias and variance. Therefore, while make a neural network with a low bias and low variance, it is important to discover the model's ideal level of complexity in relation to the inputs utilised, the number of hidden layers, the number of perceptrons, etc.[37](R. E. Uhrig,1995)

An artificial neural network is a set of artificial neurons which loosely model the neurons in a human brain. As shown in figure 1, each artificial neuron has inputs and gives a single output that can be forwarded to multiple artificial neurons. The inputs in the input layer in a neural network can be a number, an image or a document and outputs from other neurons. The output of the final neurons of the network produces the result of the complete task.

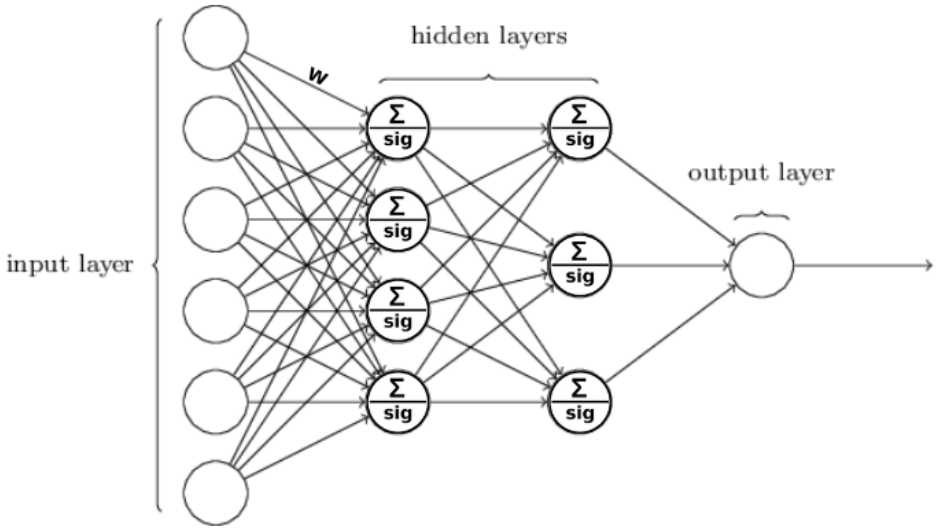

Figure 3.1: an example of multi-layer Artifical Neural Network [**?**](Kassabgi,2017)

The total of all inputs, weighted by the weights of the connections between the inputs and the neurons(w, as in figure 1), is used to calculate the output of neural networks. Each perceptron has an activation function(like the sigmoid function in the neurons of the hidden layer in figure 1) that is in a nonlinear fashion, which processes the weighted sum and transfers the findings to the linked perceptrons in the next layer. External data, such as photographs and texts, are the first inputs. The final outputs, such as identifying an item in a picture, complete the task.

The weights of the connections of inputs to the neurons is calculated after training the neural network, where the target is to find the weights that make the model more and more accurate. Gradient Descent is the method that is used to find the optimal weights for the neural network.

I will use the artificial neural networks for pricing options by training on the data that we have collected. I will be optimizing and analysing the performance of the ANNs in the research to figure out the hyperparameters that help the model predict option prices with the most accuracy.

I will now discuss the process that takes place to train a neural network, namely Forward Propagation, Backpropagation and Optimization

3.2.1 Forward Propagation

The computation and storing of intermediate variables (including outputs) for a neural network in the proper sequence from the input layer to the output layer is referred to as forward propagation (also known as forward pass). I will now go through this process step by step by taking an example of a neural network with one hidden layer.

Let's consider an input $x \in \mathbb{R}^d$ and for simplicity let's assume that the hiddle layer has no bias term. Then the inter-mediate value here is:

$$\mathbf{z} = W^{(1)}x$$

Where the weight parameter of the hidden layer is $W^{(1)} \in \mathbb{R}^{h \times d}$. Hidden activation vector of length h is obtained by passing $z \in \mathbb{R}^h$ through the activation function ϕ, which can be given by:

$$\mathbf{h} = \phi(\mathbf{z})$$

The output of the hidden layer will also be an intermediate variable. The weights of the parameters of the output layer can be assumed as $W^{(2)} \in \mathbb{R}^{q \times h}$. Then the output layer variable can be:

$$\mathbf{o} = W^{(2)}h$$

Taking the loss function to be l and the real output label is y, then the loss term will be:

$$L = l(\mathbf{o}, y)$$

Given the hyperparameter λ and using l_2 regularisation, the regularisation term will be:

$$s = \frac{\lambda}{2}(\left\|\mathbf{W}^{(1)}\right\|_F^2 + \left\|\mathbf{W}^{(2)}\right\|_F^2)$$

Hence the regularized loss of the model will be:

$$J = L + s$$

3.2.2 Back Propagation

The process of determining the gradient of neural network parameters is known as back-propagation. In essence, the approach applies the chain rule from calculus to traverse the network from the output to the input layer in reverse order. Any intermediate variables (partial derivatives) needed to calculate the gradient with respect to a few parameters are stored by the method.

Assume that we have the functions $Y = f(X)$ and $Z = f(Y)$, where in the inputs and outputs X, Y, Z are tensors. The derivative of Z with respect to X using the chain rule:

$$\frac{\partial Z}{\partial X} = \text{prod}(\frac{\partial Z}{\partial Y}, \frac{\partial Y}{\partial X})$$

Using the prod operator to multiply, as it carries out all the necessary operations like swapping inputs and transposition of the tensors.

The main focus of backpropagation is to calculate the gradients $\frac{\partial J}{\partial W^{(1)}}$ and $\frac{\partial J}{\partial W^{(2)}}$. This can be achieved by applying the chain rule to calculate the derivative of each parameter and intermediate variable. As we will need to start from the back, i.e by calculating the gradient of $J = L + s$ with respect to L the loss term and s the regularization term. Which turns out to be:

$$\frac{\partial J}{\partial L} = 1 \text{ and } \frac{\partial J}{\partial s} = 1$$

The derivative of J can be taken with respect of the variable of output layer o by the chain rule:

$$\frac{\partial J}{\partial o} = \text{prod}(\frac{\partial J}{\partial L}, \frac{\partial L}{\partial o}) = \frac{\partial L}{\partial o} \in \mathbb{R}^q$$

The derivative of s can be taken with respect to the parameters of the neural network $W^{(1)}$ and $W^{(2)}$ by the chain rule:

$$\frac{\partial s}{\partial W^{(1)}} = \lambda W^{(1)} \text{ and } \frac{\partial s}{\partial W^{(2)}} = \lambda W^{(2)}$$

Now the gradient of the model parameter closest to the output layer $\frac{\partial J}{\partial W^{(2)}}$ using chain rule:

$$\frac{\partial J}{\partial W^{(2)}} = \text{prod}(\frac{\partial J}{\partial o}, \frac{\partial o}{\partial W^{(2)}}) + \text{prod}(\frac{\partial J}{\partial s}, \frac{\partial s}{\partial W^{(2)}}) = \frac{\partial J}{\partial o}\mathbf{h}^\top + \lambda W^{(2)}$$

Continuing the backpropagation to the hidden layer we can find the gradient $\frac{\partial J}{\partial h} \in \mathbb{R}^h$ by using the chain rule. Which turns out to be:

$$\frac{\partial J}{\partial \mathbf{h}} = \text{prod}(\frac{\partial J}{\partial \mathbf{o}}, \frac{\partial \mathbf{o}}{\partial \mathbf{h}}) = \lambda \mathbf{W^{(2)}}^\top \frac{\partial J}{\partial o}$$

As the activation function ϕ is applied to all the elements separately, hence while calculating the gradient with respect to the intermediate variable $\mathbf{z}$ I use the elementwise multiplication operator $\odot$.

$$\boxed{\frac{\partial J}{\partial z} = \text{prod}(\frac{\partial J}{\partial \mathbf{h}}, \frac{\partial \mathbf{h}}{\partial z}) = \lambda {W^{(2)}}^\top \frac{\partial J}{\partial h} \odot \phi'(\mathbf{z})}$$

Finally the gradient with respect to the parameter closest to the input layer, i.e $\frac{\partial J}{\partial W^{(1)}}$ can be calculated.

$$\boxed{\frac{\partial J}{\partial W^{(1)}} = \text{prod}(\frac{\partial J}{\partial z}, \frac{\partial z}{\partial W^{(1)}}) + \text{prod}(\frac{\partial J}{\partial s}, \frac{\partial s}{\partial W^{(1)}}) = \frac{\partial J}{\partial z}\mathbf{x}^\top + \lambda W^{(1)}}$$

3.2.3 Optimization

Optimization is the process of minimizing the loss (cost) function, which in turn improves the model performance. This process is used to get the best parameters, which return the minimum loss.

To find the minimum values of the parameters, either a closed form solution can be derived algebraically or approximation using an iterative method. In deep learning, iterative methods are usually the only solution, due to the fact that the cost functions are dependent

on a large number of input variables and there is almost never an algebraic way to derive a closed-form solution.

There are different optimizers for this problem, I will use the example of gradient descent to explain the process of optimization. I will also discuss the available optimizers further in this chapter.

In gradient descent, after every iteration (that is after every forward and backward pass) it adjusts the values of the parameters (which are initialised during forward pass) according to the opposite direction of the gradient of the cost function (which is calculated during the backpropagation).

It can be mathematically formulated as:

$$\hat{y} = model_{\mathbf{W}}(x)$$

$$\mathbf{W} = \mathbf{W} - \frac{\partial J(y, \hat{y})}{\partial W}$$

$\hat{y}$	prediction by the model for input x
$\mathbf{W}$	parameters
$\frac{\partial J}{\partial W}$	gradient indicating the direction to push $\mathbf{W}$ value to decrease J
α	learning rate

3.3 Different Neural Networks

In this section I will briefly discuss the different models that I will be using throughout this thesis. I will be comparing the performance of these model against the Black-Scholes-Merton and also against each other.

3.3.1 MLP - Multi-Layer Perceptron

MLP is a type of artificial neural network, which consists of different layers, different numbers of neurons and activation functions. Past Literature has shown the use of multi-layer-perceptrons predicts option prices and the results have outdone the Black-Scholes model.[14](Karatas et al.2019) in their research evaluated the performance of several model processes and came to the conclusion that a neural architecture with four hidden layers, each containing 120 neurons and Leaky ReLu activation functions, is the most

ideal. The neural network discussed above is a multi-layer perceptron neural network.

3.3.2 CNN - Convolutional Neural Networks

Convolutional Neural Networks though known for their ability to capture information from image data are also growing in popularity in different domains. Dimensions of CNN for Option Pricing differ from model to model, but usually, the one-dimensional version of the CNN model is used, for fitting time-sequence data. The CNN pricing models usually stack 3 sets of 1-D CNN layer, a batch normalization layer and a max pooling layer, which at the end of all the sets will be followed by a dense layer with one neuron acting as the output.

3.3.3 LSTM - Long Short-Term Memory Neural Network

LSTM is a type of Recurrent Neural Network, which due to its ability to capture long-term dependencies in data, is in general used to solve classification and regression problems of time series data.LSTM uses three gates to learn hidden information in the time series, the input gate i_t, forget gate f_t and output gate o_t.

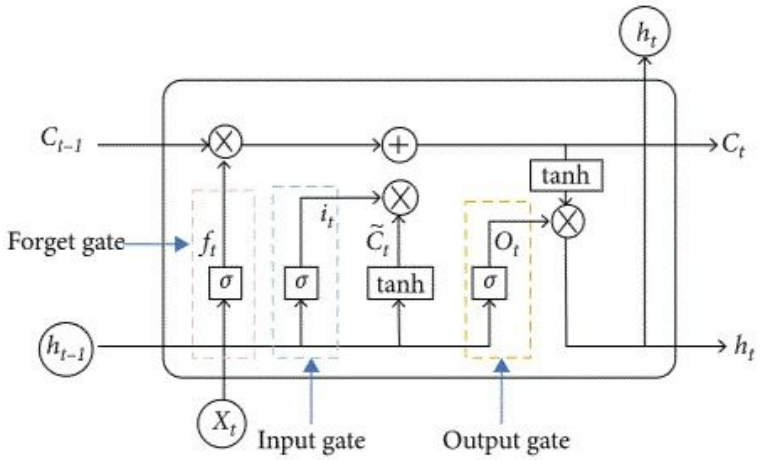

Figure 3.2: The structure of an LSTM unit[**?**](Hindawi)

The mathematical formula of LSTM can be given as:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C_t} = \sigma(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C_t}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_0)$$

$$h_t = o_t * tanh(C_t)$$

σ	Sigmoid Function
$tanh$	Hyperbolic Tangent Function
f_t	Forget Gate
i_t	Input Gate
o_t	Output Gate
W	weights
C	Cell state

3.3.4 CNN-LSTM

CNN is frequently used in feature engineering because it has the property of focusing on the most evident elements in the line of sight. LSTM, which is frequently employed in time series, has the property of growing in accordance with the passage of time. [8](Dohare & Tulika, 2021)A stock forecasting model based on CNN-LSTM is developed in accordance with the properties of CNN and LSTM. The input layer, one-dimensional convolution layer, pooling layer, LSTM hidden layer, and complete connection layer are all components of the core CNN and LSTM structure.[23](Wei et al,2020)

3.3.5 Dual-Hybrid Model

3.4 Activation Functions

The activation function acts as a mathematical "gate" between the current neuron's input and its output to the following layer.By generating a weighted total and then including bias with it, the activation function determines whether or not a neuron should be turned

on. The activation function's objective is to add non-linearity to a neuron's output. I will be discussing the activations functions that we have examined throughout this thesis.

3.4.1 Linear

The activation is proportionate to the input in a linear activation function, also referred to as the "dentity function (multiplied by 1.0).

$$\boxed{f(x) = x}$$

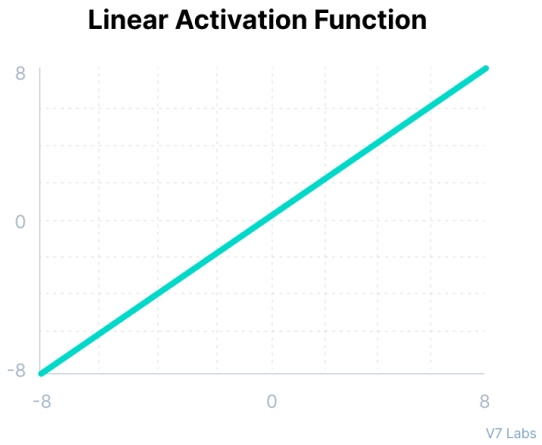

Figure 3.3: Graphical Representation of Linear Activation Function[6](Baheti,2022)

3.4.2 Sigmoid / Logistic Activation Function

Any real value may be used as an input for this function, and it returns values between 0 and 1.

As demonstrated below, the output value will be closer to 1.0 the larger the input (more positive) and closer to 0.0 the smaller the input (more negative).

$$\boxed{f(x) = \frac{1}{1 + e^{-x}}}$$

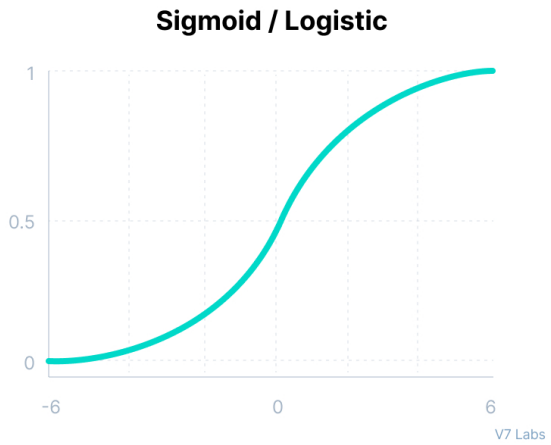

Figure 3.4: Graphical Representation of Sigmoid Activation Function[6](Baheti,2022)

3.4.3 Tanh - (Hyperbolic Tangent)

With a variation in the output range of -1 to 1, the Tanh function is remarkably close to the sigmoid/logistic activation function and even has the same S-shape. Tanh's output value approaches 1.0 when the input is greater (more positive), whereas it approaches -1.0 when the input is smaller (more negative).

$$\boxed{f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}}$$

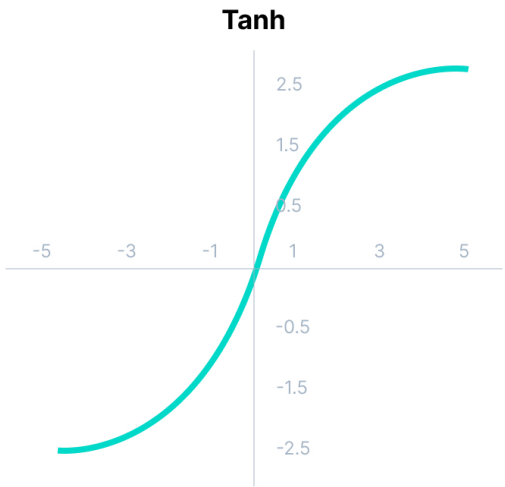

Figure 3.5: Graphical Representation of Tanh Activation Function[6](Baheti,2022)

3.4.4 RELU - Rectified Linear Activation Function

ReLU is a piecewise linear function that, if the input is positive, outputs the input directly; if not, it outputs zero. Because a model that utilises it is simpler to train and frequently performs better, it has evolved into the standard activation function for many different kinds of neural networks.

$$f(x) = max(0, x)$$

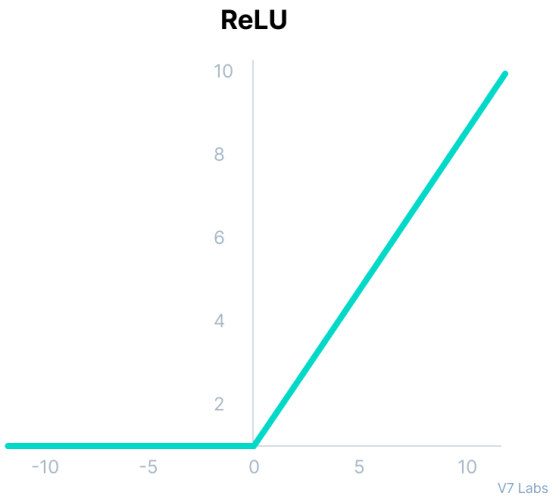

Figure 3.6: Graphical Representation of ReLU Activation Function[6](Baheti,2022)

3.4.5 ELU - Exponential Linear Unit

A variation of ReLU that alters the slope of the negative portion of the function is known as an exponential linear unit, or ELU. The negative values are defined by ELU using a log curve.

$$\text{ELU}= \begin{cases} x, & \text{for} x \geq 0 \\ \alpha(e^x - 1), & \text{for} x < 0 \end{cases}$$

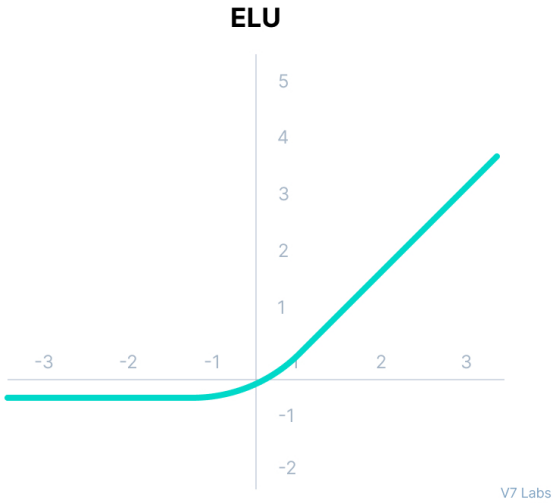

Figure 3.7: Graphical Representation of ELU Activation Function[6](Baheti,2022)

3.5 Loss Function

A neural network model's performance on a given task—typically regression or classification—is measured by a loss function. The focus during optimization is to try to minimize the loss function during backpropagation to make the Neural Network better.

3.5.1 Mean-Squarred-Error Loss Function

Mean-Squared-Error or MSE is calculated by taking the difference between the predicted value, squaring it and averaging it across the entire dataset. The MSE is mathematically defined by the following equation:

$$\boxed{MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

Advantage: MSE prevents the model from making any outlier prediction as it magnifies the error caused by that outlier by squaring it. So a high MSE value points out outliers.

Disadvantage: In real-life situations when we need a well-rounded model that performs good enough on the majority of data, outliers are often ignored. This is not possible as the squaring increases the error due to the outliers massively.

3.5.2 Mean-Absolute-Error Loss Function

Mean-Absolute-Error or MAE is calculated by taking the difference between the predicted value, applying absolute value to it and averaging it across the entire dataset. The MAE is mathematically defined by the following equation:

$$\boxed{MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|}$$

Advantage: All the errors are treated on the same linear scale, as we take an absolute value of the difference. Hence, the weights of the outliers are not going to be magnified providing a much more generic and even measure of how the model is performing on the data.

Disadvantage: As the outlier, errors are not magnified this might result in the model performing well on the majority of the data but it might end up with some very poop predictions every once in a while.

3.5.3 Huber Loss Function

Huber loss is a combination of both MSE and MAE. Hence, it brings the advantages of both of the functions into play. We can define the Huber loss as:

$$\boxed{L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2, \text{for} |y - f(x)| \leq \delta, \\ \delta|y - f(x)| - \frac{1}{2}\delta^2, & \text{otherwise} \end{cases}}$$

The Huber Loss function uses MAE for loss values bigger than delta which in turn reduces the weight placed on outliers while at the same time using MSE for loss values smaller than delta to maintain a quadratic function near the centre. Therefore results in a well-rounded model.

3.6 Optimizer

Optimizers are algorithms or methods that modify the weights and learning rates of the neural network to minimise losses. Optimization algorithms or strategies are responsible for reducing losses and providing the most accurate results possible. [9](Doshi, 2020)

3.6.1 Momentum

The exponentially weighted average of the gradients is taken into account in the approach in order to speed up the gradient descent procedure.[27](Muneer et al., 2022) The algorithm moves more quickly toward the minima when averages are used.

$$w_{t+1} = w_t - \alpha m_t$$

Where, m_t:

$$m_t = \beta m_{t-1} + (1-\beta)[\frac{\partial L}{\partial w_t}]$$

m_t	aggregate of gradients at time t [current]
w_t	Weight at time t
α	Learning rate at time t
∂L	Derivative of Loss Function Gate
∂w_t	Derivate of Weight at time t Gate
β	moving average parameter (const: 0.9)

3.6.2 RMSP - Root Mean Square Propagation

RMSP is an optimization method that outshines Adagrad and momentum in terms of result. It takes into consideration the exponential moving average instead of the cumulative sum of the squared gradients in the case of Adagrad. [33](Sawant et al., 2021)

$$w_{t+1} = w_t - [\frac{\alpha_t}{(v_t+\epsilon)^1/2}][\frac{\partial L}{\partial w_t}]$$

Where, v_t:

$$v_t = \beta v_{t-1} + (1-\beta)[\frac{\partial L}{\partial w_t}]^2$$

m_t	aggregate of gradients at time t [current]
w_t	Weight at time t
α	Learning rate at time t
∂L	Derivative of Loss Function Gate
∂w_t	Derivate of Weight at time t Gate
v_t	sum of the square of past gradients
β	moving average parameter (const: 0.9)
ϵ	A small positive constant (10-8)

3.6.3 Adam

Adam is built taking into consideration the strengths of the above optimizers, it controls the rate of gradient descent in such a way that it takes minimum time to reach the global minimum while having a step size big enough to pass all local minima while descending.[32](Prakhar, 2020)

$$w_{t+1} = w_t - \hat{v_t}(\frac{\alpha}{\hat{v_t} + \epsilon})$$

Where, $\hat{m_t}$ and $\hat{v_t}$:

$$\hat{m_t} = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v_t} = \frac{v_t}{1-\beta_2^t}$$

β_1	moving average parameter of momentum optimizer
β_2	moving average parameter of RMSP

Chapter 4

Methodology

Here I will describe the data collection and data pre-processing, followed by an explanation of the different processes like volatility forecasting, cross-validation, regularisation and hyperparameter tuning for optimizing the neural networks I will also discuss the structures of the Deep Learning Neural Networks models which are MLP, CNN, LSTM, hybrid model CNN-LSTM and the dual hybrid model in the form of Merton-Jump-Diffusion CNN-LSTM model. I will end this section with a discussion of the metrics used while examining the performance of the neural network models. All the processes of data preparation, training the model and analysis will be done using python and libraries like pandas, NumPy, math, scipy, stats, yfinance, Keras, TensorFlow, sci-kit-learn etc.

4.1 Data Collection:

The data used to carry out the research done in this thesis has been retrieved from five sources: OptionMetrics and CBOE (Chicago Board Options Exchange) data from these two sources were retrieved through Wharton Research Data Services (WRDS). The other two sources are Federal Reserve Economic Data (FRED) and Yahoo Finance's python library yfinance. The research in this thesis will focus on Russell 2000 and the NASDAQ 100 options dataset as they have European exercise styles. I have used two separate indices to showcase the generalization and the performance of the deep learning models on different datasets (different inputs). Also given the fact that Deep Learning models need a lot of data, it is also crucial that these options taken into consideration were

extremely liquid and traded across a wide range of strike prices and time-to-maturities, hence selecting London's FTSE Russell Group's Index Russell 2000 which is deemed as the bellwether of U.S Economy and the NASDAQ 100 index.

As discussed earlier I will be researching call options only and only downloading call options data for both the indices, but the results will apply to both types of options given the put-call parity (discussed in 2.9) holds.

Option price information data for both the indices downloaded from OptionMetrics consists of option information data i.e TICKER (RUT/NDX), Date, Exdate, Exercise type and Option Price data i.e StrikePrice, BestBid, BestOffer, ForwardPrice, Volume. This data provides most of the values necessary for our research. The data used for both indices are from 1st January 2012 (01-01-2012) to 1st January 2022 (01-01-2022).

I obtain the dividend yield rates of both the indices from OptionMetrics as well, this data is from 1st January 2012 (01-01-2010) to 1st January 2022 (01-01-2022). I retrieved Index data (Close Price / Underlying Asset value(S)) using Yahoo Finance (yfinance library) for both indices. The underlying asset price data observed is from 1st January 2000 to 1st January 2022 as I needed underlying data from 2000 to 2012 for volatility forecasting. As the Deep learning models are trained on volatilities by different forecasting methods and see which provides the best volatility input for the deep learning models.

RVX, which is Chicago Board Options Exchange's CBOE Volatility Index for Russell 2000 Index is also used while comparing the models on other forecasted volatilities. VXN (Chicago Board Options Exchange's CBOE Volatility Index for NASDAQ 100) is used when comparing the models on the NASDAQ 100. Both RVX and VNX have been beeretrieveded from Chicago Board Options Exchange's CBOE (through WRDS-Wharton Research Data Services). RVX and VXN data is from 1st January 2012 to 1st January 2022.

I have further used the Federal Reserve Economic Data (FRED) to get the risk-free rate (r).Market Yield on U.S. Treasury Securities at 10-Year Constant Maturity, Quoted on an Investment Basis as the risk-free rate as it is frequently regarded as one of the most accurate substitutes for a risk-free investment.

4.2 Volatility Forecasting

. Volatility is a key determinant of the price movement of options. Traders use volatility as a means of predicting future price movements. In this article, we will use an autoregressive integrated moving average (ARIMA) model to forecast the volatility of option prices based on historical data. An ARIMA model is one of the simplest methods for forecasting financial data. It assumes that data follows a trend over time and that this trend can be broken down into a series of linear functions that describe the underlying trend at different points in time. The model consists of three parts: a seasonal adjustment, an autocorrelation function, and a time series regression model. I use the data from 2002 till 2012 to train the model to predict the future volatilities, the model predicts volatility for all trading days after 1st Jan 2012 till 1st Jan 2022. I have also used traditional methods of volatility forecasting is the Historical rolling volatility for 10,30 and 60 days.

While examining what volatility is the best as an input into the neural networks, I have also used the Chicago Board Options Exchange volatility measures RVX and VXN for the respective Russell 2000 and NASDAQ 100 indexes. I will briefly be discussing all of the above in this section.

4.2.1 ARIMA

To explain the working of the ARIMA model works, I will explain the three terms in the name as done in by Bora,2021 :

AutoRegressive AR(p) is a regression model that uses the lagged values of y, until the p-th time in the past, as predictors. Here, $p =$ is the number of lagged observations in the model, ε is white noise at time t, c is a constant and ϕs are parameters.

$$\hat{y}_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \epsilon_t$$

Integrated I(d) - The difference is taken d times until the original series becomes stationary. A stationary time series is one whose properties do not depend on the time at which the series is observed. $By_t = y_{t-1}$ where B is called a backshift operator Thus, a first-order

difference is written as

$$y'_t = y_t - y_{t-1} = (1 - B)y_t$$

In general, a d th-order difference can be written as

$$y'_t = (1 - B)^d y_t$$

Moving average MA(q) - A moving average model uses a regression-like model on past forecast errors. Here, ε is white noise at time t, c is a constant, and θ s are parameters

$$\hat{y}_t = c + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \theta_q \epsilon_{t-q}$$

Combining all of the three types of models above gives the resulting ARIMA(p,d,q) model.

$$\hat{y}'_t = c + \phi_1 y'_{t-1} + \phi_2 y'_{t-2} + \ldots + \phi_p y'_{t-p} + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \theta_q \epsilon_{t-q} + \epsilon_t$$

4.2.2 Historical Volatility:

Different sources may employ somewhat different historical volatility calculations, therefore the same item with the identical settings in different tools may provide different results. I chose the most popular technique, which is to calculate historical volatility as the standard deviation of logarithmic returns based on daily closing prices.Historical Volatility is calculated by finding the average deviation of a financial instrument from its average price. Historical Volatility is calculated in a 10/30/60-day financial rolling window. The 30-day HV is given by:

$$\boxed{\sigma_t = \sqrt{252}\sqrt{\frac{1}{T-1}\sum_{i=30}^{T-1}(r_{t-i} - \bar{r})^2}}$$

4.2.3 RVX & VXN :

The RVX Index as stated in Cboe's website, is a calculation designed to produce a measure of constant, 30-day expected volatility of the U.S. stock market, derived from real-time,

mid-quote prices of Russell 2000 Index (RUT) call and put options.(Shell, 2022) It is released by the Chicago Board Option Exchange(Cboe). The RVX can also be modified to measure the volitility of the NASDAQ 100 index, which is known as VXN.

Both the RVX and VXN are calculated using:

$$\sigma = \sqrt{\frac{2}{T}\sum_{i}\frac{\Delta K_i}{K_i^2}e^{rT}Q(K_i) - \frac{1}{T}[\frac{F}{K_0} - 1]^2}$$

T	Forward Index Level
r	Time-to-Maturity
K_i	Risk-free rate
$Q(K_i)$	Strike Price of the ith out-of-money option
ΔK_i	Mid-point of the bid-ask spread of each option with strike price of K_i
Δ K$_i$	$(K_{i+1} - K_{i-1})/2$

4.3 Data Preprocessing

Before using the data, I will preprocess it to make sure it is fit for training our ANN. We will do the preprocessing in two steps dimensionality reduction and feature scaling.

4.3.1 Feature Scaling

Feature scaling is essential before training any neural network model or any machine learning model in particular. As a lot of the features have different value, for example one feature value might be 100 times bigger than the other feature value, this will lead the model to think that the feature with the larger value is superior to the other and hence the model will not allocate weight accordingly. Feature Scaling essentially sets all of the input variables to the same scale. There are many popular scaling methods like Standardization,Max-Min Normalization (Min-Max scaling) and RobustScaling, I will be discussing the Robust Scaler.

Robust Scaling Algorithms for robust scalers scale features that are robust to outliers.

RobustScaler transforms the feature vector by subtracting the median and then dividing by the interquartile range (75% value — 25% value). The features with huge values, normal to big values, are now on par in scale with the other features having smaller values, just as MinMaxScaler. Notably, unlike MinMaxSc aler, RobustScaler does not scale the data into a specified interval (like 0 and 1). It does not adhere to the exact concept of scale that is applied to the above two scaling methods.[**?** **?**] I can be mathematically stated as:

$$\boxed{x_{robust} = \frac{x_i - Q_1(x)}{Q_3(x) - Q_1(x)}}$$

I have used Robust Scaler during the prepossessing of data during this thesis, as it scales all the features to a similar value and also at the same time prevents the model from learning on any outlying data by reducing the influence of such features or data points.

4.4 Structure of the Deep Learning models

In this section, I will explain the neural network architectures that I have used and what hyperparmeters I have selected to be tuned during the optimization of each of the models. The optimization method used is discussed further in this chapter.

4.4.1 MLP

To begin with, I have used a multilayer perceptron which consists of 4 hidden layers, an input layer and an output dense layer with a single dimensional output. (Armbrust, 2022) As seen in the figure below the input layer has 5 inputs (Close Price/Strike Price , Maturity, Dividend Yield, risk free interest rate (1 M Treasury rate) and one of the forecasted volatilities. The hidden layers are 3 dense layers with 128 nodes and ReLu activation. The output layer is a dense layer with a single-dimensional output, which gives the predicted option price.

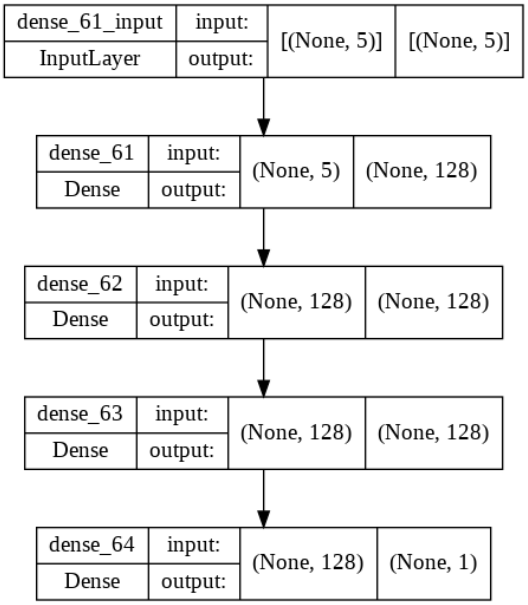

Figure 4.1: Structure of the MLP model used

The table below states the hyper-parameters that will be tuned during optimization of the MLP (the values tested have also been displayed).

Parameter	Values
Hidden Layer Size	{(128,128,128),(128,64,128),(64,128,64),(64,64,64)}
Activation Function	{'tanh','ReLu'}
optimizer	{'SGD', 'Adam'}

4.4.2 CNN

The CNN model examined in the thesis consists of a 1D convolutional layer as the input layer, which takes in 5 inputs same as our input model. But as CNN's work on 3 Dimensional data, I reshape the data that I had by adding a third dimension such that the new dataframe is of dimensions (Data, Data,1) if the original data frame had the dimensions (Data,Data). The CNN consists of 7 hidden layers, 4 of them being Convolutional 1 Dimensional layers, the remaining three are the dropout layer, a dense layer with L2 regularisation and flatten layer, their position in the stack can be seen in the figure below. The network then ends with an output layer which is a dense layer with a 1-dimensional output, which is our predicted price.

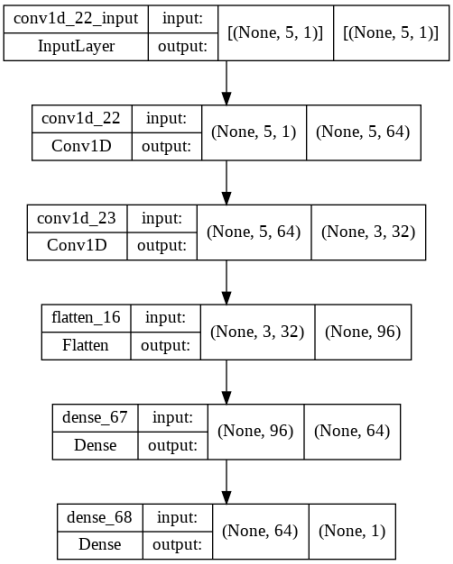

Figure 4.2: Structure of the CNN model used

The hyperparameters tuned during optimizing the CNN were:

Parameter	Values
Batch Size	{512 , 1024 , 2048}
Activation Function	{'ReLu', 'elu', 'tanh'}
optimizer	{'RMSP', 'Adam'}

4.4.3 LSTM

The LSTM model that I created and used to price options has a somewhat simple structure with a LSTM input layer having the same input dimensions as the CNN above, then we have two hidden LSTM layers each with 50 nodes and a Leaky ReLu activation function. The output layer here is also a simple dense layer with a one dimensional output.

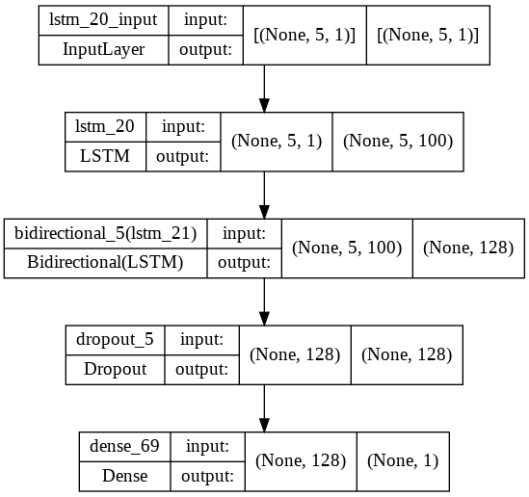

Figure 4.3: Structure of the MLP model used

Below I have mentioned the hyperparameters tuned during the optimization of the LSTM model.

Parameter	Values
Batch Size	{512 , 1024 , 2048}
Activation Function	{'Leaky ReLu', 'elu', 'tanh'}
optimizer	{'RMSP', 'Adam'}

4.4.4 CNN-LSTM

I have used a CNN-LSTM model as a hybrid model to try and outperform all the other models. In the CNN-LSTM model I have used TimeDistributed layers to allow us to use a layer for every input unit, TimeDistributed layer is very useful to work with time series data or video frames.(Mena-Oreja & Gozalvez, 2020) The model has a TimeDistributed input layer, it has 3 hidden layers one TimeDistributed 1 Dimensional Convolutional layer, one TimeDistributed flattens layer and then an LSTM layer to help manage the data in "time". This structure is followed by a dense layer which has a one-dimensional output again being the predicted option price.

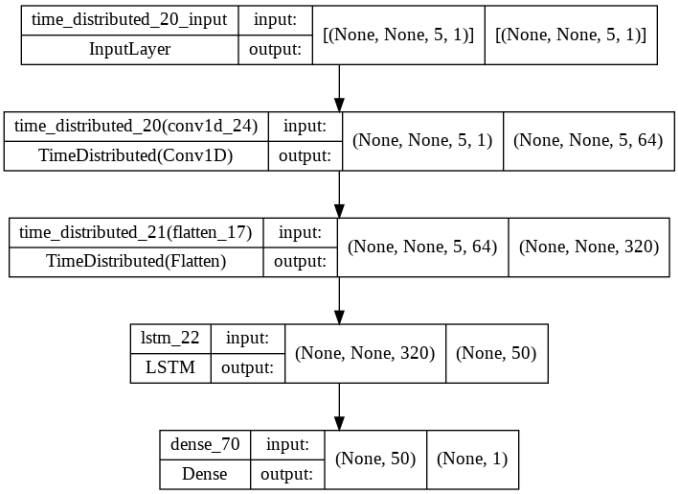

Figure 4.4: Structure of the MLP model used

CNN-LSTM models can be optimized by tuning the following hyperparameters:

Parameter	Values
Batch Size	{512 , 1024 , 2048}
Activation Function	{'Leaky ReLu', 'elu', 'tanh'}
optimizer	{'RMSP', 'Adam'}

4.4.5 Dual Hybrid Model

I have implemented a dual hybrid model, which consists of a traditional mathematical model and a hybrid neural network model. The Merton-Jump-Diffusion model is used as the traditional mathematical model, while the CNN-LSTM neural network has been used as the hybrid neural network. I have used the Merton-Jump-Diffusion model as it compensates for the assumption in the Black Scholes model that the markets and stocks follow a log-normal distribution. The Jump-Diffusion model also tends to find hidden relations and interesting ways to capture future distributions. Combining the two methods the traditional mathematical modelling and neural networks can create better performing methods to price options. [42](Zhao,2022)

The model begins by optimizing the parameters of the Jump-Diffusion Model to predict the option price. The option price predicted by the Jump-Diffusion model is then used to measure the difference between the predicted price and the actual price for each option. This difference in the predicted values is then merged in the dataset of the other input parameters to the CNN-LSTM model in an attempt to capture this information and

make the model learn better.

The structure of the hybrid model is shown in the figure below

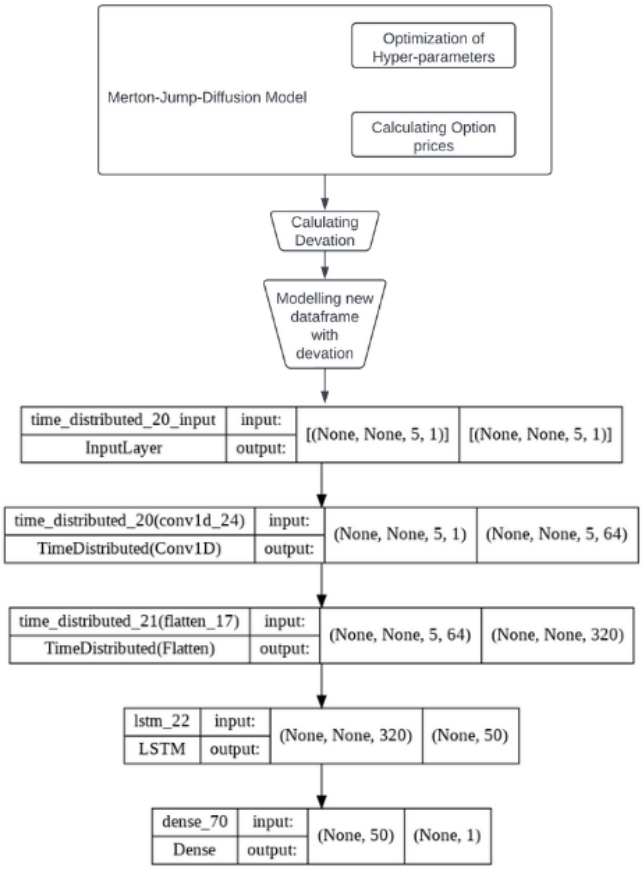

Figure 4.5: Structure of the MLP model used

The hyperparameters that I will try to optimize while tuning this model are going to be:

Parameter	Values
Batch Size	{512 , 1024 , 2048}
Activation Function	{'Leaky ReLu', 'elu', 'tanh'}
optimizer	{'RMSP', 'Adam'}

4.5 Cross-Validation:

Cross-Validation or out-of-sample testing is a process that can be used to validate the performance of a model. Cross-Validation is a re-sampling technique that takes different parts of the data set for testing and training in each iteration. It gives an estimate of how

good the model is on new data. Cross-validation also helps in preventing over-fitting.

I use k-fold-cross-validation which, is a validation strategy in which we divide the data into k-subsets and repeat the holdout procedure k times, with each of the k subsets serving as a test set and the remaining k-1 subsets serving as a training set. The average error from all k trials is then calculated, which is more trustworthy than the traditional handout technique.

The advantages that I get by using k-fold cross-validation are that I am able to reduce bias. I am able to test every data point exactly once and use it in training k-1 times. It also helps to reduce the variance of the resulting estimate as I increase the value of k. Though this comes at a cost, the training algorithm is computationally intensive as the algorithm has to be rerun from scratch k times which results in longer validation times.

4.6 GridSearchCV for Hyper-parameter Tuning

Figuring out the best hyper-parameters for the learning algorithm is known as hyperparameter optimization. A hyperparameter is a parameter whose value is used to control the learning process of the algorithm. In our case with the ANN, our hyper-parameters are going to be the number of perceptrons(nodes), a number of layers, size of the data fed into the network and the number of times the model will be trained. Optimizing these parameters will give us the best-performing model.

To perform hyper-parameter optimization I use GridSearchCV, which is a method which tests the performance of the model for different hyper-parameters in each iteration. The GridSearchCV function will provide us with the hyper-parameters under which our model performs best. I include the 10-fold cross-validation approach within the grid search CV algorithm so that the bias-variance tradeoff is taken into account during hyperparameter tuning.

All of the aforementioned models are passed through GridSearchCV, and the parameters and their respective values are taken into consideration are the same as mentioned while

discussing the structure of the Neural Networks.

In the table below I have mentioned the optimal parameters that are used for evaluating the performance of each model.

Models	Optimal Parameters
MLP	Hidden Layer size = {128,128,128}, Activation function = 'relu', Optimizer = Adam
CNN	Batch Size = {2048}, Activation function = 'relu', Optimizer = Adam
LSTM	Batch Size = {2048}, Activation function = 'elu' , Optimizer = Adam
CNN-LSTM	Batch Size = {2048}, Activation function = 'elu' ,Optimizer = Adam
J-D-LSTM	Batch Size = {2048}, Activation function = 'elu' ,Optimizer = Adam

4.7 Regularisation

Deep learning involves training an artificial neural network using a large set of data samples to generate a model that accurately predicts the outcome of future inputs or events based on the input samples and the relevant features extracted from them. In recent years, artificial neural networks have become widely used in a variety of domains, including computer vision, speech recognition, natural language processing, and financial forecasting. However, in many cases, these models are able to learn features that are not relevant to the real-world problem at hand and can consequently fail to perform as expected on new data samples. For example, a neural network trained on the task of predicting stock prices might learn to categorize historical price changes as bullish or bearish, but this information is not useful to investors since the actual dynamics of the market are not accounted for in the model. As a result, deep-learning models have a reputation for overfitting, a phenomenon in which they learn complex patterns in the training data but cannot generalize well to new examples. Overfitting can cause deep-learning models to be unreliable in real-world settings and lead to inferior performance compared with traditional machine-learning techniques. Regularization is can be considered as a set of techniques to overcome this issue while using neural networks. Regularization focuses on making the neural network less complex to solve the issue of overfitting. I have used Ridge Regression and EarlyStopping as regularization methods.

4.7.1 Ridge Regression - L2

Ridge regression or commonly known as L2 regression, is one of the most popular and widely used regularization methods. It is also known as weight decay. L2 regularization updates the loss function of the neural network with a regularization term (referred to as ω for L2). The values of the weight matrices drop with the addition of this regularisation term since it is assumed that a neural network with smaller weight matrices results in simpler models. Consequently, it will also substantially lessen overfitting.

Mathematically L2 regression can be defined as:

$$\text{New Loss function} = \text{Loss} + \frac{\lambda}{2m} * \sum \|w\|^2$$

λ is the hyper-parameter of the L2 regularization as it controls the amount by which the loss function varies, it can be tuned for better model performance.

4.7.2 EarlyStopping

Adjusting the parameters allows neural networks to minimise the loss function on the training data. But usually, this results in overfitting and reducing the overall performance of the resultant model after training. To prevent this a different set of data is kept as the validation set. As training progresses, the loss function on the validation set is recorded. When the loss function on the validation set stops improving or starts decreasing, EarlyStopping stops the model from training further rather than continuing through all the epochs.

The other advantage that EarlyStooping brings to the table is that it needs significantly less number of epochs to optimize the model while insuring that over-fitting is not happening. This is very useful when it comes to training neural networks on large datasets as it helps save a lot of computation time.

4.8 Metrics

To make out results and analyses easily comparable to previous works and future works, I will use MSE (mean squared error), RMSE (root mean squared error) and R^2 (Coefficient of determination)). I will use MSE to check the performance of the models during training and cross-validation, while RMSE and R^2 are the absolute measures of error, I have used them as the evaluation metrics for comparing the performance of the deep learning model and the BSM model.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$$

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2}$$

$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

Where, $\hat{y}-$ predicted value of y $\bar{y}-$ mean value of y

I have used MSE as the training metric and RMSe and R^2 as the evaluation metric.

Chapter 5

Results and Discussion

In this section, I will be discussing the empirical results that I obtained during the research. This section will be covering the model performance on both the index, the Russell 2000 index has been used as the evaluation index, while I have used the NASDAQ 100 index for checking the generalization of our trained models. I will be using the Russell 2000 index as the main evaluation Index as the Russell 2000 which is administered by London's FTSE Russell Group in London, is often recognised as a barometer of the US economy due to its emphasis on smaller firms focused on the US market.

The codes producing all the results presented in this chapter are available in the following Dropbox folder: Dropbox-to-Python-Notebooks

I will start by discussing how the loss of each of the models changes with each training epoch. I will further compare the performance of each model based on the volatility values they use while training, compare each of them with the Black-Scholes Model and discuss the best volatility for the models. I will also discuss which model performs best on each different volatility type. Further, discussing the performance of the models on the NASDAQ 100 index as an attempt to examine the model's performance on different datasets, i.e to examine its generalizability.

5.1 Neural Network Models

In this section I go over the performance of each model, compare them with the BSM model metrics and which volatility measure is the best for each model. I will then compare the

performance of the benchmark volatility model explicitly with the actual price values.

5.1.1 MLP

The Multi-layer-perceptron being one of the simplest neural network models is the first model I evaluated. The figure below shows how the performance of the MLP model changes while training(I have used MSE as the training metric).

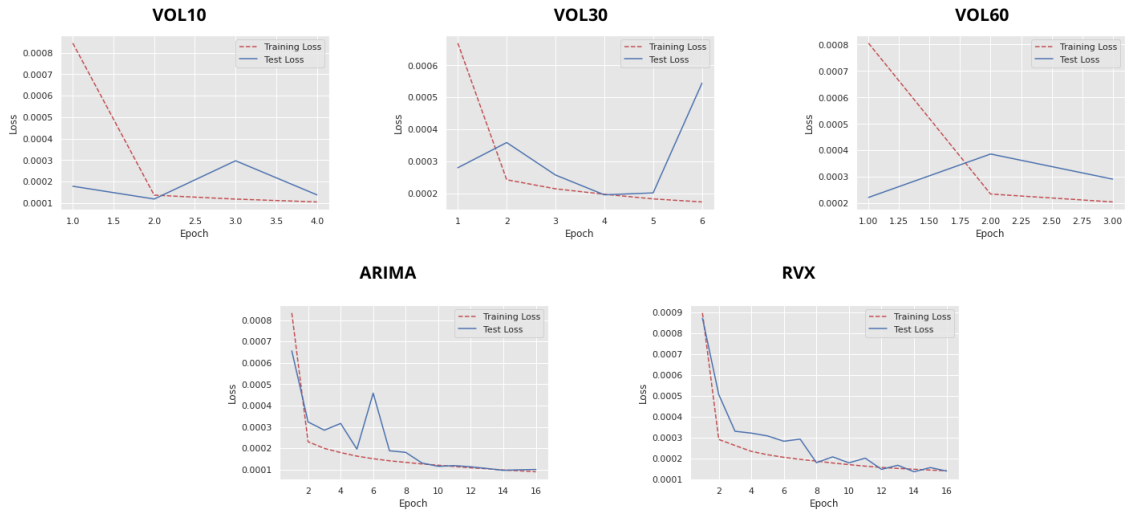

Figure 5.1: Performance of MLP while Training on Different Volatialites

From the graphs above I observe that the ARIMA and RVX volatility models perform the best on the training metric is MSE, but to check the actual performance of the model I will look into the evaluation metric (RMSE & R^2).In the table below the MSE, RMSE and R^2 values of the MLP and BSM model by each volatility is given. The values are compared by the BSM model as it is taken as the benchmark.

	MLP				
	vol10	vol30	vol60	ARIMA	RVX
MSE	1.03E-05	1.20E-05	3.24E-05	2.14E-05	9.54E-06
RMSE	0.003783918404	0.00252847025	0.003774864375	0.0039354327	0.003778832163
R2	0.9992976643	0.9996863994	0.9993010213	0.9992402928	0.9992995511

	BSM				
	vol10	vol30	vol60	ARIMA	RVX
MSE	0.0003653994107	0.0003019758593	0.0002539066955	0.0004277840142	0.000108638585
RMSE	0.01911542337	0.01737745261	0.01593444996	0.02068294017	0.0104229835
R2	0.9820762413	0.9851873258	0.987545239	0.9790161198	0.9946710046

Table 5.1: Comparison of metrics of the MLP and Black-Scholes model

As observed the MLP model despite being the most simple Neural Network architecture, out performs the Black Scholes Model irrespective of the volatility measure used. The best performance of the MLP model when considering the evalution metrics, RMSE and R^2 is when I use the Historical 30 day volatility. This can be considered the benchmark volatility for the MLP model.

The performance of the benchmark MLP model (the model with Vol30 as the volatility measure) as compared to the original prices from 1st Jan 2012 to 30th December 2021 is given below.

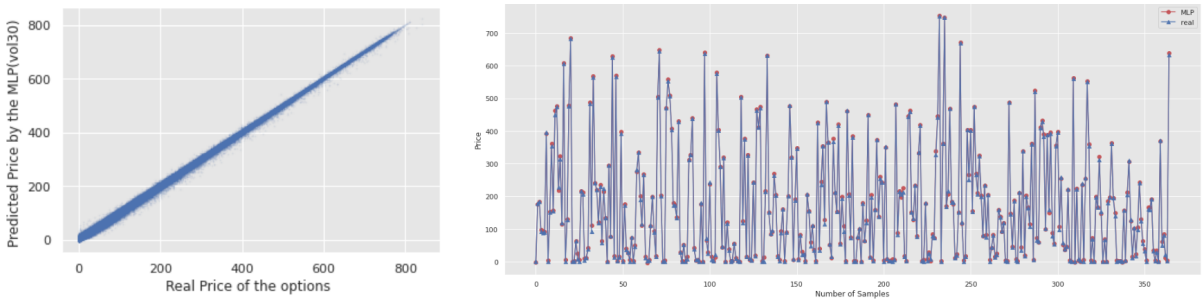

Figure 5.2: Correlation between the MLP(vol30) predicted and real option prices

From the above figure it is evident that the MLP(vol30) model is predicting values that are close to that of the real options. The MLP(vol30) predicts the options prices with the R^2 of 0.9993, but still deviates a little from the original value when the option prices are smaller.

5.1.2 CNN

The second model I used in the research was a CNN model, below I show the training performance of the CNN models (with different volatilities).MSE has been used as the training metric.

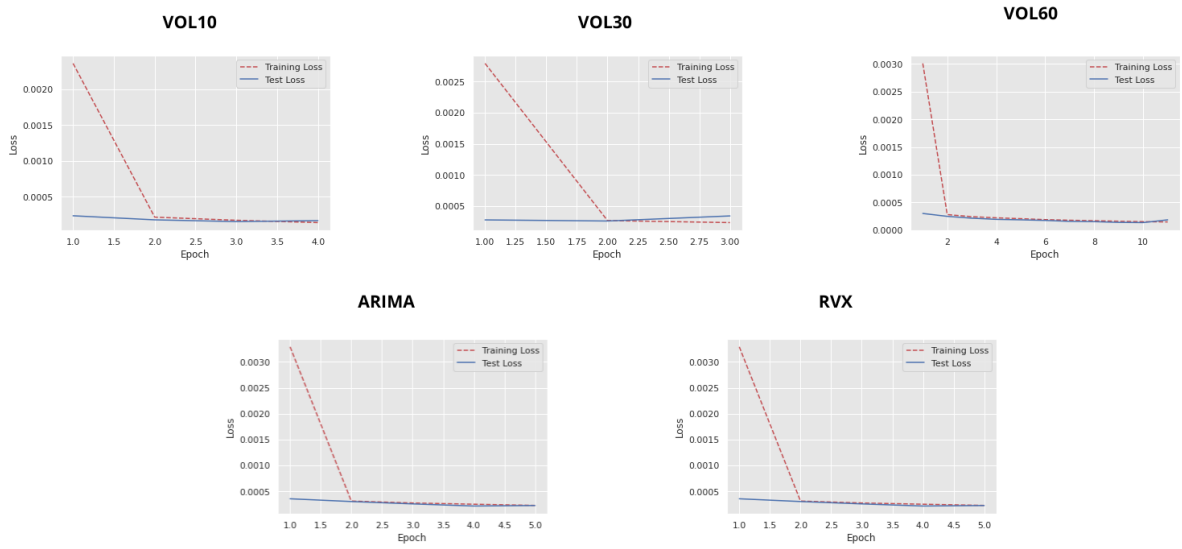

Figure 5.3: Performance of CNN while Training on Different Volatialites

In the above graphs, it can be seen that the CNN model gives almost the same training performance on all the volatility inputs. Because of such cases, I have used seprate metrics for training and evaluation. All the performance metrics of the CNN models and the BSM models with different volatilities.

CNN					
	vol10	vol30	vol60	ARIMA	RVX
MSE	8.46E-06	1.72E-05	9.38E-06	1.37E-05	1.14E-05
RMSE	0.002734643944	0.00280617493	0.00288962549	0.002687657469	0.003255832351
R2	0.9996331717	0.9996137302	0.9995904147	0.9996456691	0.9994800216

BSM					
	vol10	vol30	vol60	ARIMA	RVX
MSE	0.0003653994107	0.0003019758593	0.0002539066955	0.0004277840142	0.000108638585
RMSE	0.01911542337	0.01737745261	0.01593444996	0.02068294017	0.0104229835
R2	0.9820762413	0.9851873258	0.987545239	0.9790161198	0.9946710046

Table 5.2: Comparison of metrics of the CNN and Black-Scholes model

The CNN model gives overall better results(on the basis of all the performance metrics) as compared to the MLP model. I can observe that the CNN model gives the best performance (on the basis of evaluation metrics) when I use the CNN model with the ARIMA forecasted volatility.

The correlation of the predictions of the benchmark CNN model (the model with ARIMA volatility) to the original prices from 1st Jan 2021 to 30th December 2021 is given in the figure below.

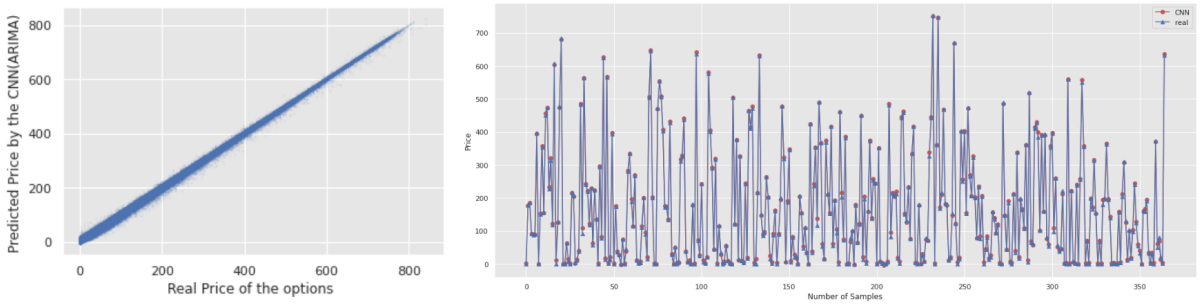

Figure 5.4: Correlation between the CNN(ARIMA) predicted and real option prices

It can be noticed that the deviations in case of the CNN (ARIMA) model are not that huge and the it is visible that the predictions are closer to the actual price as compared to the MLP.

5.1.3 LSTM

The LSTM is a Recurrent Neural Network, the training performance of the LSTM model for each volatility is shown below in the figure.

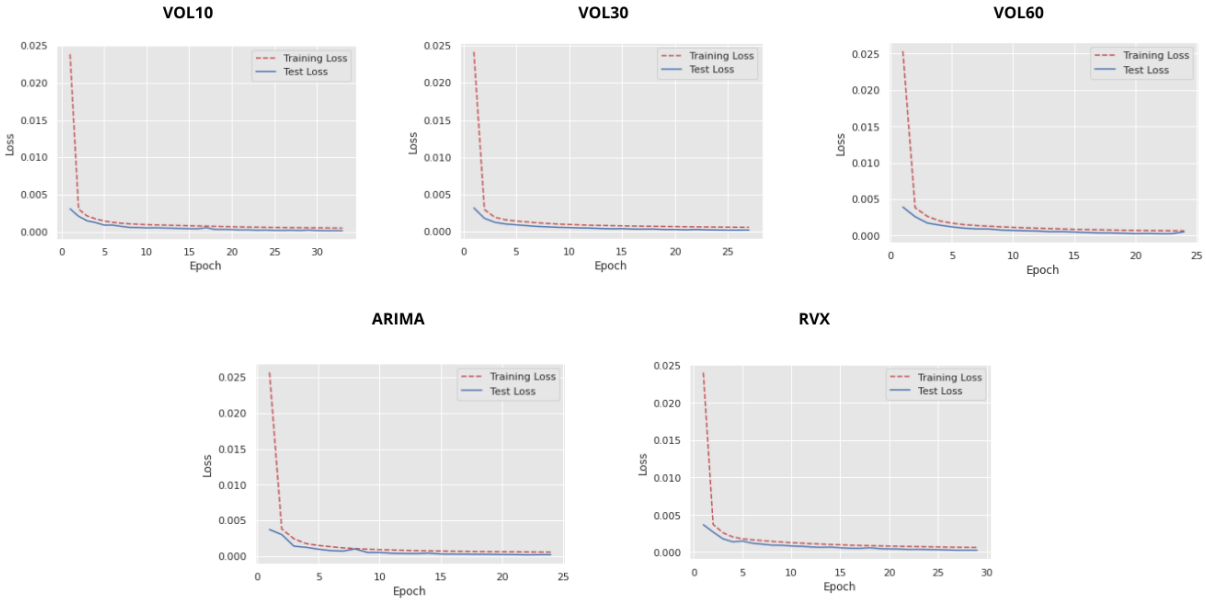

Figure 5.5: Performance of LSTM while Training on Different Volatialites

From the graphs above we observe that the LSTM is not better then the CNN model when it comes to performance while training. In the table below the R^2 and RMSE values of both the LSTM model and BSM model are given.

	LSTM				
	vol10	vol30	vol60	ARIMA	RVX
MSE	7.39E-06	2.19E-05	1.11E-05	4.03E-05	2.52E-05
RMSE	0.00324083143	0.003457693507	0.004034985655	0.00388056737	0.004331042822
R2	0.9994358284	0.9994314594	0.999307846	0.9991043335	0.9993038062

	BSM				
	vol10	vol30	vol60	ARIMA	RVX
MSE	0.0003653994107	0.0003019758593	0.0002539066955	0.0004277840142	0.000108638585
RMSE	0.01911542337	0.01737745261	0.01593444996	0.02068294017	0.0104229835
R2	0.9820762413	0.9851873258	0.987545239	0.9790161198	0.9946710046

Table 5.3: Comparison of metrics of the LSTM and Black-Scholes model

From the above table it can be said that the LSTM model performs the best when 10 Days Historical Volatility is given as the input on the basis of the evaluation metrics RMSE and R^2.

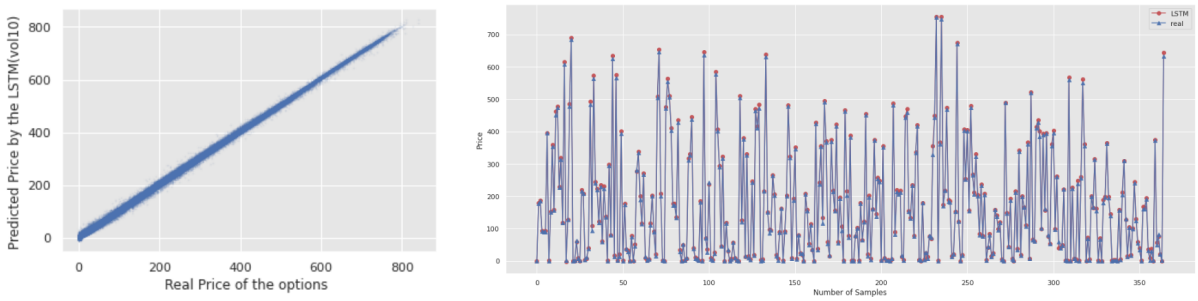

Figure 5.6: Correlation between the LSTM(vol10) predicted and real option prices

5.1.4 Hybrid Model - CNN-LSTM

The CNN-LSTM model is a hybrid Neural Network Architecture. The training performance of the CNN-LSTM model is given below (the loss is MSE as that's the training performance metric).

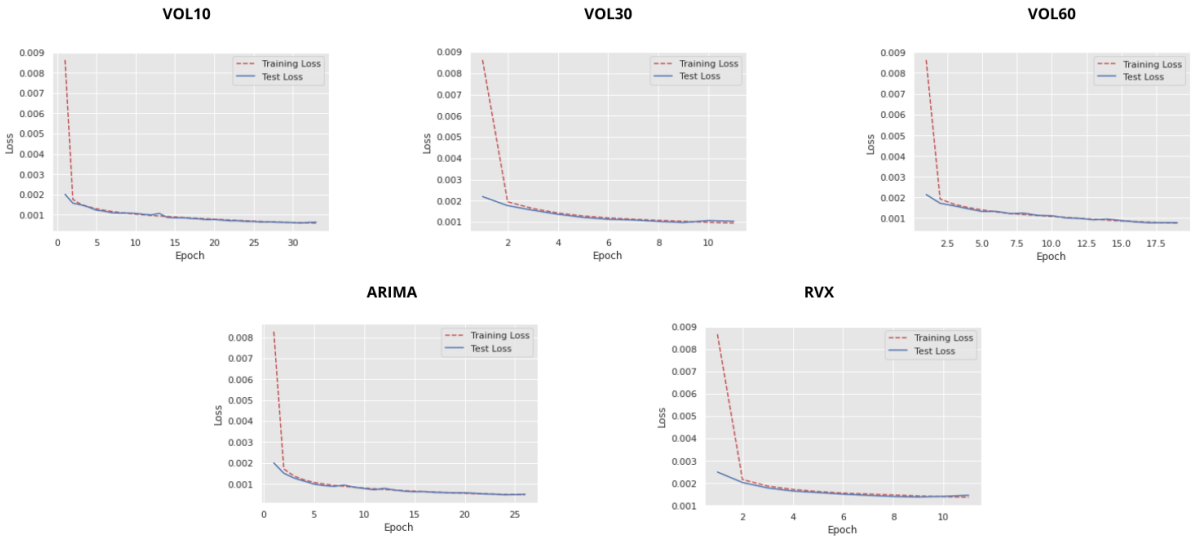

Figure 5.7: Performance of CNN-LSTM while Training on Different Volatialites

The CNN-LSTM model as seen in the above graphs do not perform that good while training as compared to the earlier models. In the table below the RMSE and the R^2 values (the evaluation metrics) are given for the CNN-LSTM models.

	CNN-LSTM				
	vol10	vol30	vol60	ARIMA	RVX
MSE	6.13E-06	7.63E-06	1.01E-05	1.34E-05	1.05E-05
RMSE	0.001776526159	0.002283098155	0.003040389517	0.002546470841	0.002813349564
R2	0.9998451881	0.999744312	0.9995465601	0.9996819183	0.9996117525

	BSM				
	vol10	vol30	vol60	ARIMA	RVX
MSE	0.0003653994107	0.0003019758593	0.0002539066955	0.0004277840142	0.000108638585
RMSE	0.01911542337	0.01737745261	0.01593444996	0.02068294017	0.0104229835
R2	0.9820762413	0.9851873258	0.987545239	0.9790161198	0.9946710046

Table 5.4: Comparison of metrics of the CNN-LSTM and Black-Scholes model

The CNN-LSTM model has the best evalutaion performance while using the 10 Day Historical Volatility. The CNN-LSTM model while using the 10 day historical volatility has the best performance as compared to all of the existing models.

The correlation of the predictions of the benchmark CNN-LSTM model(with vol10) and the actual prices from 1st January 2021 to 30th December 2021, along with the predictions and original values are given in the figure below:

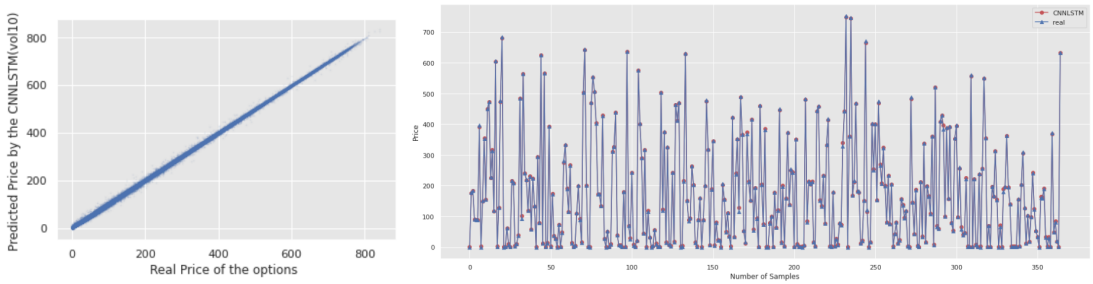

Figure 5.8: Correlation between the CNN-LSTM(vol10) predicted and real option prices

5.1.5 Dual Hybrid Model - Jump-Diffusion CNN-LSTM

In the Dual-Hybrid Model I have used a traditional mathematical model that is the Merton-Jump-diffusion Model and a hybrid CNN-LSTM Neural Network Architecture. The training performance of the model is given below (loss being MSE):

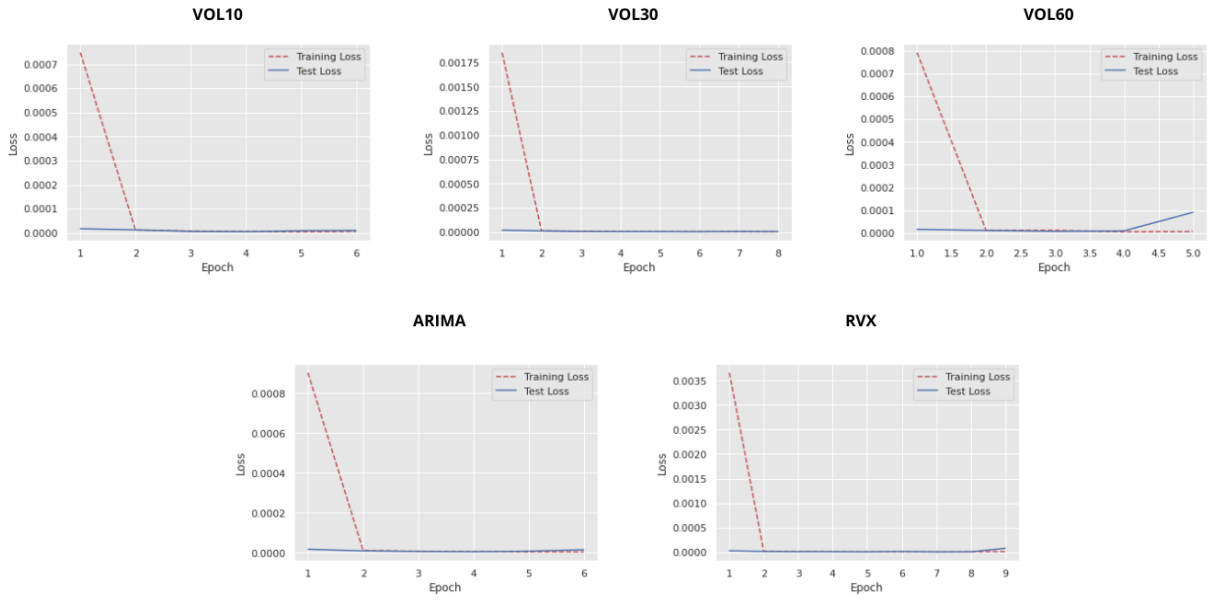

Figure 5.9: Performance of the Jump-Diffusion-CNN-LSTM while Training on Different Volatialites

As observed in the graphs above, the Jump-Diffusion-CNN-LSTM model gives the best training performance out of all the models while training. In the table below we look at the evaluataion metrics of both the dual hybrid model and the BSM model.

As observed in the table, despite of having the best performance on the training loss metric the dual hybrid model does not have a better performance than any other

	Merton-Jump-Diffusion CNN-LSTM				
	vol10	vol30	vol60	ARIMA	RVX
MSE	2.39E-06	4.73E-06	1.36E-06	4.05E-06	5.97E-06
RMSE	0.10544621606	0.12175284277	0.13690755528	0.10011594475	0.10443197855
R2	0.369887584	0.3037770457	0.2723581783	0.27598093378	0.2297187445

	BSM				
	vol10	vol30	vol60	ARIMA	RVX
MSE	0.0003653994107	0.0003019758593	0.0002539066955	0.0004277840142	0.000108638585
RMSE	0.01911542337	0.01737745261	0.01593444996	0.02068294017	0.0104229835
R2	0.9820762413	0.9851873258	0.987545239	0.9790161198	0.9946710046

Table 5.5: Comparison of metrics of the Dual-Hybrid and Black-Scholes model

neural network. This is due to the fact that the dual hybrid model takes a large amount of time to run for normal size datasets, the Russell 2000 and NASDAQ 100 options from the 2012 to 2022 is a very large dataset, hence I was not able to train the model for all the strike prices. This is what is causing the issue that the model is performing very good in training but is not able to reproduce the same results when it comes to predicting the prices for all the options with different strike prices.

The correlation of the predicted values of the benchmark Jump-Diffusion CNN-LSTM (with vol60 as input) and the real values from 1st Jan 2012 to 30th December 2021, along with the predictions and the original values are given in the figure below:

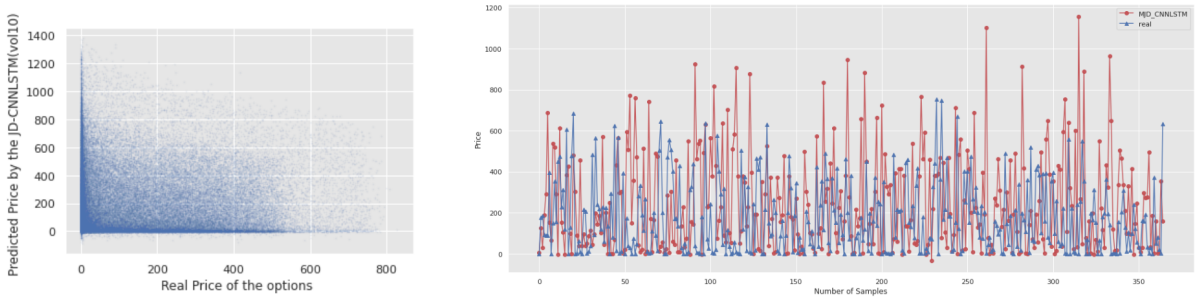

Figure 5.10: Correlation between the Jump-Diffusion-CNN-LSTN(vol60) predicted and real option prices

It can clearly be seen that the dual-hybrid model is not performing well due to getting trained on less strike prices.

5.2 Volatilies

In this section, I will be examining what neural network models perform the best for each volatility.

5.2.1 ARIMA

In the table below I have given all the values of all the metrics for all deep learning models while using volatility forecasted by the ARIMA model as the measure of volatility.

	ARIMA		
Deep Learning Model	MSE	RMSE	R2
MLP	2.14E-05	0.0039354327	0.9992402928
CNN	1.37E-05	0.002687657469	0.9996456691
LSTM	4.03E-05	0.00388056737	0.9991043335
CNN-LSTM	1.34E-05	0.002546470841	0.9996819183
Jump-Diffusion CNN-LSTM	4.05E-06	0.10011594475	0.27598093378

Table 5.6: Comparison of how each model performs using the ARIMA forecasted volatility

As observed in the table the table above CNN-LSTM model gives the best performance for the volatility forecasted by the ARIMA model.

5.2.2 10 Day Historical(vol10)

In the table below I have given all the value of all the metrics for all deep learning models while using 10-day Historical volatility as the measure of volatility.

	vol10		
Deep Learning Model	MSE	RMSE	R2
MLP	1.03E-05	0.003783918404	0.9992976643
CNN	8.46E-06	0.002734643944	0.9996331717
LSTM	7.39E-06	0.00324083143	0.9994358284
CNN-LSTM	6.13E-06	0.001776526159	0.9998451881
Jump-Diffusion CNN-LSTM	2.39E-06	0.10544621606	0.369887584

Table 5.7: Comparison of how each model performs using the 10 Day historical volatility (vol10)

As observed in the table above the CNN-LSTM model again, gives the best performance for vol10.

5.2.3 30 Day Historical(vol30)

In the table below I have given all the value of all the metrics for all deep learning models while using 30-day Historical volatility as the measure of volatility.

	vol30		
Deep Learning Model	MSE	RMSE	R2
MLP	1.20E-05	0.00252847025	0.9996863994
CNN	1.72E-05	0.00280617493	0.9996137302
LSTM	2.19E-05	0.003457693507	0.9994314594
CNN-LSTM	7.63E-06	0.002283098155	0.999744312
Jump-Diffusion CNN-LSTM	4.73E-06	0.12175284277	0.3037770457

Table 5.8: Comparison of how each model performs using the 30 Day historical volatility (vol30)

As observed in the table above, the CNN-LSTM model again gives the best performance when using the 30-day historical volatility as the input.

5.2.4 60 Day Historical(vol60)

In the table below I have given all the value of all the metrics for all deep learning models while using 60-day Historical volatility as the measure of volatility.

	vol60		
Deep Learning Model	MSE	RMSE	R2
MLP	3.24E-05	0.003774864375	0.9993010213
CNN	9.38E-06	0.00288962549	0.9995904147
LSTM	1.11E-05	0.004034985655	0.999307846
CNN-LSTM	1.01E-05	0.003040389517	0.9995465601
Jump-Diffusion CNN-LSTM	1.36E-06	0.13690755528	0.2723581783

Table 5.9: Comparison of how each model performs using the 60 Day historical volatility (vol60)

As observed in the table above the CNN model gives the best performance when using the 60-day historical volatility as the input. It is a surprise to not see the CNN-LSTM model as the best performer as it has been for the previous three.

5.2.5 RVX

In the table below I have given all the values of all the metrics for all deep learning models while using RVX the CBOE's Volatility Index for Russell 2000 as the measure of volatility.

	RVX		
Deep Learning Model	MSE	RMSE	R2
MLP	9.54E-06	0.003778832163	0.9992995511
CNN	1.14E-05	0.003255832351	0.9994800216
LSTM	2.52E-05	0.004331042822	0.9993038062
CNN-LSTM	1.05E-05	0.002813349564	0.9996117525
Jump-Diffusion CNN-LSTM	5.97E-06	0.10443197855	0.2997187445

Table 5.10: Comparison of how each model performs using the CBOE's RVX volatility for Russell 2000

As observed in the table above, the CNN-LSTM model again gives the best performance

when using the RVX volatility Index as the input.

5.3 Figuring out the best model

In this section, I will compare all the benchmark models and see which one of them performs the best for pricing options. I will not be comparing them with the Black-Scholes model (BSM) as all of these models have outperformed the BSM model as discussed above. I think that the CNN LSTM model is going to be the best performer as it was the model that performed the best on the majority of volatility. The table below has the performance metrics for all the benchmark models.

	MLP(vol30)	CNN(ARIMA)	LSTM(vol10)	CNN-LSTM(vol10)	JD-CNN-LSTM(vol60)
MSE	1.20E-05	1.37E-05	7.39E-06	6.13E-06	1.36E-06
RMSE	0.00252847025	0.002687657469	0.00324083143	0.001776526159	0.10690755528
R2	0.9996863994	0.9996456691	0.9994358284	0.9998451881	0.3223581783

Table 5.11: Comparison of all the benchmark models, (no BSM values as they outperform the BSM model)

As seen in the above table we can see that the benchmark CNN-LSTM model is the best performing model when it comes to pricing options on the Russell 20000 Index, as it has both the min RMSE and the min R2 values (I only consider RMSE and R2 as I take them as the evaluation metrics).

5.4 Genralizablity

In this section, I will cover the performance of all our benchmark models on the NASDAQ 100 Index (NDX) to see if the CNN-LSTM model is still the best model for NDX options too. If the CNN-LSTM and the rest of the models can produce similar results the models can be considered general models, and can be applied to other European style options as well.

In the table below I have given the MSE, RMSE and R^2 values for the benchmark models, along with the metrics for the Black-Scholes model (BSM) model while pricing on the NASDAQ 100 Index data.

	NASDAQ 100				
	MLP(vol30)	CNN(ARIMA)	LSTM(vol10)	CNN-LSTM(vol10)	JD-CNN-LSTM(vol60)
MSE	9.64E-06	6.44E-06	1.18E-05	5.96E-06	0.01151520725
RMSE	0.003105268979	0.002537112303	0.003442045068	0.002440934785	0.1073089337
R2	0.9993740777	0.9995821684	0.9992309492	0.9996132465	0.2681587973

	BSM(vol10)	BSM(vol30)	BSM(vol60)	BSM(ARIMA)	BSM(VXN)
MSE	0.0003711885166	0.0002978733706	0.0002339163701	0.0005146622899	9.57E-05
RMSE	0.01926625331	0.01725900839	0.01529432477	0.02268616957	0.009783024798
R2	0.9759055799	0.9806645793	0.984816127	0.9665924756	0.9937874735

Table 5.12: Comparison of how each benchmark model and the Black-Scholes model performs using the NASDAQ 100 index data

As observed in the above table the benchmark model for pricing options on the NASDAQ 100 Index is also the CNN-LSTM(vol10) model, and all the models outperform the Black-Scholes Model (BSM).

In the figure below I plot the correlation between the CNN-LSTM vol10 model's predicted prices and the real prices, as well as both the values for a slice of the total samples.

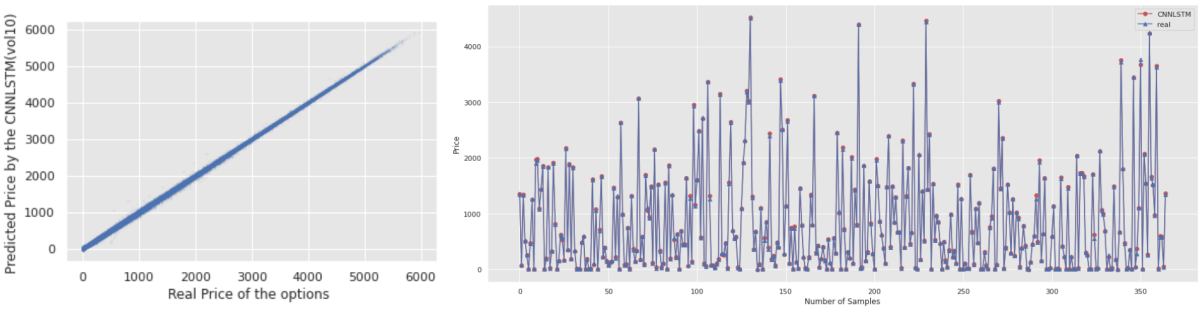

Figure 5.11: Performance of the Jump-Diffusion-CNN-LSTM while Training on Different Volatialites

As we can see from the figures above CNN-LSTM vol10 model predicts the prices pretty close to the real prices (to be precise with a 99.95% chance of being correct). I can say that the models do perform fairly well and hence can be said to be generalised.

5.5 Discussion

In this section, I will be discussing the empirical results that I have obtained above. Starting with the first research question, can deep learning models perform better than Black-Scholes when it comes to pricing European options on the Russell 2000 (given by London's FTSE Russell Group) and the NASDAQ 100 index? The results clearly prove that the deep learning methods outperform the Black-Scholes model when it comes to pricing options. The only Deep Learning model that was not able to perform better than the Black-Scholes model is the Jump-Diffusion CNN-LSTM model and that is because of the fact that it has not been trained on all the strike prices as it would take a lot of time(explained above while discussing its results).

Coming to the second question of my research, Can dual-hybrid deep learning models perform better than the rest of the models? In my research, I have not been able to train the dual-hybrid model on the entire dataset due to limitations of time and hence, it has not outperformed any neural network or for that matter even the Black-Scholes Model. But I do believe that if trained on the same amount of data as the rest of the models, it will outperform them (it can be seen from the validation loss value which is less than any other model while training).

The third research question that I wanted to answer was, whether using the volatility forecasted by the ARIMA model helps the neural networks perform better. What I found out from the results prove that the volatility forecasted by the ARIMA model may help some deep learning model (in our case the CNN model) perform better than they would with any other volatility, but it is by no means the best volatility measure as 4 out 5 models give better performance while using other volatility measures.

The last question was to figure out which volatility measure allows the model to perform the best. As per the results, the CNN-LSTM model which used the 10-day historical volatility (vol10) performs the best while pricing options on both the indexes. I think the reason behind this will be the fact that 10-day volatility is the measure which is updated most frequently updated, due to its short measuring span and that the volatility measure only takes into account the last 10 days and hence might be providing a better volatility measure for helping the models learn from the change is volatility faster than the other methods. But I won't say that vol10 should be the only measure used as some other model might perform better with other volatilities.

5.6 Limitations

In this research I compare the performance of the Deep Learning methods when it comes to pricing options with the Black-Scholes model, which only prices European options, limiting the scope of this research to only European-style options.

Also when it comes to Volatility forecasting there are a lot of different methods like HIST (historical average volatility), EMWA (Exponentially Weighted Moving Average), GARCH, CALM etc. But I use only HIST volatility, the COBE's Index volatilities and the ARIMA model to forecast volatilities. Hence, this research cannot conclude that vol10 will be the best measure of volatility to give as a training input for Deep Learning Models.

Deep Learning Models have a lot of different structures and can be set up in infinitely many architectures, this architecture along with other variables affects the performance of the Deep learning model. Though, this thesis covers different types of deep learning architectures from Multi-layer-perceptrons to CNNs and LSTMs as well as hybrid (CNN-LSTM) and dual-hybrid models (the Jump-Diffusion CNN-LSTM). Despite using all these models, there are definitely more models and architectures of the same one as well, which limits the research from looking into all possible options available and selecting the best possible.

Chapter 6

Conclusion and Recommendations

6.1 Conclusion

In this research, I used Deep Learning models MLP, CNN, LSTM, CNN-LSTM and a dual hybrid model (Jump diffusion CNN-LSTM) to predict option prices of the Russell 2000 and NASDAQ 100 Index from 1st Jan 2012 to 30th Dec 2021.

I start by using the ARIMA model to forecast the volatility from 1st Jan 2012 to 30th Dec 2021, as I train each model on different sets of volatilities in an attempt to see if the ARIMA model produces better results.

I further train and optimize the models using cross-validation and GridSearchCV. Then I proceed to calculate the evaluation metrics for each of the different Deep Learning models and the Black-Scholes model.

The research found out the CNN-LSTM model when used with 10-day Historical volatility is the best model when it comes to predicting the prices of European-style call options. The research also found that using the volatility forecasted by the ARIMA model was not the best input for Deep Learning models. Though the Dual Hybrid Jump-Diffusion CNN-LSTM model didn't have great results, it is a promising model for the future as I was only able to train it on significantly less amount of data because of the time it takes to train. As far as the best volatility is concerned it cannot be determined by my research as I did not cover all the popular volatility forecasting methods as my research was more focused on the Deep Learning aspect.

I find out that the models are fairly generalised as they perform fairly well if not better on other datasets (in our case the NASDAQ 100). And as I assume the put-call parity to be true, it can be said that our models will price European-style options, put and call both, while also on multiple indexes as the models are pretty generalised.

As a whole, this research aims to suggest a dual hybrid deep learning model, with a traditional mathematical model and a hybrid deep learning model. A Jump-Diffusion CNN-LSTM model was the outcome.

6.2 Future Research

As I have discussed in the results section, despite of not so good performance of the dual hybrid model- the Jump-Diffusion CNN-LSTM model, I find the model to be promising, as I did only run it on a fraction of the original dataset as it takes a lot of time to train. But as far as training is considered it was producing the best MSE values out of all the models for training and validation during train. As it was not trained on the entirety of the dataset its performance cannot be truly justified by the metrics that I calculate related to the model. There is potential in the dual hybrid model, there might be another mathematical model instead of the Jump-Diffusion model which produces better results but dual hybrid models are worth giving a shot at and have a lot of potential.

Bibliography

[1] Explains how to proceed with the data as it is overwhelmingly large. 4

[2] Intuition of adam optimizer, Oct 2020. URL https://www.geeksforgeeks.org/intuition-of-adam-optimizer/.

[3] E. Alpaydin. *Introduction to machine learning*. MIT Press, 2014. 18

[4] H. Amilon. A neural network versus black-scholes: A comparison of pricing and hedging performances. *Journal of Forecasting*, 22(4):317–335, 2003. doi: 10.1002/for.867. 4

[5] F. Armbrust. Deep sustainable finance : An end-to-end text analysis of the financial and environmental narratives in corporate disclosures, Jan 2022. URL http://dx.doi.org/10.18419/opus-12270.

[6] Baheti. Activation functions in neural networks [12 types amp; use cases]. *V7*, 2022. URL https://www.v7labs.com/blog/neural-networks-activation-functions. 27, 28, 29, 30

[7] M. Choudhry. Fixed income securities and derivatives handbook. 2010. doi: 10.1002/9781118531976.

[8] A. K. Dohare and Tulika. A cnn and lstm-based data prediction model for wsn. *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, 2021. doi: 10.1109/icac3n53548.2021.9725465. 18, 26

[9] S. Doshi. Various optimization algorithms for training neural network, Aug 2020. URL https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6. 31

[10] B. Dumas, J. Fleming, and R. Whaley. Implied volatility functions: Empirical tests.

IEEE/IAFE 1996 Conference on Computational Intelligence for Financial Engineering (CIFEr), 1998. doi: 10.1109/cifer.1996.501845. 2

[11] N. Gogia. Why scaling is important in machine learning?, May 2022. URL https://medium.com/analytics-vidhya/why-scaling-is-important-in-machine-learning-aee5781d161a.

[12] N. Gradojevic and M. Caric. Revisiting non-parametric exchange rate prediction. *Journal of Applied Business Research (JABR)*, 25(2), 2011. doi: 10.19030/jabr.v25i2.1038. 5

[13] A. Hayes. What is the black-scholes model?, Jul 2022. URL https://www.investopedia.com/terms/b/blackscholes.asp. 1

[14] A. Hirsa, T. Karatas, and A. Oskoui. Supervised deep neural networks (dnns) for pricing/calibration of vanilla/exotic options under various different processes, 2019. URL https://arxiv.org/abs/1902.05810. 24

[15] J. C. Hull. *Options, futures, and other derivatives*. Pearson, 2015. 8

[16] J. M. HUTCHINSON, A. W. LO, and T. POGGIO. A nonparametric approach to pricing and hedging derivative securities via learning networks. *The Journal of Finance*, 49(3):851–889, 1994. doi: 10.1111/j.1540-6261.1994.tb00081.x. 2, 4

[17] J.-Y. Joo, S.-H. Ahn, Y. T. Yoon, and J.-W. Choi. Option valuation applied to implementing demand response via critical peak pricing. *2007 IEEE Power Engineering Society General Meeting*, 2007. doi: 10.1109/pes.2007.385559.

[18] M. D. Joshi. *The concepts and practice of Mathematical Finance*. Cambridge University Press, 2003. 8

[19] P. Lajbcygier. Improving option pricing with the product constrained hybrid neural network. *IEEE Transactions on Neural Networks*, 15(2):465–476, 2004. doi: 10.1109/tnn.2004.824265. 4

[20] T. Lee, V. P. Singh, and K. H. Cho. Deep learning for hydrometeorology and environmental science. *Water Science and Technology Library*, 2021. doi: 10.1007/978-3-030-64777-3. 18

[21] L. Liang and X. Cai. Time-sequencing european options and pricing with deep learning – analyzing based on interpretable ale method. *Expert Systems with Applications*, 187:115951, 2022. doi: 10.1016/j.eswa.2021.115951. 8

[22] S. Liu, C. Oosterlee, and S. Bohte. Pricing options and computing implied volatilities using neural networks. *Risks*, 7(1):16, 2019. doi: 10.3390/risks7010016. 5

[23] W. Lu, J. Li, Y. Li, A. Sun, and J. Wang. A cnn-lstm-based model to forecast stock prices. *Complexity*, 2020:1–10, 2020. doi: 10.1155/2020/6622927. 26

[24] H. Marzi and M. Turnbull. Use of neural networks in forecasting financial market. *2007 IEEE International Conference on Granular Computing (GRC 2007)*, 2007. doi: 10.1109/grc.2007.78. 8

[25] J. Mena-Oreja and J. Gozalvez. A comprehensive evaluation of deep learning-based techniques for traffic prediction. *IEEE Access*, 8:91188–91212, 2020. doi: 10.1109/access.2020.2994415. 18

[26] F. Mostafa, T. S. Dillon, and E. Chang. *Computational Intelligence Applications to Option Pricing, Volatility Forecasting and Value at Risk*, volume 697 of *Studies in Computational Intelligence*. Springer, 2017. ISBN 978-3-319-51666-0. doi: 10.1007/978-3-319-51668-4. URL https://doi.org/10.1007/978-3-319-51668-4. 18

[27] A. Muneer, S. Mohd Taib, S. Mohamed Fati, A. O. Balogun, and I. Abdul Aziz. A hybrid deep learning-based unsupervised anomaly detection in high dimensional data. *Computers, Materials amp; Continua*, 70(3):5363–5381, 2022. doi: 10.32604/cmc.2022.021113. 32

[28] R. Paolucci. Deriving the black-scholes model, Apr 2020. URL https://medium.com/swlh/deriving-the-black-scholes-model-5e518c65d0bc. 8

[29] H. Park, N. Kim, and J. Lee. Parametric models and non-parametric machine learning models for predicting option prices: Empirical comparison study over kospi 200 index options. *Expert Systems with Applications*, 41(11):5227–5237, 2014. doi: 10.1016/j.eswa.2014.01.032. 5

[30] M. H. Petersen. *Option Pricing Using Artificial Neural Networks*. PhD thesis, 2019. 4, 8, 18

[31] G. Poitras. The early history of option contracts. *Vinzenz Bronzin's Option Pricing Models*, page 487–518, 2009. doi: 10.1007/978-3-540-85711-2_24.

[32] Prakhar. Intuition of adam optimizer, Oct 2020. URL https://www.geeksforgeeks.org/intuition-of-adam-optimizer/. 33

[33] S. Sawant, A. Vishwakarma, P. Sawant, and P. Bhavathankar. Analytical and sentiment based text generative chatbot. *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 2021. doi: 10.1109/icccnt51525.2021.9580069. 18, 32

[34] J. Selig. What is machine learning? a definition., Jul 2022. URL https://www.expert.ai/blog/machine-learning-definition/. 18

[35] M. Shell. Asymmetric risk reward, Aug 2022. URL https://asymmetryobservations.com/category/asymmetric-risk-reward/.

[36] C. Shi. Brownian motion, ito's lemma, and the black-scholes formula (part ii), Jun 2019. URL https://www.linkedin.com/pulse/brownian-motion-itos-lemma-black-scholes-formula-part-chuan-shi-1d/. 8

[37] R. Uhrig. Introduction to artificial neural networks. *Proceedings of IECON '95 - 21st Annual Conference on IEEE Industrial Electronics*, 1995. doi: 10.1109/iecon.1995.483329. 19

[38] G. Vainberg. Option pricing models and volatility. 2012. doi: 10.1002/9781119202097.

[39] B. WASHBURN and M. DİK. Derivation of black-scholes equation using itô's lemma. *Proceedings of International Mathematical Sciences*, 2021. doi: 10.47086/pims.956201.

[40] J. Witzany. Derivatives. *Springer Texts in Business and Economics*, 2020. doi: 10.1007/978-3-030-51751-9. 17

[41] Y. Yin and P. G. Moffatt. Correcting the bias in the practitioner black-scholes method. *Journal of Risk and Financial Management*, 12(4):157, 2019. doi: 10.3390/jrfm12040157.

[42] K. Zhang and K. Lay Teo. A penalty-based method from reconstructing smooth local volatility surface from american options. *Journal of Industrial amp; Management Optimization*, 11(2):631–644, 2015. doi: 10.3934/jimo.2015.11.631. 18, 43

[43] K. Zhao, J. Zhang, and Q. Liu. Dual-hybrid modeling for option pricing of csi 300etf. *Information*, 13(1), 2022. doi: 10.3390/info13010036. 3